Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi, *Senior Member, IEEE*

***Abstract*—Reinforcement learning (RL) algorithms have been around for decades and employed to solve various sequential decision-making problems. These algorithms, however, have faced great challenges when dealing with high-dimensional environments. The recent development of deep learning has enabled RL methods to drive optimal policies for sophisticated and capable agents, which can perform efficiently in these challenging environments. This article addresses an important aspect of deep RL related to situations that require multiple agents to communicate and cooperate to solve complex tasks. A survey of different approaches to problems related to multiagent deep RL (MADRL) is presented, including nonstationarity, partial observability, continuous state and action spaces, multiagent training schemes, and multiagent transfer learning. The merits and demerits of the reviewed methods will be analyzed and discussed with their corresponding applications explored. It is envisaged that this review provides insights about various MADRL methods and can lead to the future development of more robust and highly useful multiagent learning methods for solving real-world problems.**

***Index Terms*—Continuous action space, deep learning, deep reinforcement learning (RL), multiagent, nonstationary, partial observability, review, robotics, survey.**

I. Introduction

REINFORCEMENT learning was instigated by a trial-and-error (TE) procedure, conducted by Thorndike in an experiment on cat's behaviors in 1898 [1]. In 1954, Minsky [2] designed the first neural computer called stochastic neural-analog reinforcement calculators (SNARCs), which simulated the rat's brain to solve the maze puzzle. SNARCs remarked the uplift of TE learning to a computational period. Almost two decades later, Klopf [3] integrated the mechanism of temporal-difference (TD) learning from psychology into the computational model of TE learning. That integration succeeded in making TE learning a feasible approach to large systems. In 1989, Watkins and Dayan [4] brought the theory of optimal control [5] including the *Bellman equation* and Markov decision process (MDP) together with TD learning to form a well-known *Q-learning*. Since then, *Q*-learning has been applied to solve various real-world problems, but it is unable to solve high-dimensional problems where the number of calculations increases drastically with the number of inputs. This problem, known as the *curse of dimensionality*, exceeds the computational constraint of conventional computers. In 2015, Mnih *et al.* [6] made an important breakthrough by combining *deep learning* with reinforcement learning (RL) to partially overcome the curse of dimensionality. The deep RL has attracted significant attention from the research community since then. Milestones of the development of RL are presented in Fig. 1, which span the TE method to the deep RL.

The RL originates from animal learning in psychology and thus it can mimic the human learning ability to select actions that maximize long-term profit in their interactions with the environment. The RL has been widely used in robotics and autonomous systems, e.g., Mahadevan and Connell [7] designed a robot that can push cubes (1992); Schaal [8] created a humanoid robot that can effectively solve the pole-balancing task (1997); Benbrahim and Franklin [9] made a biped robot that can learn to walk without any knowledge of the environment (1997); Riedmiller *et al.* [10] built a soccer robot team (2009); and Mulling *et al.* [11] trained a robot to play table tennis (2013).

Modern RL is truly marked by the success of the deep RL in 2015 when Mnih *et al.* [6] made use of a structure named deep *Q*-network (DQN) in creating an agent that outperformed a professional player in a series of 49 classic Atari games [12]. In 2016, Google's DeepMind created a self-taught AlphaGo program that could beat the best professional Go players, including China's Ke Jie and Korea's Lee Sadol [13]. Deep RL has also been used to solve MuJoCo physics problems [14] and 3-D maze games [15]. In 2017, OpenAI announced a bot that could beat the best professional gamer on the online game Dota 2, which is supposed to be more complex than the Go game. More importantly, deep RL has become a promising approach to solving real-world problems due to its practical methodology, e.g., optimal control of nonlinear systems [16], pedestrian regulation [17], or traffic grid signal control [18]. Enterprise corporations, such as Google, Tesla, and Uber, have been in a race to make self-driving cars. Moreover, recent

Manuscript received February 7, 2019; revised July 11, 2019, October 18, 2019, and December 15, 2019; accepted February 25, 2020. This article was recommended by Associate Editor D. Liu. *(Corresponding author: Thanh Thi Nguyen.)*

Thanh Thi Nguyen is with the School of Information Technology, Deakin University (Burwood Campus), Burwood, VIC 3125, Australia (e-mail: thanh.nguyen@deakin.edu.au).

Ngoc Duy Nguyen and Saeid Nahavandi are with the Institute for Intelligent Systems Research and Innovation, Deakin University (Waurn Ponds Campus), Waurn Ponds, VIC 3216, Australia (e-mail: duy.nguyen@deakin.edu.au; saeid.nahavandi@deakin.edu.au).

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

Digital Object Identifier 10.1109/TCYB.2020.2977374

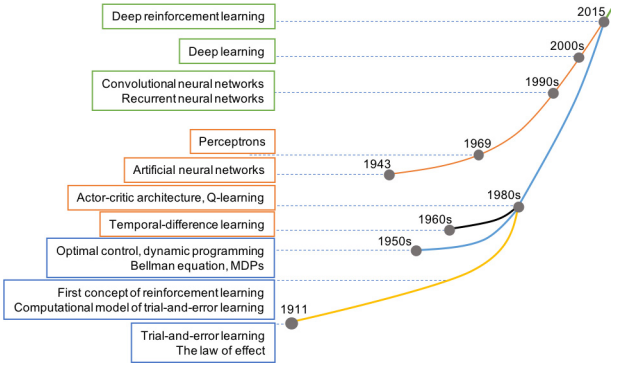

Fig. 1. Emergence of deep RL through different essential milestones.

advances of the deep RL have been extended to solve NP-hard problems such as vehicle routing problem, which is critical in logistics [19], [20].

As real-world problems have become increasingly complex, there are many situations where a single deep RL agent is not able to cope with. In such situations, the applications of a multiagent system (MAS) are indispensable. In an MAS, agents must compete or cooperate to obtain the best overall results. Examples of such systems include multiplayer online games, cooperative robots in the production factories, traffic control systems, and autonomous military systems like unmanned aerial vehicles (UAVs), surveillance, and spacecraft. Among many applications of deep RL in the literature, there are a large number of studies using deep RL in MAS, henceforth multiagent deep RL (MADRL). Extending from a single-agent domain to a multiagent environment creates several challenges. Previous surveys considered different perspectives, for example, Busoniu *et al.* [21] examined the stability and adaptation aspects of agents, Bloembergen *et al.* [22] analyzed the evolutionary dynamics, Hernandez-Leal *et al.* [23] considered emergent behaviors, communication and cooperation learning perspectives, and da Silva *et al.* [24] reviewed methods for knowledge reuse autonomy in multiagent RL (MARL). This article presents an overview of technical challenges in multiagent learning as well as deep RL approaches to these challenges. We cover numerous MADRL perspectives, including nonstationarity, partial observability, multiagent training schemes, transfer learning in MAS, and continuous state and action spaces in multiagent learning. Applications of MADRL in various fields are also reviewed and analyzed in the current study. In the last section, we present extensive discussions and interesting future research directions of MADRL.

II. Background: Reinforcement Learning

A. Preliminary

RL is a TE learning 1) by interacting directly with the environment; 2) to self-teach over time; and 3) eventually achieve designated goal. Specifically, RL defines any decision maker (learner) as an *agent* and anything outside the agent as an *environment*. The interactions between the agent and the environment are described via three essential elements: 1) state s; 2) action a; and 3) reward r [25]. The state of the environment at time step t is denoted as s_t. Thereby, the agent examines s_t and performs a corresponding action a_t. The environment then alters its state s_t to s_{t+1} and provides a feedback reward r_{t+1} to the agent.

The agent's decision is formalized by defining the concept of *policy*. A policy π is a mapping function from any perceived state s to the action a taken from that state. A policy is *deterministic* if the probability of choosing an action a from s: $p(a|s) = 1$ for all state s. In contrast, the policy is *stochastic* if there exists a state s so that $p(a|s) < 1$. In either case, we can define the policy π as a probability distribution of candidate actions that will be selected from a certain state as

$$\begin{aligned}\pi &= \Psi(s) \\ &= \left\{ p(a_i|s) \;\middle|\; \forall a_i \in \Delta_\pi \;\wedge \sum_i p(a_i|s) = 1 \right\} \end{aligned} \tag{1}$$

where Δ_π represents all candidate actions (*action space*) of the policy π. For clarity, we assume that the action space is discrete because the continuous case can be straightforwardly inferred by using the integral notation. Furthermore, we presume that the next state s_{t+1} and feedback reward r_{t+1} are entirely determined by the current state–action pair (s_t, a_t) regardless of the history. Any RL problem satisfies this "memoryless" condition is known as MDP. Therefore, the *dynamics* (model) of an RL problem is completely specified by giving all *transition probabilities* $p(a_i|s)$.

B. Bellman Equation

Remind that the agent receives a feedback reward r_{t+1} for every time step t until it reaches the terminal state s_T. However, the immediate reward r_{t+1} does not represent the long-term profit, we instead leverage a generalized *return value* R_t at time step t

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T \\ &= \sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1} \end{aligned} \tag{2}$$

where γ is a discounted factor so that $0 \le \gamma < 1$. The agent becomes farsighted when γ approaches to 1 and vice versa the agent becomes shortsighted when γ is close to 0.

The next step is to define a *value function* that is used to evaluate how "good" of a certain state s or a certain state–action pair (s, a). Specifically, the value function of the state s under the policy π is calculated by obtaining the expected return value from s: $V_\pi(s) = \mathbb{E}[R_t|s_t = s, \pi]$. Likewise, the value function of state–action pair (s, a) is $Q_\pi(s, a) = \mathbb{E}[R_t|s_t = s, a_t = a, \pi]$. We can leverage value functions to compare how "good" between two policies π and π' using the following rule [25]:

$$\begin{aligned} \pi \le \pi' \iff & [V_\pi(s) \le V_{\pi'}(s) \;\forall s] \\ & \vee \;\; [Q_\pi(s, a) \le Q_{\pi'}(s, a) \;\forall (s, a)]. \end{aligned} \tag{3}$$

Based on (2), we can expand $V_\pi(s)$ and $Q_\pi(s, a)$ to present the relationship between two consecutive states $s = s_t$ and $s' = s_{t+1}$ as [25]

$$V_\pi(s) = \sum_a \pi(s, a) \sum_{s'} p(s'|s, a)\big(\mathbb{W}_{s \to s'|a} + \gamma V_\pi(s')\big) \tag{4}$$

and

$$Q_\pi(s,a) = \sum_{s'} p\big(s'|s,a\big)\Bigg(\mathbb{W}_{s\to s'|a} + \gamma \sum_{a'} \pi\big(s',a'\big) Q_\pi\big(s',a'\big)\Bigg) \quad (5)$$

where $\mathbb{W}_{s\to s'|a} = \mathbb{E}[r_{t+1}|s_t = s, a_t = a, s_{t+1} = s']$. Solving (4) or (5), we can find the value function $V(s)$ or $Q(s,a)$, respectively. Equations (4) and (5) are called the Bellman equations.

Dynamic programming and its variants [26]–[31] can be used to approximate the solutions of the Bellman equations. However, it requires the complete dynamics information of the problem, and thus when the number of states is large, this approach is infeasible due to the lack of memory and computational power of the conventional computer. In the next section, we will review two *model-free* RL methods [require no knowledge of transition probabilities $p(a_i|s)$] to approximate the value functions.

TABLE I
CHARACTERISTICS OF RL METHODS

Category	Pros	Cons
Model-free	• Dynamics of environment is unknown • Deal with larger state-space environments	○ Requires "exploration" condition
On-policy	• Stable when using with function approximator • Suitable with continuous state-space problems	○ Policy must be stochastic
Off-policy	• Simplify algorithm design • Can tackle with different kinds of problems • Policy can be deterministic	○ Unstable when using with function approximator
Bootstrapping	• Learn faster in most cases	○ Not as good as nonbootstrapping methods on mean square error

TABLE II
COMPARISONS BETWEEN DYNAMIC PROGRAMMING AND RL METHODS

Category	Dynamic programming	RL Methods			
		MC	Sarsa	Q-learning	AC
Model-free		✓	✓	✓	✓
On-policy		✓	✓		✓
Off-policy		✓		✓	✓
Bootstrapping	✓		✓	✓	✓

C. RL Methods

In this section, we review two well-known learning schemes in RL: 1) Monte-Carlo (MC) and 2) TD learning. These methods do not require the dynamic information of the environment, that is, they can deal with larger state-space problems than dynamic programming approaches.

1) Monte-Carlo Method: This method estimates the value function by repeatedly generating episodes and recording average return at each state or each state–action pair. The MC method does not require any knowledge of transition probabilities, that is, the MC method is model free. However, this approach has made two essential assumptions to ensure the convergence happens: 1) the number of episodes is large and 2) every state and every action must be visited with a significant number of times.

Generally, MC algorithms are divided into two groups: 1) *on-policy* and 2) *off-policy*. In on-policy methods, we use the policy π for both evaluation and exploration purposes. Therefore, the policy π must be stochastic or *soft*. In contrast, off-policy uses different policy $\pi' \neq \pi$ to generate the episodes and hence π can be deterministic. Although the off-policy is desirable due to its simplicity, the on-policy method is more stable when working with continuous state-space problems and when using together with a function approximator (such as neural networks) [32].

2) Temporal-Difference Method: Similar to MC, the TD method is also learning from experiences (model-free method). However, unlike MC, the TD learning does not wait until the end of episode to make an update. It makes an update on every step within the episode by leveraging the Bellman equation (4) and hence possibly provides faster convergence. Equation (6) presents a one-step TD method

$$V^i(s_t) \longleftarrow \alpha V^{i-1}(s_t) + (1-\alpha)\Big(r_{t+1} + \gamma V^{i-1}(s_{t+1})\Big) \quad (6)$$

where α is a step-size parameter and $0 < \alpha < 1$. The TD learning uses previous estimated values V^{i-1} to update the current ones V^i, which is known as the *bootstrapping* method. Basically, the bootstrapping method learns faster than non-bootstrapping ones in most of the cases [25]. The TD learning is also divided into two categories: 1) on-policy TD control (*Sarsa*) and 2) off-policy TD control (*Q-learning*).

In practice, MC and TD learning often use table memory structure (*tabular method*) to save the value function of each state or each state–action pair. This makes them inefficient due to the lack of memory in solving complex problems where the number of states is large. Therefore, an actor–critic (AC) architecture is designed to subdue this limitation. Specifically, AC includes two separate memory structures for an agent: 1) *actor* and 2) *critic*. The actor structure is used to select a suitable action according to the observed state and transfer to the critic structure for evaluation. The critic structure uses a TD error function to decide the future tendency of the selected action. AC methods can be on-policy or off-policy depending on the implementation details. Table I summarizes the characteristics of RL methods as well as their advantages and disadvantages. Table II highlights the differences between dynamic programming and RL methods, which include MC, Sarsa, *Q*-learning, and AC. As opposed to dynamic programming, RL algorithms are model free [33]. While other RL methods, such as Sarsa, *Q*-learning, and AC use the bootstrapping method, MC requires to restart episodes to update its value function. Notably, algorithms based on AC are the most generic as they can belong to any category.

III. DEEP RL: SINGLE AGENT

A. Deep Q-Network

Deep RL is a broad term that indicates the combination between deep learning and RL to deal with high-dimensional environments [34]–[36]. In 2015, Mnih *et al.* [6] at the first

time announced the success of this combination by creating an autonomous agent that can play competently a series of 49 Atari games. Concisely, the authors proposed a novel structure called DQN that leverages the convolutional neural network (CNN) [37] to directly interpret the graphical representation of the input state s from the environment. The output of DQN produces Q-values of all possible actions $a \in \Delta_\tau$ taken at state s, where Δ_τ denotes the action space [38]. Therefore, DQN can be seen as a policy network τ, parameterized by β, which is continually trained so as to approximate optimal policy. Mathematically, DQN uses the Bellman equation to minimize the loss function $\mathcal{L}(\beta)$ as

$$\mathcal{L}(\beta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q\big(s', a'|\beta'\big) - Q(s, a|\beta)\right)^2\right]. \quad (7)$$

However, using a neural network to approximate value function is proved to be unstable and may result in divergence due to the bias originated from correlative samples [32]. To make the samples uncorrelated, Mnih *et al.* [6] created a *target network* τ', parameterized by β', which is updated in every N steps from the estimation network τ. Moreover, generated samples are stored in an *experience replay memory*. The samples are then retrieved randomly from the experience replay and fed into the training process [39], as described in Fig. 2.

Although DQN basically solved a challenging problem in RL, the curse of dimensionality, this is just a rudimental step in solving completely real-world applications. DQN has numerous drawbacks, which can be remedied by different schemes, from a simple form to complex modifications that we will discuss in the next section.

B. DQN Variants

The first and simplest form of DQN's variant is double DQN (DDQN) proposed in [40] and [41]. The idea of DDQN is to separate the selection of "greedy" action from action evaluation. In this way, DDQN expects to reduce the overestimation of Q-values in the training process. In other words, the *max* operator in (7) is decoupled into two different operators, as represented by the following loss function:

$$\mathcal{L}_{\mathrm{DDQN}}(\beta) = \mathbb{E}\left[\left(r + \gamma Q\left(s', \arg\max_{a'} Q\big(s', a'|\beta\big)|\beta'\right) - Q(s, a|\beta)\right)^2\right]. \quad (8)$$

Empirical experimental results on 57 Atari games show that the normalized performance of DDQN without tuning is two times greater than DQN and three times greater than DQN when tuning.

Second, the experience replay in DQN plays an important role to break the correlations between samples, and at the same time, remind "rare" samples that the policy network may rapidly forget. However, the fact that selecting randomly samples from the experience replay does not completely separate the sample data. Specifically, we prefer rare and goal-related samples to appear more frequent than redundancy ones. Therefore, Schaul *et al.* [42] proposed a *prioritized experience replay* that gives priority to a sample i based on its absolute value of TD error

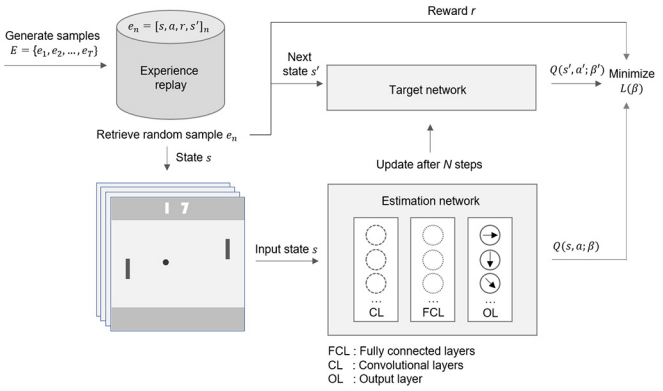

Fig. 2. Deep Q-network structure with experience replay memory and target network whose parameters are updated from the estimation network after every N steps to ensure a stable learning process.

$$p_i = |\delta_i| = |r_i + \gamma \max_a Q\big(s_i, a|\beta'\big) - Q(s_{i-1}, a_{i-1}|\beta)|. \quad (9)$$

The prioritized experience replay when combined with DDQN provides stable convergence of the policy network and achieves a performance up to five times greater than DQN in terms of normalized mean score on 57 Atari games.

DQN's policy evaluation process is struggle to work in "redundant" situations, that is, there are more than two candidate actions that can be selected without getting any negative results. For instance, when driving a car and there are no obstables ahead, we can follow either the left lane or the right lane. If there is an obstacle ahead in the left lane, we must be in the right lane to avoid crashing. Therefore, it is more efficient if we focus only on the road and obstacles ahead. To resolve such situations, Wang *et al.* [43] proposed a novel network architecture called *dueling network*. In the dueling architecture, there are two collateral networks that coexist: one network, parameterized by θ, estimates the state-value function $V(s|\theta)$ and the other one, parameterized by θ', estimates the advantage action function $A(s, a|\theta')$. The two networks are then aggregated using the following equation to approximate the Q-value function:

$$Q\big(s, a|\theta, \theta'\big) = V(s|\theta) + \left(A\big(s, a|\theta'\big) - \frac{1}{|\Delta_\pi|}\sum_{a'} A\big(s, a'|\theta'\big)\right). \quad (10)$$

Because the dueling network represents the action-value function, it was combined with DDQN and prioritized experience replay to boost the performance up to six times more than the standard DQN on the Atari domain [43].

Another drawback of DQN is that it uses a history of four frames as an input to the policy network. DQN is therefore inefficient to solve problems where the current state depends on a significant amount of history information such as "Double Dunk" or "Frostbite" [44]. These games are often called partially observable MDP problems. The straightforward solution is to replace the fully connected layer right after the last convolutional layer of the policy network with a recurrent *long short-term memory*, as described in [44]. This DQN's variant called

deep recurrent Q-network (DRQN) outperforms the standard DQN up to 700 percent in "Double Dunk" and "Frostbite" games. Furthermore, Lample and Chaplot [45] successfully created an agent that beats an average player on "Doom," a 3-D FPS (first-person shooter) environment by adding a game feature layer in DRQN. Another interesting variant of DRQN is deep attention recurrent Q-network (DARQN) [46]. In that article, Sorokin *et al.* added attention mechanism into DRQN so that the network can focus only on important regions in the game, allowing smaller network's parameters and hence speeding the training process. As a result, DARQN achieves a score of 7263 compared with 1284 and 1421 of DQN and DRQN on the game "Seaquest," respectively.

IV. Deep RL: Multiagent

MASs have attracted great attention because they are able to solve complex tasks through the cooperation of individual agents. Within an MAS, the agents communicate with each other and interact with the environment. In a multiagent learning domain, the MDP is generalized to a stochastic game, or a Markov game. Let us denote n as the number of agents, S as a discrete set of environmental states, and $A_i, i = 1, 2, \ldots, n$, as a set of actions for each agent. The joint action set for all agents is defined by $A = A_1 \times A_2 \times \cdots \times A_n$. The state transition probability function is represented by $p : S \times A \times S \to [0, 1]$ and the reward function is specified as $r : S \times A \times S \to \mathbb{R}^n$. The value function of each agent is dependent on the joint action and joint policy, which is characterized by $V^\pi : S \times A \to \mathbb{R}^n$. The following sections describe challenges and MADRL solutions as well as their applications to solve real-world problems.

A. MADRL: Challenges and Solutions

1) Nonstationarity: Controlling multiple agents poses several additional challenges as compared to single-agent setting such as the *heterogeneity* of agents, how to define suitable collective goals or the scalability to a large number of agents that requires the design of compact representations, and more importantly the nonstationarity problem. In a single-agent environment, an agent is concerning only the outcome of its own actions. In a multiagent domain, an agent observes not only the outcomes of its own action but also the behavior of other agents. Learning among the agents is complex because all agents potentially interact with each other and learn concurrently. The interactions among multiple agents constantly reshape the environment and lead to nonstationarity. In this case, learning among the agents sometimes causes changes in the policy of an agent, and can affect the optimal policy of other agents. The estimated potential rewards of an action would be inaccurate and therefore, good policies at a given point in the multiagent setting could not remain so in the future. The convergence theory of Q-learning applied in a single-agent setting is not guaranteed to most multiagent problems as the Markov property does not hold anymore in the nonstationary environment [47]. Therefore, collecting and processing information must be performed with certain recurrence while ensuring that it does not affect the agents' stability. The exploration–exploitation dilemma could be more involved under multiagent settings.

The popular *independent Q-learning* [48] or experience replay-based DQN [6] was not designed for the nonstationary environments. Castaneda [49] proposed two variants of DQN, namely, deep repeated update Q-network (DRUQN) and deep loosely coupled Q-network (DLCQN), to deal with the nonstationarity problem in MAS. The DRUQN is developed based on the repeated update Q-learning (RUQL) model introduced in [50] and [51]. It aims to avoid policy bias by updating the action value inversely proportional to the likelihood of selecting an action. On the other hand, DLCQN relies on the loosely coupled Q-learning proposed in [52], which specifies and adjusts an independence degree for each agent using its negative rewards and observations. Through this independence degree, the agent learns to decide whether it needs to act independently or cooperate with other agents in different circumstances. Likewise, Diallo *et al.* [53] extended DQN to a *multiagent concurrent DQN* and demonstrated that this method can converge in a nonstationary environment. Foerster *et al.* [54] alternatively introduced two methods for stabilizing experience replay of DQN in MADRL. The first method uses the importance sampling approach to naturally decay obsolete data while the second method disambiguates the age of the samples retrieved from the replay memory using a fingerprint.

Recently, to deal with nonstationarity due to concurrent learning of multiple agents in MAS, Palmer *et al.* [55] presented a method, namely, lenient-DQN (LDQN) that applies *leniency* with decaying temperature values to adjust policy updates sampled from the experience replay memory. Leniency in the context of a multiagent setting describes the situation where a learning agent ignores the poor actions of a co-learner, which leads to low rewards, and still cooperates with the co-learner with the hope that the co-learner can improve his actions in the future. For example, agents A and B are learning to play football. Due to a mistake or insufficient training, agent B could not handle the ball passed to him by agent A. In this situation, with the leniency, agent A will think that agent B can improve his skills and thus agent A continues passing the ball to agent B instead of thinking that agent B does not have skills to play football and not passing the ball to agent B again [56]. The LDQN is applied to the coordinated multiagent object transportation problems and its performance is compared with the hysteretic-DQN (HDQN) [57]. The experimental results demonstrate the superiority of LDQN against HDQN in terms of convergence to optimal policies in a stochastic reward environment. The notion of leniency along with a scheduled replay strategy was also incorporated into the weighted DDQN (WDDQN) in [58] to deal with nonstationarity in MAS. The experiments show the better performance of WDDQN against the DDQN in two multiagent environments with stochastic rewards and large state space.

2) Partial Observability: In real-world applications, there are many circumstances where agents only have partial observability of the environment. This matter is more severe in the multiagent problems as they are normally more complex and

large scale. In other words, complete information of states pertaining to the environment is not known to the agents when they interact with the environment. In such situations, the agents observe partial information about the environment and need to make the "best" decision during each time step. This type of problem can be modeled using the partially observable MDP (POMDP).

In the current literature, a number of deep RL models have been proposed to handle POMDP. Hausknecht and Stone [44] proposed DRQN based on a long short-term memory network. With the recurrent structure, the DRQN-based agents are able to learn the improved policy in a robust sense in the partially observable environment. Unlike DQN, DRQN approximates $Q(o, a)$, which is a Q-function with an observation o and an action a, by a recurrent neural network. DRQN treats a hidden state of the network h_{t-1} as an internal state. The DRQN therefore is characterized by the Q-function $(o_t, h_{t-1}, a; \theta_i)$ where θ_i is the parameters of the network at the ith training step. In [59], DRQN is extended to deep distributed recurrent Q-network (DDRQN) to handle multiagent POMDP problems. The success of DDRQN is relied on three notable features, that is, last-action inputs, interagent weight sharing, and disabling experience replay. The first feature, that is, last-action inputs, requires the provision of the previous action of each agent as input to its next step. The interagent weight sharing means that all agents use weights of only one network, which is learned during the training process. The disabling experience replay simply excludes the experience replay feature of DQN. The DDRQN therefore learns a Q-function of the form $Q(o_t^m, h_{t-1}^m, m, a_{t-1}^m, a_t^m; \theta_i)$, where each agent receives its own index m as the input. Weight sharing decreases learning time because it reduces the number of parameters to be learned. Although each agent has a different observation and hidden state, this approach, however, assumes that agents have the same set of actions. To address complex problems, autonomous agents often have different sets of actions. For example, UAVs maneuvre in the air while robots operate on the ground. Therefore, action spaces of UAVs and robots are different and thus the interagent weight sharing feature cannot be applied.

Scaling to a system of many agents in partially observable domains is a challenging problem. Gupta *et al.* [60] extended the *curriculum learning* technique to an MAS, which integrates with three classes of deep RL, including policy gradient, TD error, and AC methods. The curriculum principle is to start learning to complete simple tasks first to accumulate knowledge before proceeding to perform complex tasks. This is suitable with an MAS environment where fewer agents initially collaborate before extending to accommodate more agents to complete increasingly difficult tasks. The experimental results show the vitality of the curriculum learning method in scaling deep RL algorithms to complex multiagent problems.

Hong *et al.* [61] introduced a deep policy inference Q-network (DPIQN) to model MASs and its enhanced version deep recurrent policy inference Q-network (DRPIQN) to cope with partial observability. Both DPIQN and DRPIQN are learned by adapting the network's attention to policy features and their own Q-values at various stages of the training process. The experiments show the better overall performance of both DPIQN and DRPIQN over the baseline DQN and DRQN [44]. Also in the context of partial observability, but extended to multitask, multiagent problems, Omidshafiei *et al.* [57] proposed a method called multitask MARL (MT-MARL) that integrates hysteretic learners [62], DRQNs [44], distillation [63], and concurrent experience replay trajectories (CERTs), which are a decentralized extension of experience replay strategy proposed in [6]. The agents are not explicitly provided with task identity (thus partial observability) while they cooperatively learn to complete a set of *decentralized POMDP* tasks with sparse rewards. This method, however, has a disadvantage that cannot perform in an environment with heterogeneous agents.

Apart from partial observability, there are circumstances that agents must deal with extremely noisy observations, which are weakly correlated with the true state of the environment. Kilinc and Montana [64] introduced a method denoted as MADDPG-M that combines deep deterministic policy gradient (DDPG) and a communication medium to address these circumstances. Agents need to decide whether their observations are informative to share with other agents and the communication policies are learned concurrently with the main policies through experience. Recently, Foerster *et al.* [65] proposed a Bayesian action decoder (BAD) algorithm for learning multiple agents with cooperative partial observable settings. A new concept, namely, public belief MDP, is introduced based on BAD that employs an approximate Bayesian update to attain a public belief with publicly observable features in the environment. BAD relies on a factorized and approximate belief state to discover conventions to enable agents to learn optimal policies efficiently. This is closely relevant to *theory of mind* that humans normally use to interpret others' actions. The experimental results on a proof-of-principle two-step matrix game and the cooperative partial-information card game Hanabi demonstrate the efficiency and superiority of the proposed method against traditional policy gradient algorithms.

3) MAS Training Schemes: The direct extension of single agent deep RL to multiagent environment is to learn each agent independently by considering other agents as part of the environment as the independent Q-learning algorithm proposed in [66]. This method is vulnerable to overfitting [67] and computationally expensive, and therefore the number of agents involved is limited. An alternative and popular approach is the *centralized learning and decentralized execution* where a group of agents can be trained simultaneously by applying a centralized method via an open communication channel [68]. Decentralized policies where each agent can take actions based on its local observations have an advantage under partial observability and in limited communications during execution. Centralized learning of decentralized policies has become a standard paradigm in multiagent settings because the learning process may happen in a simulator and a laboratory where there are no communication constraints, and extra state information is available [68].

Three different training schemes for an MAS consisting of centralized learning, concurrent learning, and parameter

Algorithm 1 PS-TRPO

1: Initialize parameters of policy network Θ_0, and trust region size Δ
2: **for** $i \longleftarrow 0, 1, \ldots$ **do**
3: Generate trajectories for all agents as $\tau \sim \pi_{\theta_i}$ using the policy with shared parameters.
4: For each agent m, compute the advantage values $A_{\pi_{\theta_i}}(o^m, m, a^m)$ with m is the agent index.
5: Search $\pi_{\theta_{i+1}}$ that maximizes $L(\theta) = E_{o\sim p_{\theta_k}, a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a|o,m)}{\pi_{\theta_k}(a|o,m)} A_{\theta_k}(o, m, a)\right]$ subject to $\bar{D}_{KL}(\pi_{\theta_i} \parallel \pi_{\theta_{i+1}}) \leq \Delta$ where D_{KL} is the KL divergence between distributions of two policies, and p_θ are the discounted frequencies of state visitation caused by π_θ.
6: **end for**

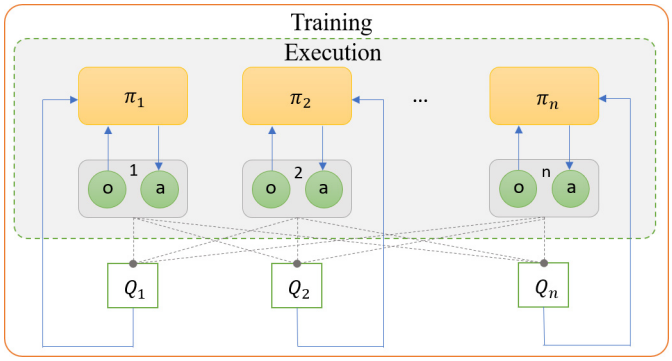

Fig. 3. Centralized learning and decentralized execution-based MADDPG where policies of agents are learned by the centralized critic with augmented information from other agents' observations and actions.

sharing are examined in [60]. Centralized policy attempts to obtain a joint action from joint observations of all agents while the concurrent learning trains agents simultaneously using the joint reward signal. In the latter, each agent learns its own policy independently based on the private observation. Alternatively, the parameter sharing scheme allows agents to be trained simultaneously using the experiences of all agents although each agent can obtain unique observations. With the ability to execute decentralized policies, parameter sharing can be used to extend a single-agent deep RL algorithm to accommodate a system of many agents. Particularly, the combination of *parameter sharing and TRPO*, namely, PS-TRPO, has been proposed in [60], and briefly summarized in Algorithm 1. The PS-TRPO has demonstrated great performance when dealing with high-dimensional observations and continuous action spaces under partial observability.

Foerster *et al.* [69] introduced reinforced interagent learning (RIAL) and differentiable interagent learning (DIAL) methods based on the centralized learning approach to improve agent learning communication. In RIAL, deep Q-learning has a recurrent structure to address the partial observability issue, in which independent Q-learning offers individual agents to learn their own network parameters. DIAL pushes gradients from one agent to another through a channel, allowing end-to-end backpropagation across agents. Likewise, Sukhbaatar *et al.* [70] developed a communication neural net (CommNet) allowing dynamic agents to learn continuous communication alongside their policy for fully cooperative tasks. Unlike CommNet, He *et al.* [71] proposed a method, namely, deep reinforcement opponent network (DRON) that encodes observation of the opponent agent into DQN to jointly learn a policy and behaviors of opponents without domain knowledge.

Both decentralized and centralized perspectives are incorporated into the hierarchical master–slave architecture to form a model called master–slave MARL (MS-MARL) in [72] and [73] to solve the communication problem in MAS. The master agent receives and collectively processes messages from slave agents and then generates unique instructive messages to each slave agent. The slave agents use their own information and instructive messages from the master agent to take actions. This model significantly reduces the communication burden within a MAS compared to the peer-peer architecture, especially when the system has many agents.

The multiagent deep deterministic policy gradient (MADDPG) method based on the AC policy gradient algorithms is proposed in [74]. MADDPG features the centralized learning and decentralized execution paradigm in which the critic uses extra information to ease the training process while actors take actions based on their own local observations. Fig. 3 illustrates the multiagent decentralized actor and centralized critic components of MADDPG where only the actors are used during the execution phase.

Recently, another multiagent AC method, namely, counterfactual multiagent (COMA), which is also relied on the centralized learning and decentralized execution scheme, is introduced in [75]. Unlike MADDPG [74], COMA can handle the *multiagent credit assignment* problem [76] where agents are difficult to work out their contribution to the team's success from global rewards generated by joint actions in cooperative settings. COMA, however, has a disadvantage that focuses only on discrete action space while MADDPG is able to learn continuous policies effectively.

4) Continuous Action Spaces: Most deep RL models can only be applied to discrete spaces [77]. For example, DQN [6] is limited only to problems with discrete and low-dimensional action spaces, although it can handle high-dimensional observation spaces. The DQN aims to find action that has maximum action-value, and therefore requires an iterative optimization process at every step in the continuous action (state) spaces. Discretizing the action space is a possible solution to adapt deep RL methods to continuous domains. However, this creates many problems, notably is the curse of dimensionality: the exponential increase of action numbers against the number of degrees of freedom.

Schulman *et al.* [78] proposed a trust region policy optimization (TRPO) method, which can be extended to continuous states and actions, for optimizing stochastic control policies in the domain of robotic locomotion and image-based game playing. Lillicrap *et al.* [77] introduced an off-policy algorithm, namely, DDPG, which utilizes the AC architecture [79], [80] to handle the continuous action spaces. Based

TABLE III
MULTIAGENT LEARNING CHALLENGES AND THEIR SOLVING METHODS

Challenges	Value-based	Actor-critic	Policy-based
Partial observability	DRQN [44]; DDRQN [59]; RIAL and DIAL [69]; Action-specific DRQN [89]; MT-MARL [57]; PS-DQN [60]; RL as a Rehearsal (RLaR) [68]	PS-DDPG and PS-A3C [60]; MADDPG-M [64]	DPIQN and DRPIQN [61]; PS-TRPO [60]; Bayesian action decoder (BAD) [65]
Non-stationarity	DRUQN and DLCQN [49]; Multi-agent concurrent DQN [53]; Recurrent DQN-based multi-agent importance sampling and fingerprints [54]; Hysteretic-DQN [57]; Lenient-DQN [55]; WDDQN [58]	MADDPG [74]; PS-A3C [60]	PS-TRPO [60]
Continuous action spaces		Recurrent DPG [82]; DDPG [77]	TRPO [78]; PS-TRPO [60]
Multi-agent training schemes	Multi-agent extension of DQN [66]; RIAL and DIAL [69]; CommNet [70]; DRON [71]; MS-MARL [72], [73]; Linearly fuzzified joint Q-function for MAS [90]	MADDPG [74]; COMA [75]	
Transfer learning in MAS	Policy distillation [63]; Multi-task policy distillation [85]; Multi-agent DQN [87]	Progressive networks [83]	Actor-Mimic [86]

on the deterministic policy gradient (DPG) [81], DDPG deterministically maps states to specific actions using a parameterized actor function while keeping DQN learning on the critic side. This approach, however, requires a large number of training episodes to find solutions, as found common in model-free reinforcement methods. Heess *et al.* [82] extended DDPG to recurrent DPG (RDPG) to handle problems with continuous action spaces under partial observability, where the true state is not available to the agents when making decisions. Recently, Gupta *et al.* [60] introduced the PS-TRPO method for multiagent learning (see Algorithm 1). This method is based on the foundation of TRPO so that it can deal with continuous action spaces effectively.

5) Transfer Learning for MADRL: Training the Q-network or generally, a deep RL model of a single agent is often very computationally expensive. This problem is significantly severe for a system of multiple agents. To improve the performance and reduce computational costs during the training process of multiple deep RL models, several studies have promoted transfer learning for deep RL. Rusu *et al.* [63], [83] proposed a *policy distillation* method and *progressive neural networks* to promote transfer learning in the context of the deep RL. These methods, however, are computationally complex and expensive [84]. Yin and Pan [85] likewise introduced another policy distillation architecture to apply knowledge transfer for the deep RL. That method reduces training time and outperforms DQNs but its exploration strategy is still not efficient. Parisotto *et al.* [86] proposed the *actor-mimic* method for multitask and transfer learning that improves learning speed of a deep policy network. The network can obtain an expert performance on many games simultaneously, although its model is not so complex. That method, however, requires a sufficient level of similarity between the source and target tasks and is vulnerable to negative transfer.

A multiagent environment is reformulated into an image-like representation and CNNs are utilized in [87] to estimate Q-values for each agent in question. That approach can address the scalability problem in MAS when the transfer learning method can be used to speed up the training process. A policy network trained on a different but related environment is used for the learning process of other agents to reduce computational expenses. The experiments carried out on the pursuit-evasion problem [88] show the effectiveness of the transfer learning approach in the multiagent domain.

Table III presents a summary of reviewed articles that address different multiagent learning challenges. It can be seen that many extensions of DQN have been proposed in the literature while policy-based or AC methods have not adequately been explored in multiagent environments.

B. MADRL Applications

Since the success of deep RL marked by the DQN proposed in [6], many algorithms have been proposed to integrate deep learning to multiagent learning. These algorithms can solve complex problems in various fields. This section provides a survey of these applications with a focus on the integration of deep learning and MARL. Table IV summarizes the features and limitations of approaches to these applications.

An MADRL model is introduced in [91] to deal with energy sharing problem in a zero-energy community that comprises a collection of zero-energy buildings, which have the total energy use over a year smaller than or equal to the renewables generation within each building. A deep RL agent is used to characterize each building to learn appropriate actions in sharing energy with other buildings. The global reward is modelled by the negative of the community energy status as reward $= -(\sum_{i=1}^{n} c(h_i) - g(h_i))$, where $c(h_i)$ and $g(h_i)$ are the energy consumed and energy generated by the ith building. The community monitoring service is introduced to manage the group membership activities such as joining and leaving the group or maintaining a list of active agents. The experiments show the superiority of the proposed model compared to the random action selection strategy in terms of net zero-energy balance as a community.

A combination of hierarchical RL and MADRL methods was developed to coordinate and control multiple agents in problems where agents' privacy is prioritized [92]. Such distributed scheduling problems could be a multitask dialogue where an automated assistant needs to help a user to plan

TABLE IV
SUMMARY OF TYPICAL MADRL APPLICATIONS IN DIFFERENT FIELDS

Applications	Basic DLR	Features	Limitations
Energy sharing optimization [91]	DQN	•Each building is characterized by a DRL agent to learn appropriate actions independently. •Agents' actions include: consume and store excess energy, request neighbor or supply grid for additional energy, grant or deny requests. •Agents collaborate via shared or global rewards to achieve a common goal, *i.e.* zero-energy status.	•Agents' behaviors cannot be observed in an online fashion. •Limited number of houses, currently ten houses at maximum were experimented. •Energy price is not considered.
Federated control [92]	Hierarchical-DQN (h-DQN) [133]	•Divide the control problem into disjoint subtasks and leverage *temporal abstractions*. •Use *meta-controller* to guide decentralized controllers. •Able to solve distributed scheduling problems such as multi-task dialogue, urban traffic control.	•Does not address non-stationarity problem. •Number of agents is currently limited at six. •Meta-controller's optimal policy becomes complicated and inefficient when the number of agents increases.
Sequential social dilemma (SSD) [93]	DQN	•Introduce an SSD model to extend MGSD to capture sequential structure of real-world social dilemmas. •Describe SSDs as general-sum Markov games with partial observations. •Multi-agent DQN is used to find equilibria of SSD problems.	•Assume agent's learning is independent and regard the others as part of the environment. •Agents do not recursively reason about one another's learning.
Common-pool resource (CPR) appropriation [98]	DQN	•Introduce a new CPR appropriation model using an MAS containing spatially and temporally environment dynamics. •Use the *descriptive agenda* [134] to describe the behaviors emerging when agents learn in the presence of other learning agents. •Simulate multiple independent agents with each learned by DQN.	•Single agent DQN is extended to multi-agent environment where the Markov assumption is no longer hold. •Agents do not do anything of *rational negotiation*, e.g. bargaining, building consensus, or making appeals.
Swarm systems [100]	DQN and DDPG	•Agents can only observe local environment (partial observability) but not the global state. •Use guided approach for multi-agent learning where actors make decisions based on locally sensed information whilst critic has central access to global state.	•Can only work with homogeneous agents. •Unable to converge to meaningful policies in huge dimensionality and partial observed problem.
Traffic lights control [102]	DDDQN and IDQN	•Learning multiple agents is performed using IDQN where each agent is modelled by DDDQN. •First approach to address heterogeneous multi-agent learning in urban traffic control. •Fingerprint technique is used to stabilize the experience replay memory to handle non-stationarity.	•The proposed deep RL approach learns ineffectively in high traffic conditions. •The fingerprint does not improve the performance of experience replay although the latter is required for efficient learning.
Keepaway soccer [103]	DQN	•Low-dimensional state space, described by only 13 variables. •Heterogeneous MAS, each agent has different experience replay memory and different network policy.	•Number of agents is limited, currently setting with 3 keepers vs. 2 takers. •Heterogeneous learning speed is significantly lower than homogeneous case.
Task and resources allocation [105]	CommNet [70]	•Propose distributed task allocation where agents can request help from cooperating neighbors. •Three types of agents are defined: manager, participant and mediator. •Communication protocols are learned simultaneously with agents' policies through CommNet.	•May not be able to deal with heterogeneous agents. •Computational deficiencies regarding the decentralization and reallocation characteristics. •Experiments only on small state action spaces.
Large-scale fleet management [106]	Actor-critic and DQN	•Reallocate vehicles ahead of time to balance the transport demands and supplies. •Geographic context and collaborative context are integrated to coordinate agents. •Two proposed algorithms, *contextual multi-agent actor-critic* and *contextual deep Q-learning*, can achieve explicit coordination among thousands of agents.	•Can only deal with discrete actions and each agent has a small (simplified) action space. •Assume that agents in the same region at the same time interval (*i.e.* same spatial-temporal state) are homogeneous.
Action markets [107]	DQN	•Agents can exchange their atomic actions for environmental rewards. •Reduce greedy behavior and thus negative effects of individual reward maximization. •The proposed approach significantly increases the overall reward compared to methods without action trading.	•Agents cannot find prices for actions by themselves because they are given at design time. •Strongly assume that agents cannot make offers that they do not eventually hold.

for several independent tasks, for example, purchase a train ticket to the city, book a movie ticket, and make a dinner reservation in a restaurant. Each of these tasks is handled by a decentralized controller while the assistant is a meta-controller which benefits from *temporal abstractions* to lessen the communication complexity, and thus able to find a globally consistent solution for the user (Fig. 4). On the other hand, Leibo *et al.* [93] introduced a sequential social dilemma (SSD) model based on the general-sum Markov games under partial observability to address the evolution of cooperation in MAS. Being able to capture the sequential structure of real-world social dilemmas, SSD is an extension of matrix game social dilemma (MGSD) that has been applied to various phenomena in social science and biology [22], [52], [94]. The

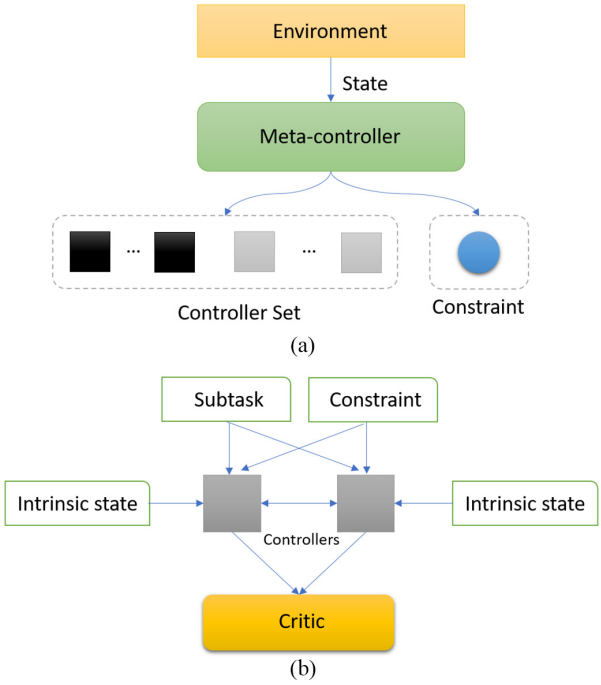

Fig. 4. Federated control with a hierarchical MADRL method. (a) Meta-controller receives a state from the environment and assigns a subtask and its associated constraint to controllers. (b) Controllers have separate partial views of the environment but need to communicate to complete the subtask.

general-sum modeling requires solving algorithms to either track different potential equilibria for each agent or be able to find a cyclic strategy consisting of multiple policies learned by using different state space sweeps [95], [96]. The DQN is utilized to characterize self-interested independent learning agents to find equilibria of the SSD, which cannot be solved by the standard evolution and learning methods used for MGSD [97]. Pérolat *et al.* [98] also demonstrated the application of MADRL in social science phenomenon, that is, the common-pool resource (CPR) appropriation. The proposed method comprises a spatially and temporally dynamic CPR environment [99] and an MAS of independent self-interested DQNs. The CPR appropriation problem is solved by self-organization that adjusts the incentives felt by independent individual agents over time.

The swarm systems are formulated in [100] as a special case of the decentralized POMDP [101] and used an AC deep RL approach to control a group of cooperative agents. The Q-function is learned using a global state information, which can be the view of a camera capturing the scene in swarm robotics examples. The group can perform complex tasks such as search and rescue or distributed assembly although individual agent has limited sensory capability. That model has a drawback as it assumes agents to be homogenous. The use of IDQN was proposed in [102] to address the heterogeneity problem in the multiagent environment of urban traffic light control. Each agent is learned by the dueling DDQN (DDDQN), which integrates dueling networks, DDQN, and prioritized experience replay. The heterogeneous agents are trained independently and simultaneously, considering other agents as part of the environment. The nonstationarity of the multiagent environment is dealt with by a technique of

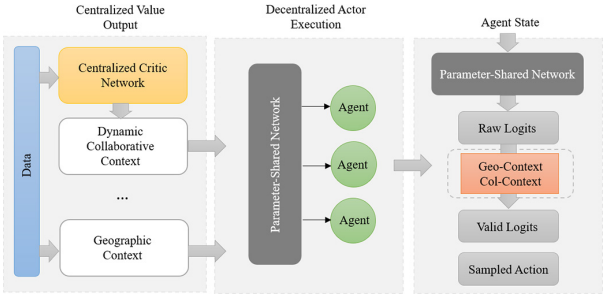

Fig. 5. Contextual multiagent AC architecture proposed in [106] where decentralized execution is coordinated using the centralized value network's outputs as illustrated in the left part while the right part shows how context is embedded in the policy network.

fingerprinting that disambiguates the age of training samples and stabilizes the replay memory.

A special application of DQN to the heterogeneous MAS where the state space is low dimensional was presented in [103]. The experiments are performed on a multiagent Keepaway soccer problem whose state comprises only 13 variables. To handle heterogeneity, each DQN agent is set up with different experience replay memory and neural network. Agents cannot communicate with each other but only can observe others' behaviors. While DQNs can enhance results in terms of game score in the heterogeneous team learning settings in low-dimensional environments, their learning process is significantly slower than that in the homogeneous cases.

Establishing communication channels among agents during learning is an important step in designing and constructing MADRL algorithms. Nguyen *et al.* [104] characterized the communication channel via human knowledge represented by images and allow deep RL agents to communicate using these shared images. The *asynchronous advantage AC* (A3C) algorithm [80] is used to learn optimal policy for each agent, which can be extended to multiple heterogeneous agents. On the other hand, Noureddine *et al.* [105] introduced a method, namely, the task allocation process using cooperative deep RL, to allow multiple agents to interact with each other and allocate resources and tasks effectively. Agents can request help from their cooperative neighbors in a loosely coupled distributed multiagent environment. The CommNet model [70] is used to facilitate communications among agents, which are characterized by DRQN [44].

The large-scale fleet management problem is addressed in [106] using MADRL through two algorithms, namely, *contextual deep Q-learning* and *contextual multiagent AC*. These algorithms aim to balance the difference between demand and supply by reallocating transportation resources that help to reduce traffic congestion and increase transportation efficiency. The contextual multiagent AC model is illustrated in Fig. 5 where a parameter-shared policy network is used to coordinate the agents, which represent available vehicles or equivalently the idle drivers.

Recently, an interesting approach to an MAS where agents can trade their actions in exchange for other resources, e.g., environmental rewards, is introduced in [107]. The action trading was inspired by the fundamental theorem of welfare economics that competitive markets adjust toward the

Pareto efficiency. Specifically, agents need to extend their action spaces and learn two policies simultaneously: one for the original stochastic reward and another for trading environmental reward. The behavior market realized from the action trading helps mitigate greedy behavior (like the tit-for-tat game-theoretic strategy proposed in [108]), enable agents to incentivize other agents and reduce the negative effects of individual reward maximization.

V. Conclusion and Research Directions

This article presents an overview of different challenges in multiagent learning and solutions to these challenges using deep RL methods. We group the surveyed articles into five categories, including nonstationarity, partial observability, multiagent training schemes, multiagent transfer learning, and continuous state and action spaces. We have highlighted the advantages and disadvantages of the approaches to address the challenges. Applications of MADRL methods in different fields are also reviewed thoroughly. We have found that the integration of deep learning into traditional MARL methods has been able to solve many complex problems, such as urban traffic light control, energy sharing problem in a zero-energy community, large-scale fleet management, task and resources allocation, swarm robotics, and social science phenomena. The results indicate that deep RL-based methods provide a viable approach to handling complex tasks in the MAS domain.

Learning from demonstration, including *imitation learning* and *inverse RL* has been effective in single-agent deep RL [109]. On the one hand, imitation learning tries to map between states to actions as a supervised approach. It directly generalizes the expert strategy to unvisited states so that it is close to a multiclass classification problem in cases of finite action set. On the other hand, the inverse RL agent needs to infer a reward function from the expert demonstrations. Inverse RL assumes that the expert policy is optimal regarding the unknown reward function [110], [111]. These methods, however, have not yet been explored fully in multiagent environments. Both imitation learning and inverse RL have great potential for applications in MAS. It is expected that they can reduce the learning time and improve the effectiveness of MAS. A very straightforward challenge arose from these applications is the requirement of multiple experts who are able to demonstrate the tasks collaboratively. Furthermore, the communication and reasoning capabilities of experts are difficult to be characterized and modeled by autonomous agents in the MAS domain. These pose important research questions toward extensions of the imitation learning and the inverse RL to MADRL methods. In addition, for complex tasks or behaviors which are difficult for humans to demonstrate, there is a need for alternative methods that allow human preferences to be integrated into deep RL [104], [112], [113].

The deep RL has considerably facilitated autonomy, which allows to deploy many applications in robotics or autonomous vehicles. The most common drawback of deep RL models, however, is the ability to interact with the human through human–machine teaming technologies. In complex and adversarial environments, there is a critical need for human intellect teamed with technology because humans alone cannot sustain the volume, and machines alone cannot issue creative responses when new situations are introduced. Recent advances of *human-on-the-loop* architecture [114] can be fused with MADRL to integrate humans and autonomous agents to deal with complex problems. In human-on-the-loop, agents execute their tasks autonomously until completion, with a human in a monitoring or supervisory role reserving the ability to intervene in operations carried out by agents. A human-on-the-loop-based architecture can be fully autonomous if human supervisors allow task completion by agents entirely on their own [114].

The model-free deep RL has been able to solve many complex problems both in single-agent and multiagent domains. This category of methods however requires a huge number of samples and long learning time to achieve a good performance. The *model-based* methods have been effective in terms of sample efficiency, transferability, and generality in various problems using single-agent as well as multiagent models. Although the deep learning extensions of the model-based methods have been studied recently in single-agent domain, e.g., [115]–[120], these extensions have not been investigated widely in the multiagent domain. This creates a research gap that could be developed to a research direction in model-based MADRL. In addition, dealing with high-dimensional observations using model-based approaches or combining elements of model-based planning and model-free policy is another active, exciting but underexplored research area.

Scaling to large systems, especially dealing with many heterogeneous agents, has been a primary challenge in the RL research domain since its first days. As the world dynamics become more and more complex, this challenge has always been required to resolve. Since agents have common behaviors, such as actions, domain knowledge, and goals (homogeneous agents), the scalability can be achievable by (partially) centralized training and decentralized execution [121], [122]. In the heterogeneous setting with many agents, the key challenge is how to provide the most optimal solution and maximize the task completion success based on self-learning with effective coordinative and cooperative strategies among the agents. A research direction to address this difficulty is well worth an investigation.

Regarding applications of multiagent learning, there have been many studies using traditional MARL methods to solve various problems such as controlling a group of autonomous vehicles or drones [123], robot soccer [124], controlling traffic signals [125], coordinating collaborative bots in factories and warehouse [126], controlling electrical power networks [127] or optimizing distributed sensor networks [128], automated trading [129], machine bidding in competitive e-commerce and financial markets [130], resource management [131], and transportation [132]. Since the emergence of DQN [6], efforts to extend traditional RL to deep RL in the multiagent domain have been found in the literature but they are still very limited (see Table IV for applications available in the current literature). Many applications of MARL can now be solved effectively by MADRL based on its high-dimension handling capability. Therefore, there is a need for further empirical research to apply MADRL methods to effectively

solve complex real-world problems such as the aforementioned applications.

REFERENCES

[1] E. L. Thorndike, "Animal intelligence: An experimental study of the associate processes in animals," *Amer. Psychol.*, vol. 53, no. 10, pp. 1125–1127, 1898.

[2] M. L. Minsky, *Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain Model Problem*, Princeton Univ., Princeton, NJ, USA, 1954.

[3] A. Klopf, *Brain Function and Adaptive Systems: A Heterostatic Theory*, Air Force Cambridge Res., Cambridge, U.K., 1972.

[4] C. J. Watkins and P. Dayan, "*Q*-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[5] R. Bellman, "On the theory of dynamic programming," *Proc. Nat. Acad. Sci. USA*, vol. 38, no. 8, pp. 716–719, 1952.

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

[7] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," *Artif. Intell.*, vol. 55, nos. 2–3, pp. 311–365, 1992.

[8] S. Schaal, "Learning from demonstration," in *Proc. Adv. Neural Inf. Process. Syst.* 1997, pp. 1040–1046.

[9] H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robot. Auton. Syst.*, vol. 22, nos. 3–4, pp. 283–302, 1997.

[10] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Auton. Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[11] K. Mulling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Robot. Res.*, vol. 32, no. 3, pp. 263–279, 2013.

[12] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, May 2013.

[13] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[14] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1329–1338.

[15] C. Beattie *et al.*, "DeepMind lab," 2016. [Online]. Available: arXiv:1612.03801.

[16] B. Luo, D. Liu, and H.-N. Wu, "Adaptive constrained optimal control design for data-based nonlinear discrete-time systems with critic-only structure," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2099–2111, Jun. 2017.

[17] Z. Wan, C. Jiang, M. Fahad, Z. Ni, Y. Guo, and H. He, "Robot-assisted pedestrian regulation based on deep reinforcement learning," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1669–1682, Apr. 2020, doi: 10.1109/TCYB.2018.2878977.

[18] T. Tan, F. Bao, Y. Deng, A. Jin, Q. Dai, and J. Wang, "Cooperative deep reinforcement learning for large-scale traffic grid signal control," *IEEE Trans. Cybern.*, early access, doi: 10.1109/TCYB.2019.2904742.

[19] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9839–9849.

[20] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" 2018. [Online]. Available: arXiv:1803.08475.

[21] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[22] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers, "Evolutionary dynamics of multi-agent learning: A survey," *J. Artif. Intell. Res.*, vol. 53, pp. 659–697, Aug. 2015.

[23] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Is multiagent deep reinforcement learning the answer or the question? A brief survey," 2018. [Online]. Available: arXiv:1810.05587.

[24] F. L. da Silva, M. E. Taylor, and A. H. R. Costa, "Autonomously reusing knowledge in multiagent reinforcement learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 5487–5493.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[26] B. Luo, D. Liu, H.-N. Wu, D. Wang, and F. L. Lewis, "Policy gradient adaptive dynamic programming for data-based optimal control," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3341–3354, Oct. 2017.

[27] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, 2012.

[28] H. Zhang, H. Jiang, C. Luo, and G. Xiao, "Discrete-time nonzero-sum games for multiplayer using policy-iteration-based adaptive dynamic programming algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3331–3340, Oct. 2017.

[29] W. Gao, Z. P. Jiang, F. L. Lewis, and Y. Wang, "Cooperative optimal output regulation of multi-agent systems using adaptive dynamic programming," in *Proc. Amer. Control Conf. (ACC),* May 2017, pp. 2674–2679.

[30] J. Zhang, H. Zhang, and T. Feng, "Distributed optimal consensus control for nonlinear multiagent system with unknown dynamic," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3339–3348, Aug. 2018.

[31] H. Zhang, H. Su, K. Zhang, and Y. Luo, "Event-triggered adaptive dynamic programming algorithm for non-zero-sum games of unknown nonlinear systems via generalized fuzzy hyperbolic models," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 11, pp. 2202–2214, Nov. 2019, doi: 10.1109/TFUZZ.2019.2896544.

[32] J. N. Tsitsiklis and B. Van Roy, "Analysis of temporal-difference learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.* 1997, pp. 1075–1081.

[33] B. Luo, D. Liu, T. Huang, and D. Wang, "Model-free optimal tracking control via critic-only *Q*-learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 10, pp. 2134–2144, Oct. 2016.

[34] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[35] Y. Li, "Deep reinforcement learning: An overview," 2017. [Online]. Available: arXiv:1701.07274.

[36] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "System design perspective for human-level agents using deep reinforcement learning: A survey," *IEEE Access*, vol. 5, pp. 27091–27102, 2017.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* 2012, pp. 1097–1105.

[38] T. Nguyen, "A multi-objective deep reinforcement learning framework," 2018. [Online]. Available: arXiv:1803.02965.

[39] B. Luo, Y. Yang, and D. Liu, "Adaptive *Q*-learning for data-based optimal output regulation with experience replay," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3337–3348, Dec. 2018.

[40] H. V. Hasselt, "Double *Q*-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2613–2621.

[41] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double *Q*-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, Feb. 2016, pp. 2094–2100.

[42] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015. [Online]. Available: arXiv:1511.05952.

[43] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. D. Freitas, " Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1995–2003.

[44] M. J. Hausknecht and P. Stone, "Deep recurrent *Q*-learning for partially observable MDPs," in *Proc. AAAI Fall Symp. Series*, Sep. 2015, pp. 29–37.

[45] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 2017, pp. 2140–2146.

[46] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva, "Deep attention recurrent *Q*-network," 2015. [Online]. Available: arXiv:1512.01693.

[47] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," 2017. [Online]. Available: arXiv:1707.09183.

[48] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 330–337).

[49] A. O. Castaneda, "Deep reinforcement learning variants of multi-agent learning algorithms," M.S. thesis, School Informat., Univ. Edinburgh, Edinburgh, U.K., 2016.

[50] S. Abdallah and M. Kaisers, "Addressing the policy-bias of *Q*-learning by repeating updates," in *Proc. 12th Int. Conf. Auton. Agents Multiagent Syst.*, May 2013, pp. 1045–1052.

[51] S. Abdallah and M. Kaisers, "Addressing environment non-stationarity by repeating *Q*-learning updates," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1582–1612, 2016.

[52] C. Yu, M. Zhang, F. Ren, and G. Tan, "Emotional multiagent reinforcement learning in spatial social dilemmas," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3083–3096, Dec. 2015.

[53] E. A. O. Diallo, A. Sugiyama, and T. Sugawara, "Learning to coordinate with deep reinforcement learning in doubles Pong game," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 14–19.

[54] J. Foerster *et al.*, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2017, pp. 1146–1155.

[55] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, Jul. 2018, pp. 443–451.

[56] L. Panait, K. Tuyls, and S. Luke, "Theoretical advantages of lenient learners: An evolutionary game theoretic perspective," *J. Mach. Learn. Res.*, vol. 9, pp. 423–457, Mar. 2008.

[57] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2017, pp. 2681–2690.

[58] Y. Zheng, Z. Meng, J. Hao, and Z. Zhang, "Weighted double deep multiagent reinforcement learning in stochastic cooperative environments," in *Proc. Pac. Rim Int. Conf. Artif. Intell.*, Aug. 2018, pp. 421–429.

[59] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent *Q*-networks," 2016. [Online]. Available: arXiv:1602.02672.

[60] J. K. Gupta, M. Egorov, and M. J. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, May 2017, pp. 66–83.

[61] Z.-W. Hong, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, and C.-Y. Lee, "A deep policy inference *Q*-network for multi-agent systems," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, Jul. 2018, pp. 1388–1396.

[62] L. Matignon, G. Laurent, and N. L. Fort-Piat, "Hysteretic *Q*-learning: An algorithm for decentralized reinforcement learning in cooperative multi-agent teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2007, pp. 64–69.

[63] A. A. Rusu *et al.*, "Policy distillation," 2015. [Online]. Available: arXiv:1511.06295.

[64] O. Kilinc and G. Montana, "Multi-agent deep reinforcement learning with extremely noisy observations," 2018. [Online]. Available: arXiv:1812.00922.

[65] J. N. Foerster *et al.*, "Bayesian action decoder for deep multi-agent reinforcement learning," 2018. [Online]. Available: arXiv:1811.01458.

[66] A. Tampuu *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, 2017, Art. no. e0172395.

[67] M. Lanctot *et al.*, "A unified game-theoretic approach to multiagent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4193–4206.

[68] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, May 2016.

[69] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.

[70] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2244–2252.

[71] H. He, J. L. Boyd-Graber, K. Kwok, and H. Daumé, III, "Opponent modeling in deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1804–1813.

[72] X. Kong, B. Xin, F. Liu, and Y. Wang, "Effective master–slave communication on a multi-agent deep reinforcement learning system," in *Proc. 31st Conf. Hierarchical Reinforcement Learn. Workshop (NIPS)*, 2017, pp. 1–6.

[73] X. Kong, B. Xin, F. Liu, and Y. Wang, "Revisiting the master–slave architecture in multi-agent deep reinforcement learning," 2017. [Online]. Available: arXiv:1712.07305.

[74] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor–critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[75] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 22nd AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.

[76] A. Harati, M. N. Ahmadabadi, and B. N. Araabi, "Knowledge-based multiagent credit assignment: A study on task type and critic information," *IEEE Syst. J.*, vol. 1, no. 1, pp. 55–67, Sep. 2007.

[77] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971.

[78] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 1889–1897.

[79] V. R. Konda and J. N. Tsitsiklis, "Actor–critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[80] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1928–1937.

[81] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2014, pp. 387–395.

[82] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," 2015. [Online]. Available: arXiv:1512.04455.

[83] A. A. Rusu *et al.*, "Progressive neural networks," 2016. [Online]. Available: arXiv:1606.04671.

[84] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, 2017.

[85] H. Yin and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proc. 31st AAAI Conf. Artif. Intell.*, Jan. 2017, pp. 1640–1646.

[86] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor–MIMIC: Deep multitask and transfer reinforcement learning," 2015. [Online]. Available: arXiv:1511.06342.

[87] M. Egorov, *Multi-Agent Deep Reinforcement Learning*, Stanford Univ., Stanford, CA, USA, 2016.

[88] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Auton. Robots*, vol. 31, no. 4, p. 299, 2011.

[89] P. Zhu, X. Li, and P. Poupart, "On improving deep reinforcement learning for POMDPs," 2017. [Online]. Available: arXiv:1704.07978.

[90] D. Luviano-Cruz, F. Garcia-Luna, L. Perez-Dominguez, and S. Gadi, "Multi-agent reinforcement learning using linear fuzzy model applied to cooperative mobile robots," *Symmetry*, vol. 10, no. 10, p. 461, 2018.

[91] A. Prasad and I. Dusparic, "Multi-agent deep reinforcement learning for zero energy communities," 2018. [Online]. Available: arXiv:1810.03679.

[92] S Kumar, P. Shah, D. Hakkani-Tur, and L. Heck, "Federated control with hierarchical multi-agent deep reinforcement learning," 2017. [Online]. Available: arXiv:1712.08266.

[93] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proc. 16th Int. Conf. Auton. Agents Multiagent Syst.*, May 2017, pp. 464–473.

[94] E. M. de Cote, A. Lazaric, and M. Restelli, "Learning to cooperate in multi-agent social dilemmas," in *Proc. ACM 5th Int. Joint Conf. Auton. Agents Multiagent Syst.*, May 2006, pp. 783–785.

[95] M. Zinkevich, A. Greenwald, and M. L. Littman, "Cyclic equilibria in Markov games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1641–1648.

[96] J. Perolat, B. Piot, B. Scherrer, and O. Pietquin, "On the use of non-stationary strategies for solving two-player zero-sum Markov games," in *Proc. 19th Int. Conf. Artif. Intell. Stat.*, May 2016, pp. 893–901.

[97] M. Kleiman-Weiner, M. K. Ho, J. L. Austerweil, M. L. Littman, and J. Tenenbaum, "Coordinate to cooperate or compete: Abstract goals and joint intentions in social interaction," in *Proc. 38th Annu. Conf. Cogn. Sci. Soc.*, Jan. 2016, pp. 1679–1684.

[98] J. Pérolat, J. Z. Leibo, V. F. Zambaldi, C. Beattie, K. Tuyls, and T. Graepel, "A multi-agent reinforcement learning model of common-pool resource appropriation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3643–3652.

[99] M. A. Janssen, R. Holahan, A. Lee, and E. Ostrom, "Lab experiments for the study of social-ecological systems," *Science*, vol. 328, no. 5978, pp. 613–617, 2010.

[100] M. Huttenrauch, A. Sosic, and G. Neumann, "Guided deep reinforcement learning for swarm systems," 2017. [Online]. Available: arXiv:1709.06011.

[101] F. A. Oliehoek, "Decentralized POMDPs," in *Reinforcement Learning*. Berlin, Germany: Springer, 2012, pp. 471–503.

[102] J. A. Calvo and I. Dusparic, "Heterogeneous multi-agent deep reinforcement learning for traffic lights control," in *Proc. 26th Irish Conf. Artif. Intell. Cogn. Sci.*, 2018, pp. 1–12.

[103] M. Kurek and W. Jaskowski, "Heterogeneous team deep *Q*-learning in low-dimensional multi-agent environments," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Sep. 2016, pp. 1–8.

[104] T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Multi-agent deep reinforcement learning with human strategies," 2018. [Online]. Available: arXiv:1806.04562.

[105] D. Noureddine, A. Gharbi, and S. Ahmed, "Multi-agent deep reinforcement learning for task allocation in dynamic environment," in *Proc. 12th Int. Conf. Softw. Technol. (ICSOFT)*, 2017, pp. 17–26.

[106] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," 2018. [Online]. Available: arXiv:1802.06444.

[107] K. Schmid, L. Belzner, T. Gabor, and T. Phan, "Action markets in deep multi-agent reinforcement learning," in *Proc. Int. Conf. Artif. Neural Netw.*, Oct. 2018, pp. 240–249.

[108] A. Lerer and A. Peysakhovich, "Maintaining cooperation in complex social dilemmas using deep reinforcement learning," 2017. [Online]. Available: arXiv:1707.01068.

[109] B. Piot, M. Geist, and O. Pietquin, "Bridging the gap between imitation learning and inverse reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1814–1826, Aug. 2017.

[110] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3909–3917.

[111] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6765–6774).

[112] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4299–4307.

[113] N. D. Nguyen, S. Nahavandi, and T. Nguyen, "A human mixed strategy approach to deep reinforcement learning," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2018, pp. 4023–4028.

[114] S. Nahavandi, "Trusted autonomy between humans and robots: Toward human-on-the-loop in robotics and autonomous systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 3, no. 1, pp. 10–17, Jan. 2017.

[115] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[116] S. Gu, T. P. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep *Q*-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 2829–2838.

[117] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2786–2793.

[118] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7559–7566.

[119] I. V. Serban, C. Sankar, M. Pieper, J. Pineau, and Y. Bengio, "The bottleneck simulator: A model-based deep reinforcement learning approach," 2018. [Online]. Available: arXiv:1807.04723.

[120] D. Corneil, W. Gerstner, and J. Brea, "Efficient model-based deep reinforcement learning with variational state tabulation," 2018. [Online]. Available: arXiv:1802.04325.

[121] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," 2018. [Online]. Available: arXiv:1803.11485.

[122] J. Foerster *et al.*, "Learning with opponent-learning awareness," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, Jul. 2018, pp. 122–130.

[123] S.-M. Hung and S. N. Givigi, "A *Q*-learning approach to flocking with UAVs in a stochastic environment," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 186–197, Jan. 2017.

[124] D. Schwab, Y. Zhu, and M. Veloso, "Zero shot transfer learning for robot soccer," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, Jul. 2018, pp. 2070–2072.

[125] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intell. Transp. Syst.*, vol. 11, no. 7, pp. 417–423, 2017.

[126] A. Kattepur, H. K. Rath, A. Simha, and A. Mukherjee, "Distributed optimization in multi-agent robotics for industry 4.0 warehouses," in *Proc. 33rd Annu. ACM Symp. Appl. Comput.*, Apr. 2018, pp. 808–815.

[127] M. S. Rahman, M. A. Mahmud, H. R. Pota, M. J. Hossain, and T. F. Orchi, "Distributed multi-agent-based protection scheme for transient stability enhancement in power systems," *Int. J. Emerg. Elect. Power Syst.*, vol. 16, no. 2, pp. 117–129, 2015.

[128] J. He, J. Peng, F. Jiang, G. Qin, and W. Liu, "A distributed *Q*-learning spectrum decision scheme for cognitive radio sensor network," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 5, 2015, Art. no. 301317.

[129] P. C. Pendharkar and P. Cusatis, "Trading financial indices with reinforcement learning agents," *Expert Syst. Appl.*, vol. 103, pp. 1–13, Aug. 2018.

[130] O. Brandouy, P. Mathieu, and I. Veryzhenko, "On the design of agent-based artificial stock markets," in *Proc. Int. Conf. Agents Artif. Intell.*, Jan. 2011, pp. 350–364.

[131] M. Hussin, N. A. W. A. Hamid, and K. A. Kasmiran, "Improving reliability in resource management through adaptive reinforcement learning for distributed systems," *J. Parallel Distrib. Comput.*, vol. 75, pp. 93–100, Jan. 2015.

[132] B. Fernandez-Gauna, I. Etxeberria-Agiriano, and M. Grana, "Learning multirobot hose transportation and deployment by distributed round-robin *Q*-learning," *PLoS ONE*, vol. 10, no. 7, 2015, Art. no. e0127129.

[133] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.

[134] Y. Shoham, R. Powers, and T. Grenager, "If multi-agent learning is the answer, what is the question?" *Artif. Intell.*, vol. 171, no. 7, pp. 365–377, 2007.

Thanh Thi Nguyen received the Ph.D. degree in mathematics and statistics from Monash University, Melbourne, VIC, Australia, in 2013.

He was a Visiting Scholar with the Computer Science Department, Stanford University, Stanford, CA, USA, in 2015 and the Edge Computing Lab, John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA, in 2019. He is currently a Senior Lecturer with the School of Information Technology, Deakin University, Burwood, VIC, Australia. He has expertise in various areas, including artificial intelligence, deep learning, deep reinforcement learning, cyber security, IoT, and data science.

Dr. Nguyen received the Alfred Deakin Postdoctoral Research Fellowship in 2016 and the European–Pacific Partnership for the ICT Expert Exchange Program Award from European Commission in 2018.

Ngoc Duy Nguyen received the master's degree in computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2011. He is currently pursuing the Ph.D. degree with the Institute for Intelligent Systems Research and Innovation, Deakin University, Waurn Ponds, VIC, Australia.

His research interests involve machine learning, optimization problems, and system design.

Mr. Nguyen received the Best Master's Thesis Award from the Department of Information and Communication Engineering, Sungkyunkwan University.

Saeid Nahavandi (Senior Member, IEEE) received the Ph.D. degree from Durham University, Durham, U.K., in 1991.

He is an Alfred Deakin Professor, a Pro-Vice-Chancellor (Defence Technologies), a Chair of Engineering, and the Director of the Institute for Intelligent Systems Research and Innovation, Deakin University, Waurn Ponds, VIC, Australia. He has published over 600 papers in various international journals and conferences. His research interests include modeling of complex systems, robotics, and haptics.

Prof. Nahavandi is the Co-Editor-in-Chief of the IEEE SYSTEMS JOURNAL, an Associate Editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and an Editorial Board Member of IEEE ACCESS. He is a fellow of the Engineers Australia and the Institution of Engineering and Technology.