Is multiagent deep reinforcement learning the answer or the question? A brief survey

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor
University of Alberta CCIS 3-232 Edmonton, Canada
Borealis AI

arXiv:1810.05587v1 [cs.MA] 12 Oct 2018

Abstract

Deep reinforcement learning (DRL) has achieved outstanding results in recent years. This has led to a dramatic increase in the number of applications and methods. Recent works have explored learning beyond single-agent scenarios and have considered multiagent scenarios. Initial results report successes in complex multiagent domains, although there are several challenges to be addressed. In this context, first, this article provides a clear overview of current multiagent deep reinforcement learning (MDRL) literature. Second, it provides guidelines to complement this emerging area by (i) showcasing examples on how methods and algorithms from DRL and multiagent learning (MAL) have helped solve problems in MDRL and (ii) providing general lessons learned from these works. We expect this article will help unify and motivate future research to take advantage of the abundant literature that exists in both areas (DRL and MAL) in a joint effort to promote fruitful research in the multiagent community.

Keywords: Multiagent learning, multiagent systems, multiagent reinforcement learning, deep reinforcement learning, survey

1. Introduction

Almost 20 years ago Stone and Veloso's seminal survey [1] laid the groundwork for defining the area of multiagent systems (MAS) and its open problems in the context of AI. About ten years ago, Shoham, Powers, and Grenager [2] noted that the literature on multiagent learning (MAL) was growing and it was not possible to enumerate all relevant articles. Since then, the number of published MAL works continues to steadily rise, which led to different surveys on the area, ranging from analyzing the basics of MAL and their challenges [3, 4, 5], to addressing specific subareas: game theory and MAL [2, 6], cooperative scenarios [7, 8], and evolutionary dynamics of MAL [9]. In just the last couple of years, two surveys related to MAL have been published: learning in non-stationary environments [10] and agents modeling agents [11].

The research interest in MAL has been accompanied by successes, first, in single-agent video games [12]; more recently, in two-player games, e.g., playing Go [13, 14], poker [15], and games of two competing teams, e.g., DOTA 2 [16].

While different techniques and algorithms were used in the above scenarios, in general, they are all a combination of techniques from two main areas: reinforcement learning (RL) [17] and deep learning [18].

RL is an area of machine learning where an agent learns by interacting (i.e., taking actions) within a dynamic environment. However, one of its main drawbacks is the need for defining

(hand-crafting) features used to learn. In recent years, deep learning has had successes in different areas such as computer vision and natural language processing [18]. One of the key aspects of deep learning is the use of *neural networks* (NNs) that can find compact representations in high-dimensional data [19], thus eliminating the need for manual feature design.

Complementing each other, deep learning and reinforcement learning generated a new area: deep reinforcement learning (DRL) [19], in which deep neural networks are trained to approximate the optimal policy and/or the value function, where the promise of generalization is expected to be delivered by the representation ability of deep NNs (as the function approximator). One of the key advantages of DRL is that it enables RL to scale to problems with high-dimensional state and action spaces.

DRL has been regarded as an important component in constructing general AI systems [20] and has been successfully integrated with other techniques, e.g., search [13], planning [21], and more recently with multiagent systems, with an emerging area of *multiagent deep reinforcement learning (MDRL)*.

Learning in multiagent settings is fundamentally more difficult than the single-agent case with problems like non-stationarity, curse of dimensionality, and multiagent credit assignment [2, 5, 10, 22, 23, 24]. Despite this complexity, top AI conferences like AAAI, ICML, ICLR, IJCAI and NIPS, and specialized conferences such as AAMAS, have published works reporting successes in MDRL. In light of these works, we believe it is pertinent to first, have an overview of the recent MDRL works, and second, understand how these recent works relate to the existing literature.

This paper contributes to the state of the art with a brief survey of the current works in MDRL in an effort to complement existing surveys [10, 11, 19]. In particular, we identified four categories to group recent works (see Figure 1):

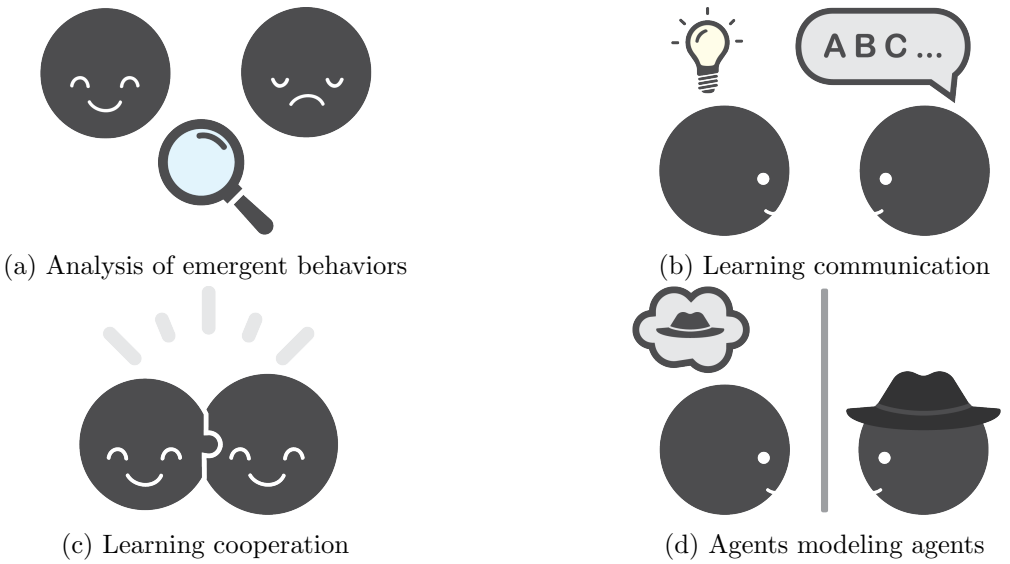
- Analysis of emergent behaviors
- Learning communication
- Learning cooperation
- Agents modeling agents

For each category we provide a description as well as outline the recent works (see Section 3 and Tables 1–4). Then, we take a step back and reflect on how these new works relate to the existing literature (see Section 4). In that context, first, we present examples on how methods and algorithms from (non-deep) MAL and RL helped to solve problems in MDRL (see Section 4.1). Second, we present general *lessons learned* from the existing MDRL works (see Section 4.2). Third, we outline some open questions (see Section 4.3). Lastly, we present some conclusions from this work (see Section 5).

Our goal is to outline a recent and active area (i.e., MDRL), as well as to motivate future research to take advantage of the ample and existing literature in multiagent learning. We expect that researchers with experience on either DRL or MAL could benefit from this article to have a common understanding about recent works and open problems in MDRL and to avoid having scattered sub-communities with little interaction [2, 10, 11, 25].

2. Single-agent learning

This section presents the formalism of reinforcement learning and its main components before outlining *deep* reinforcement learning along with recent algorithms. For a more detailed description



(a) Analysis of emergent behaviors (b) Learning communication

(c) Learning cooperation (d) Agents modeling agents

Figure 1: Categories of different MDRL works. (a) Analysis of emergent behaviors: evaluate DRL algorithms in multiagent scenarios. (b) Learning communication: agents learn with actions and through messages. (c) Learning cooperation: agents learn to cooperate using only actions and local observations. (d) Agents modeling agents: agents reason about others to fulfill a task (e.g., cooperative or competitive). For a more detailed description see Sections 3.3–3.6 and Tables 1–4.

we refer the reader to excellent books and surveys on the area [17, 19, 26].

2.1. Reinforcement learning

RL formalizes the interaction of an agent with an environment using a Markov decision process (MDP) [27]. An MDP is defined by the tuple $\langle S, \mathcal{A}, R, T, \gamma \rangle$ where S represents a finite set of states. $\mathcal{A}$ represents a finite set of actions. The transition function $T : S \times \mathcal{A} \to \Delta(S)$ maps each state-action pair to a probability distribution over the possible successor states, where $\Delta(S)$ denotes the set of all probability distributions over S. Thus, for each $s, s' \in S$ and $a \in \mathcal{A}$, the function T determines the probability of a transition from state s to state s' after executing action a. The reward function $R : S \times \mathcal{A} \times S \to \mathbb{R}$ defines the immediate and possibly stochastic reward that an agent would receive given that the agent executes action a while in state s and it is transitioned to state s'. $\gamma \in [0, 1]$ represents the discount factor that balances the trade-off between immediate rewards and future rewards.

MDPs are adequate models to obtain optimal decisions in *single* agent environments. Solving an MDP will yield a policy $\pi : S \to A$, which is a mapping from states to actions. An optimal policy π^* is the one that maximizes the expected discounted sum of rewards. There are different techniques for solving MDPs assuming a complete description of all its elements. One of the most common techniques is the value iteration algorithm [28] which is based on the Bellman equation:

$$V^{\pi}(s) = \sum_{a \in A} \pi(s, a) \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V^{\pi}(s')]$$

This equation expresses the *value* of a state which can be used to obtain the optimal policy $\pi^* = \arg\max_{\pi} V^{\pi}(s)$, i.e., the one that maximizes that value function, and the optimal value

function $V^*(s)$.

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S.$$

Value iteration requires a complete and accurate representation of states, actions, rewards, and transitions. However, this may be difficult to obtain in many domains. For this reason, RL algorithms often learn from experience interacting with the environment in discrete time-steps.

Q-learning. One of the most well known algorithms for RL is Q-learning [29]. It has been devised for stationary, single-agent, fully observable environments with discrete actions. A Q-learning agent keeps the estimate of its expected payoff starting in state s, taking action a as $\hat{Q}(s, a)$. Each tabular entry $\hat{Q}(s, a)$ is an estimate of the corresponding optimal Q^* function that maps state-action pairs to the discounted sum of future rewards starting with action a at state s and following the optimal policy thereafter. Each time the agent transitions from a state s to a state s' via action a receiving payoff r, the Q table is updated as follows:

$$\hat{Q}(s, a) = \hat{Q}(s, a) + \alpha[(r + \gamma \max_{b} \hat{Q}(s', b)) - \hat{Q}(s, a)]$$

with the learning rate $\alpha \in [0, 1]$ and typically decreasing over the course of many iterations. Q-learning is proven to converge towards Q^* if each state-action pair is visited *infinitely* often under specific parameters [29, 30].

REINFORCE (Monte Carlo policy gradient). In contrast to value-based methods for the RL problem, policy gradient methods can learn parameterized policies without using intermediate value estimates. Policy parameters are learned based on gradients of some performance measure with gradient ascent method [31]. For example, REINFORCE [32] uses estimated return by Monte Carlo methods with full episode trajectories to learn policy parameters θ where $\pi(a|s, \theta) \approx \pi(a|s)$.

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}$$

G_t represents the return, α is the learning rate, and $A_t \sim \pi$. Policy gradient methods can have high variance. To address this challenge, actor-critic algorithms, which approximate policy gradient methods, have been proposed. Actor-critic algorithms combine value-based and policy-gradient based methods [33]. The actor represents the policy, i.e., action-selection mechanism, whereas a critic is used for the value function learning. The policy update takes into account critic's value estimate reducing the variance compared to vanilla policy gradient methods. When the critic also learns a state-action function besides a state value function, an *advantage function* can be computed as a baseline for variance reduction [17].

Policy gradient methods have a clear connection with deep reinforcement learning since *the policy might be represented by a neural network* whose input is a representation of the state, whose output are action selection probabilities, and whose weights are the policy parameters.

2.2. Deep reinforcement learning

Even when tabular RL methods such as Q-learning had successes, there were drawbacks: RL could be slow to learn in large state spaces, the methods did not generalize (across the state space), and state representations needed to be hand-specified [17]. Fortunately, these challenges can be addressed by using deep learning, i.e., neural networks as function approximators as follows:

$$Q(s, a; \theta) \approx Q(s, a)$$

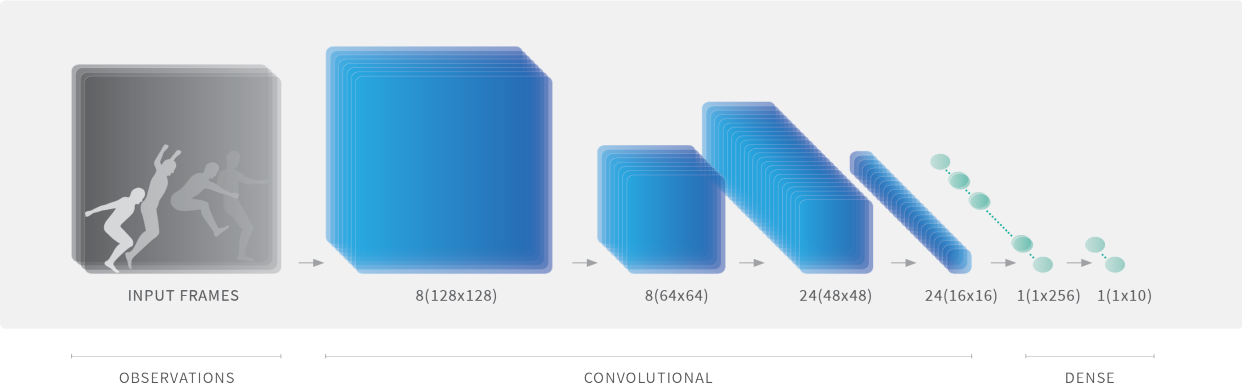


Figure 2: Deep Q-Network (DQN) [12], an example of standard Deep RL neural network architecture, composed of several layers: *Convolutional* layers employ filters to learn features from high-dimensional data with a much smaller number of neurons and *Dense* layers are fully-connected layers. The last layer represents the actions the agent can take (in this case, 10 possible actions). Deep Recurrent Q-Network (DRQN) [37], which extends DQN to partially observable domains [38], is identical to this setup except the penultimate layer (1×256 Dense layer) is replaced with a recurrent LSTM layer [39].

where θ represents the neural network weights. First, deep learning helps to generalize across states improving the sample efficiency for large state-space RL problems. Second, deep learning can be used to reduce (or eliminate) the need for manually designing features to represent state information [18].

However, extending deep learning to RL problems comes with additional challenges including non-i.i.d. data for highly correlated sequential agent interactions and non-stationary data distribution due to *learning* agent behavior [34]. Below we mention how the existing DRL methods address these challenges when briefly reviewing value-based methods, such as DQN [12]; policy gradient methods, like PPO [35]; and actor-critic methods like A3C [36]. We refer the reader to a recent survey on single-agent DRL [19] for a more detailed discussion of the literature.

Value-based methods. The major breakthrough work blending deep learning with Q-learning was Deep Q-Network (DQN) [12]. DQN uses a deep neural network for function approximation (see Figure 2) and maintains an *experience replay (ER)* buffer to store interactions $\langle s, a, r, s' \rangle$. In contrast to tabular Q-learning, DQN keeps an additional copy of neural network parameters, i.e., θ^-, for the target network besides the θ parameters to stabilize the learning, i.e., to alleviate the non-stationary data distribution. For each training iteration i, DQN minimizes the mean-squared error (MSE) between the Q-network and its target network using the loss function:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2]$$

where target network parameters θ^- are set to Q-network parameters θ periodically and mini-batches of $\langle s, a, r, s' \rangle$ tuples are sampled from the ER buffer, as depicted in Figure 3.

The ER buffer provides stability for learning as random batches sampled from the buffer helps alleviating the problems caused by the non-i.i.d. data. However, it comes with disadvantages, such as memory requirements and a mismatch between buffer content from earlier policy and from the current policy [40].

DQN assumes full state observability and it does not perform well in *partially observable* domains [38]. Deep Recurrent Q-Networks (DRQN) [37] proposed using *recurrent neural networks*,

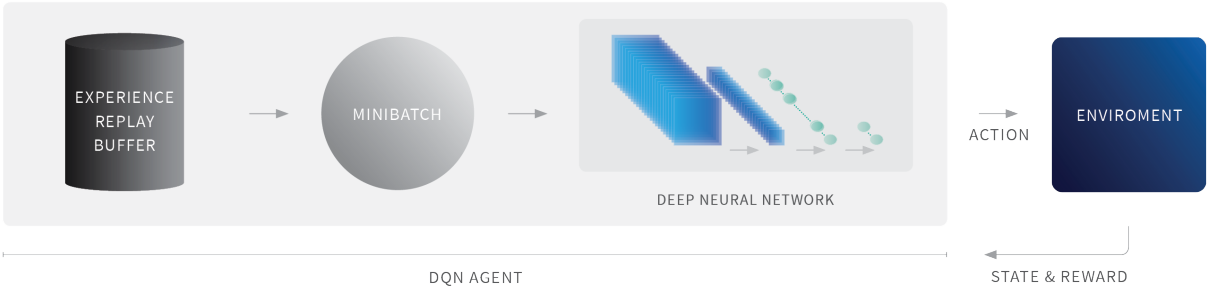


Figure 3: Representation of a DQN agent that uses an experience replay buffer to keep $\langle s, a, r, s' \rangle$ tuples to use within minibatch updates. The policy is parametrized with a NN that outputs an action at every timestep.

in particular, LSTMs (Long Short-Term Memory) [39] in DQN, for this setting. The main change to the standard DQN is to replace the first post-convolutional fully connected layer with an LSTM layer (see Figure 2). With this addition, DRQN has memory capacity so that it can even work with only one input rather than a stacked input of consecutive frames.

Policy gradient methods. For many tasks, particularly for physical control, the action space is continuous and high dimensional where DQN is not suitable. Deep deterministic policy gradient (DDPG) [41] is a model-free off-policy actor-critic algorithm for such domains where several adjustments to the DQN algorithm have been made. An actor-critic based model is employed where target network parameters slowly change, in contrast to hard reset to learned network parameters as in DQN. Given the off-policy nature, DDPG generates exploratory behavior by adding sampled noise from some noise processes to its actor policy. The authors also used batch normalization [42] to ensure generalization across many different tasks without performing manual normalizations.

A3C (Asynchronous Advantage Actor-Critic) [40] is an algorithm that employs a *parallelized* asynchronous training scheme (using multiple CPU threads) for efficiency. It is an on-policy RL method that does not use an experience replay buffer. A3C allows multiple workers to simultaneously interact with the environment and compute gradients locally. All the workers pass their computed local gradients to a global NN which performs the optimization and synchronizes with the workers asynchronously (see Figure 4). There is also A2C (Advantage Actor-Critic) method that combines all the gradients from all the workers to update the global NN *synchronously*.

The UNREAL framework [36] is built on top of A3C. In particular, UNREAL proposes unsupervised *auxiliary tasks* (e.g., reward prediction) to speed up the learning process which require no additional feedback from the environment. In contrast to A3C, UNREAL uses an ER buffer that is sampled with more priority given to interactions with positive rewards to improve the critic network.

Another distributed architecture is the Importance Weighted Actor-Learner Architecture (IMPALA) [43]. Unlike A3C or UNREAL, IMPALA actors communicate *trajectories of experience* (sequences of states, actions, and rewards) to a centralized learner, thus IMPALA decouples acting from learning.

TRPO (Trust Region Policy Optimization) [35] and PPO (Proximal Policy Optimization) [44] are state-of-the art policy gradient algorithms. Compared to vanilla policy gradient algorithms, PPO is designed to prevent abrupt changes in policies during training by incorporating the change in policy to the loss function. Another advantage of PPO is that it can be used in a distributed

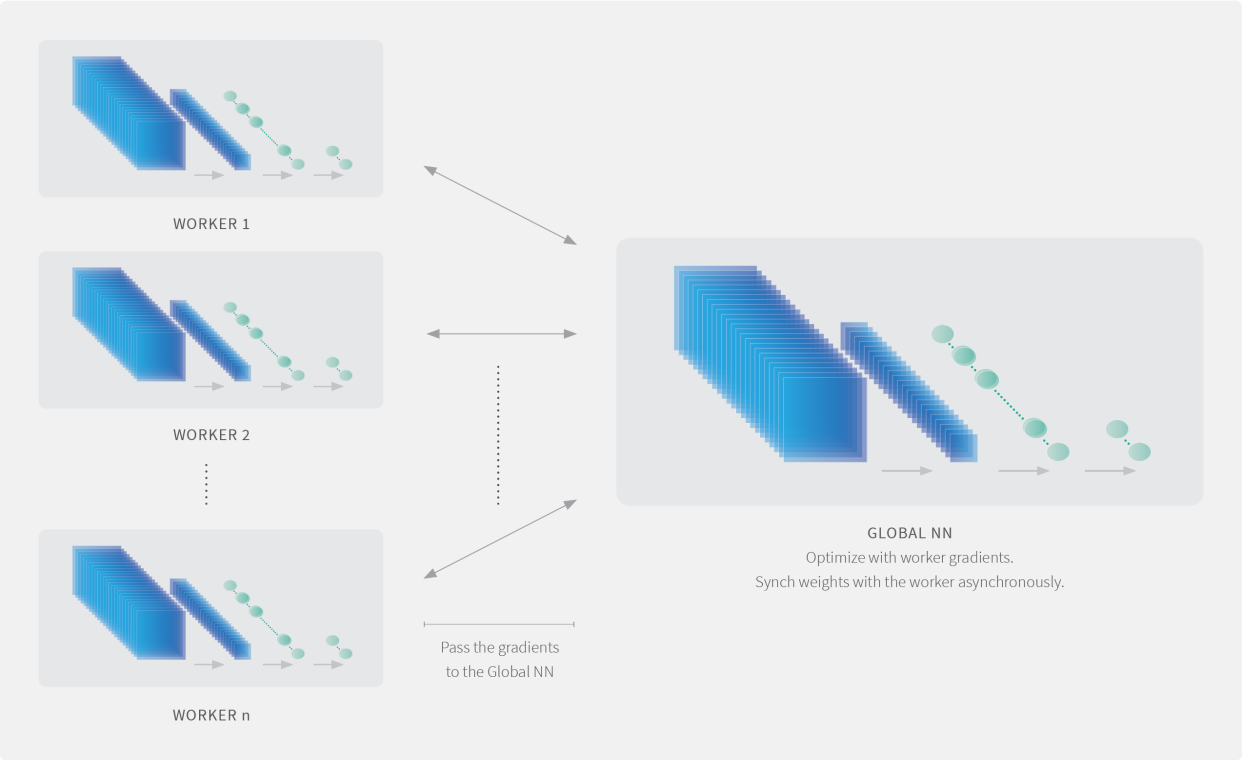


Figure 4: Asynchronous Advantage Actor-Critic (A3C) employs multiple (CPUs) workers without needing an ER buffer. Each worker has its own NN and independently interacts with the environment to compute the loss and gradients. Workers then pass computed gradients to the global NN that optimizes the parameters and synchronizes with the worker *asynchronously*. This distributed system is designed for single-agent deep RL. A major advantage of this method is that it does not require a GPU; it can use multiple CPU threads on standard computers.

fashion, DPPO [45]. Note that *distributed approaches* like DPPO or A3C use only parallelization to improve the learning for single agent DLR and they should not be considered MDRL approaches.

We have reviewed recent algorithms in DRL, while the list is not exhaustive, it provides an overview of the different state-of-art techniques and algorithms which will become useful while describing the MDRL techniques in the next section.

3. Multiagent Deep Reinforcement Learning (MDRL)

First, we briefly introduce the general framework on multiagent learning and then we dive into the categories and the research on MDRL.

3.1. Multiagent Learning

Learning in a multiagent environment is inherently more complex than in the single-agent case, as agents interact at the same time with environment and potentially with each other [5]. Directly using single-agent algorithms in a multiagent setting is a natural approach, called *independent learners* [46], even if assumptions under which these algorithms were derived are violated. In particular the *Markov property*[1] becomes invalid since the environment is no longer stationary [4,

[1]The future dynamics, transitions, and rewards fully depend on the current state.

6, 47]. This approach ignores the multiagent nature of the setting entirely and it can fail when an opponent adapts or learns, for example, based on the past history of interactions [2].

In order to understand why multiagent domains are non-stationary from agents' local perspectives, consider a stochastic (also known as Markov) game $(S, \mathcal{N}, A, T, R)$, which can be seen as an extension of an MDP to multiple agents [48]. One key distinction is that the transition, T, and reward function, R, depend on the actions of all, $\mathcal{N}$, agents.

Given a learning agent i and using the common shorthand notation $-\boldsymbol{i} = \mathcal{N} \setminus \{i\}$ for the set of opponents, the value function now depends on the joint action $\boldsymbol{a} = (a_i, \boldsymbol{a_{-i}})$, and the joint policy $\boldsymbol{\pi}(s, \boldsymbol{a}) = \prod_j \pi_j(s, a_j)$:

$$V_i^{\boldsymbol{\pi}}(s) = \sum_{\boldsymbol{a} \in A} \boldsymbol{\pi}(s, \boldsymbol{a}) \sum_{s' \in S} T(s, a_i, \boldsymbol{a_{-i}}, s')[R(s, a_i, \boldsymbol{a_{-i}}, s') + \gamma V_i(s')]. \tag{1}$$

Consequently, the optimal policy is a best response dependent on the other agents' policies,

$$\begin{aligned} \pi_i^*(s, a_i, \boldsymbol{\pi_{-i}}) &= BR_i(\boldsymbol{\pi_{-i}}) = \arg\max_{\pi_i} V_i^{(\pi_i, \boldsymbol{\pi_{-i}})}(s) \\ &= \arg\max_{\pi_i} \sum_{\boldsymbol{a} \in A} \pi_i(s, a_i) \boldsymbol{\pi_{-i}}(s, \boldsymbol{a_{-i}}) \sum_{s' \in S} T(s, a_i, \boldsymbol{a_{-i}}, s')[R(s, a_i, \boldsymbol{a_{-i}}, s') + \gamma V_i^{(\pi_i, \boldsymbol{\pi_{-i}})}(s')]. \end{aligned}$$

Specifically, the opponents' joint policy $\boldsymbol{\pi_{-i}}(s, \boldsymbol{a_{-i}})$ can be non-stationary, thus becoming the parameter of the best response function.

There are other common problems in MAL, including convergence [2, 49, 50], action shadowing [24, 51], the curse of dimensionality [5], and multiagent credit assignment [23]. Describing each problem is out of the scope of this survey. However, we refer the interested reader to excellent resources on general MAL [4, 22, 52], as well as surveys in specific areas: game theory and multiagent reinforcement learning [5, 6], cooperative scenarios [7, 8], evolutionary dynamics of multiagent learning [9], learning in non-stationary environments [10], and agents modeling agents [11].

3.2. MDRL categorization

In the previous section we outlined *some* recent works in single-agent DRL since an exhaustive list is out of the scope of this article. This explosion of works has led DRL to be extended and combined with other techniques [19]. One natural extension to DRL is to test whether these approaches could be applied in a multiagent environment.

We analyzed of the most recent works (that are not covered by previous MAL surveys [10, 11]) that have a clear connection with MDRL.[2] We propose 4 categories which take inspiration from previous surveys [1, 5, 7, 11] and that conveniently describe and represent current works.[3]

- *Analysis of emergent behaviors.* These works do not propose learning algorithms — their main focus is to analyze and evaluate DRL algorithms, e.g., DQN [53, 54, 55] and others [56, 57, 58], in a multiagent environment. See Section 3.3 and Table 1.

[2] We do not consider genetic algorithms or swarm intelligence in this work.

[3] Note that some of these works fit into two categories, however, for the ease of exposition when describing them we only do so in one category.

- *Learning communication* [56, 57, 59, 60, 61]. These works explore a sub-area that is attracting attention[4] and that had not been explored much in the MAL literature. See Section 3.4 and Table 2.

- *Learning cooperation.* While learning to communicate is an emerging area, fostering cooperation in learning agents has a long history of research in MAL [7, 8]. The works in this category mostly take inspiration from MAL to extend to the MDRL setting, with both value-based methods [62, 63, 64, 65, 66, 67, 68, 69] and policy gradients methods [61, 70, 71]. See Section 3.5 and Table 3.

- *Agents modeling agents.* Albrecht and Stone [11] presented a thorough survey in this topic and we have found many works that fit into this category in the MDRL setting, some taking inspiration from DRL [72, 73, 74], and others from MAL [69, 75, 76, 77, 78]. Modeling agents is helpful not only to cooperate, but also for modeling opponents [69, 72, 74, 75], inferring hidden goals [73], and accounting for the learning behavior of other agents [76]. See Section 3.6 and Table 4.

In the rest of this section we describe each category along with the summaries of related works.

3.3. Emergent behaviors

A group of works have studied and analyzed the emergence of behaviors (e.g., cooperative or competitive) using *independent* DRL agents in different settings.

One of the earliest works by Tampuu et al. [53] placed two independent DQN learning agents to play the Pong game. Their focus was to adapt the reward function for the learning agents, which resulted in either cooperative or competitive emergent behaviors.

Leibo et at. [54] also studied independent DQNs although in the context of *sequential social dilemmas*.[5] The focus of this work was to highlight that cooperative or competitive behaviors exist not only as discrete (atomic) actions, but they are temporally extended (over policies).

Recently, Bansal et al. [58] explored the emergent behaviors in competitive scenarios using the MuJoCo simulator [79]. They trained independent learning agents with PPO [44] plus two main modifications to deal with the MAL nature of the problem. First, they used *exploration rewards* which are dense rewards that allow agents to learn basic (non-competitive) behaviors — this type of reward is annealed through time giving more weight to the environmental (competitive) reward. Second, they propose *opponent sampling* which maintains a pool of older versions of the opponent to sample from, in contrast to using the most recent version.

Raghu et al. [55] investigated how DRL algorithms (DQN, A2C, and PPO) performed in a family of two-player zero-sum games with tunable complexity, called Erdos-Selfridge-Spencer games. Their reasoning is threefold: (i) these games provide a parametrized family of environments where (ii) optimal behavior can be completely characterized, and (iii) support multiagent play. Their work showed that algorithms can exhibit wide variation in performance as the algorithms are tuned to the game's difficulty.

Lazaridou et al. [56] proposed a framework for language learning that relies on multiagent communication. The agents, represented by (feed-forward) neural networks, need to develop an

[4] For example, see recent workshops on Emergent Communication: `https://sites.google.com/site/emecom2017/` and `https://sites.google.com/site/emecom2018/`

[5] A sequential social dilemma is a Markov game that satisfies certain social dilemma inequalities [54].

Table 1: These papers analyze *emergent behaviors* in MDRL. A detailed description is given in Section 3.3.

Work	Summary
Tampuu et al. [53]	Train DQN agents to play Pong.
Leibo et al. [54]	Train DQN agents to play sequential social dilemmas.
Bansal et al. [58]	Train PPO agents in competitive MuJoCo scenarios.
Raghu et al. [55]	Train PPO, A3C, and DQN agents in attacker-defender games.
Lazaridou et al. [56]	Train agents represented with NN to learn a communication language.
Mordatch and Abbeel [57]	Learn communication with an end-to-end differentiable model to train with backpropagation.

Table 2: These papers propose algorithms for *learning communication*, together with a deep neural network architecture. A more detailed description is given in Section 3.4.

Algorithm	Architecture	Summary
RIAL [59]	DRQN	Use a single network (parameter sharing) to train agents that take environmental and communication actions.
DIAL [59]	DRQN	Use gradient sharing during learning and communication actions during execution.
CommNet [60]	Multilayer NN	Use a continuous vector channel for communication on a single network.
BiCNet [61]	Bidirectional RNN	Use the actor-critic paradigm where communication occurs in the latent space.

Table 3: These papers aim to *learn cooperation*. We highlight the closest work (DRL or MAL) in which it is based. A more detailed description is given in Section 3.5.

Algorithm	Based on	Summary
Fingerprints [62]	IQN	Deal with ER problems in MDRL by conditioning the value function on a fingerprint that disambiguates the age of the sampled data.
Lenient-DQN [63]	DQN	Achieve cooperation by leniency, optimism in the value function by forgiving suboptimal (low-rewards) actions.
Hysteretic-DRQN [64]	DRQN	Achieve cooperation by using two learning rates, depending on the updated values together with multitask learning via policy distillation.
WDDQN [65]	DQN	Achieve cooperation by leniency, weighted double estimators, and a modified prioritized experience replay buffer.
FTW [66]	IMPALA	Agents act in a mixed environment (composed of teammates and opponents), it proposes a two-level architecture and a population-based learning.
VDN [67]	IQN	Decompose the team action-value function into pieces across agents, where the pieces can be easily added.
QMIX [68]	VDN	Decompose the team action-value function together with a mixing network that can recombine them.
COMA [70]	-	Use a centralized critic and a counter-factual advantage function based on solving the multiagent credit assignment.
MADDPG [71]	DDPG	Use an actor-critic approach where the critic is augmented with information from other agents, the actions of all agents.

Table 4: These papers consider *agents modeling agents*. A more detailed description is given in Section 3.6.

Algorithm	Summary
DRON [72]	Have a network to infer the opponent behavior together with the standard DQN architecture.
DPIQN, DPIRQN [74]	Learn policy features from raw observations that represent high-level opponent behaviors via auxiliary tasks.
SOM [73]	Assume the reward function depends on a hidden goal of both agents and then use an agent's own policy to infer the goal of the other agent.
NFSP [75]	Compute approximate Nash equilibria via self-play and two neural networks.
DCH [69]	Policies can overfit to opponents: better compute approximate best responses to a mixture of policies.
LOLA [76]	Use a learning rule where the agent accounts for the parameter update of other agents in order to maximize its own reward.
ToMnet [77]	Use an architecture for end-to-end learning and inference of diverse opponent types.
Deep Bayes-ToMoP [78]	Best respond to opponents using Bayesian policy reuse, theory of mind, and deep networks.

emergent language to solve a task. The task is formalized as a *signaling game* [80] in which two agents, a sender and a receiver, obtain a pair of images. The sender is told one of them is the target and is allowed to send a message (from a fixed vocabulary) to the receiver. Only when the receiver identifies the target image do both agents receive a positive reward. A key objective of this work was to analyze if the agent's language could be human-interpretable, showing limited yet encouraging results.

Similarly, Mordatch and Abbeel [57] investigated the emergence of language with the difference that in their setting there were no explicit roles for the agents (i.e., sender, receiver). To learn, they proposed an end-to-end differentiable model of all agent and environment state dynamics over time to calculate the gradient of the return with backpropagation.

3.4. Learning communication

As we discussed in the previous section, one of the desired emergent behaviors of multiagent interaction is the emergence of communication [56, 57]. This setting usually considers a set of *cooperative agents* in *partially observable* environment where agents need to maximize their shared utility by means of communicating information.

RIAL (Reinforced Inter-Agent Learning) and DIAL (Differentiable Inter-Agent Learning) are two methods using deep networks to learn to communicate [59]. Both methods use a neural net that outputs the agent's Q values (as done in standard DRL algorithms) and a message to communicate to other agents in the next timestep. RIAL is based on DRQN and also uses the concept of *parameter sharing*, i.e., using a single network whose parameters are shared among all agents. In contrast, DIAL directly passes gradients via the communication channel during learning, and messages are discretized and mapped to the set of communication actions during execution.

While RIAL and DIAL used a discrete communication channel, CommNet [60] considered a continuous vector channel. Through this channel agents receive the summed transmissions of other agents. The authors assume full cooperation and train a single network for all the agents. There are two distinctive characteristics of CommNet from previous works: it allows multiple communication cycles at each timestep and a dynamic variation of agents at run time, i.e., agents come and go in the environment.

In contrast to previous approaches, in Multiagent Bidirectionally Coordinated Network (BiCNet) [61] communication takes place in the latent space (i.e., in the hidden layers). It also uses parameter sharing, however, it proposes bidirectional RNNs [81] to model the actor and critic networks of their model. Note that in BiCNeT agents do not *explicitly* share a message and thus it can be considered a method for learning cooperation.

3.5. Learning cooperation

Although *explicit communication* is a new emerging trend in MDRL, there has already been a large amount of work in MAL for cooperative settings that do not involve communication [7, 8]. Therefore, it was a natural starting point for many recent MDRL works.

Foerster et al. [62] addressed the problem of *cooperation with independent Q-learning agents*, where the agents use the standard DQN architecture of neural networks and an experience replay buffer (see Section 2.2 and Figure 3). However, the ER buffer introduces problems, such as: the dynamics that generated the data in the ER no longer reflect the current dynamics, making the experience obsolete [62]. Their solution is to add information to the experience tuple that can help to *disambiguate the age of the sampled data* from the replay memory. Two approaches were proposed: Multiagent Importance Sampling adds the probability of the joint action, and Multiagent

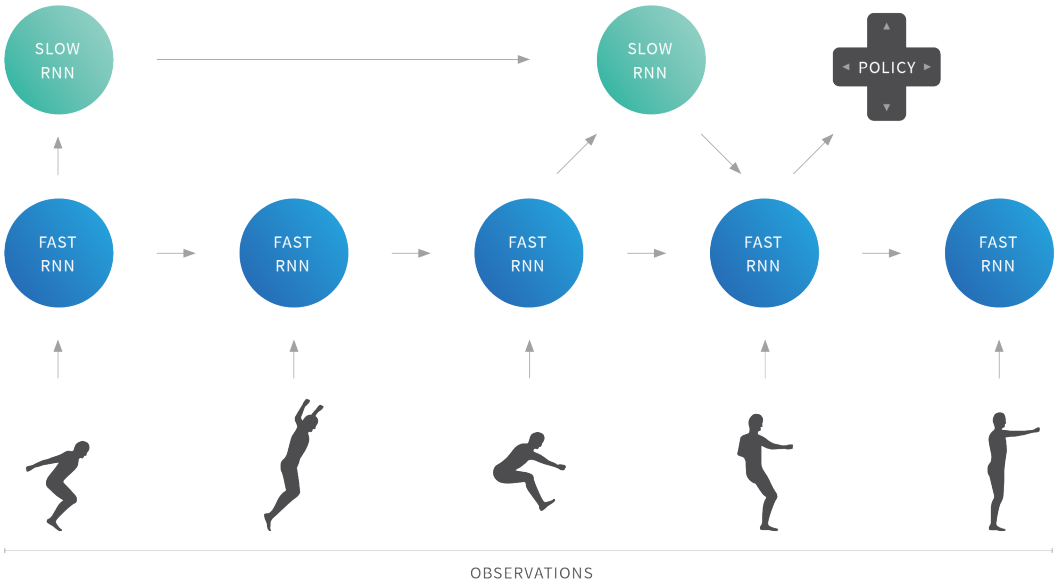


Figure 5: An schematic view of the architecture used in FTW (For the Win) [66]: two unrolled RNNs operate at different time-scales, the idea is that the *Slow RNN* helps with long term temporal correlations. Observations are latent space output of some convolutional neural network to learn non-linear features. Feudal Networks [85] is another work (in single agent DRL) that also maintains a multi-time scale hierarchy where the slower network sets the goal, and faster network tries to achieve them.

Fingerprints adds the estimate of other agents' policies. For the practical implementation, good results were obtained by using the training iteration number and exploration rate as the fingerprint.

Lenient-DQN (LDQN) [63] extended the *leniency* concept to multiagent DRL, this is, introducing optimism in the value function update to foster cooperation [82]. However, similar to other works [62], the authors experienced problems with the ER buffer and arrived at a similar solution: adding information to the experience tuple, in their case, the leniency value. When sampling from the ER buffer, this value is used to determine a leniency condition; if the condition is not met then the sample is ignored.

In a similar vein, Hysteretic Deep Recurrent Q-Networks (HDRQNs) [64] were proposed for fostering cooperation among independent learners. The motivation is similar to LDQN, making a optimistic value update, however, their solution is different. Here, two learning rates are used (inspired by Hysteretic Q-learning [8]) and the ER buffer is modified into *concurrent experience replay trajectories*, which are composed of three dimensions: agent index, the episode, and the timestep. When training, the sampled traces have the same starting timesteps. Moreover, to improve on generalization over different tasks they make use of policy distillation [83] (see Section 4.1).

Weighted Double Deep Q-Network (WDDQN) [65] is based on having double estimators [84], i.e., two Q-networks, which intuitively balance between overestimation and underestimation. It also uses a *lenient* reward to be optimistic during initial phase of coordination and proposes a *scheduled* replay strategy in which samples closer to the terminal state are given higher priority since those are more rare in occurrence.

While previous approaches were mostly inspired by how MAL algorithms could be extended to

MDRL, other works take as base the results by single-agent DRL. One example is the For The Win (FTW) [66] agent which is based on the IMPALA architecture [43] (see Section 2.2). Their setting is a game where two opposing teams compete to capture each other's flags [86]. To deal with the MAL problem they propose two main additions: a *hierarchical two-level representation* with RNNs operating at different timescales, as depicted in Figure 5, and a *population based training [87]* where 30 agents were trained in parallel together with a stochastic matchmaking scheme that biases agents to be of similar *skills* [88]. An interesting result from this work is that the population-based training obtained better results than training via self-play, which was a standard concept in previous works [13, 89].

FTW is based on value-based methods, but others have considered policy gradient methods (see Section 2.1) for MDRL. For example, Lowe et al. [71] noted that using standard policy gradient methods on multiagent environments yields high variance and performs poorly. Therefore, to overcome this issue they propose the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [71] which is an extension of DDPG [41] with an actor-critic architecture where *the critic is augmented with other agents' actions*, while the actor only has local information (turning the method into a centralized training with decentralized execution). In this case, the ER buffer records experiences of *all* agents. In contrast to the previous algorithms, MADDPG was tested in both cooperative and competitive scenarios.

Another approach based on policy gradients is the Counterfactual Multi-Agent Policy Gradients (COMA) [70]. COMA focuses on the fully centralized setting and the *multiagent credit assignment problem* [90], i.e., how the agents should deduce their contributions when learning in a cooperative setting in the presence only of global rewards. Their proposal is to compute a *counterfactual baseline*, that is, marginalize out the action of the agent while keeping the rest of the other agents' actions fixed. Then, an advantage function can be computed comparing the current Q value to the counterfactual.

On the one hand, fully centralized approaches (e.g., COMA) do not suffer from non-stationarity but have constrained scalability. On the other hand, independent learning agents are better suited to scale but suffer from non-stationarity issues. There are some hybrid approaches that learn *a centralized but factored* Q value function [91]. Value Decomposition Networks (VDNs) [67] aim to decompose a team value function into an additive decomposition of the individual value functions. Similarly, QMIX [68] relies on the idea of factorizing, however, instead of sum, QMIX assumes a *mixing network* that combines the local values in a non-linear way, which can represent complex action-value functions.

3.6. Agents modeling agents

An important ability for agents to have is to reason about the behaviors of other agents by constructing models that make predictions about the modeled agents [11]. An early work for modeling agents while using deep neural networks was the Deep Reinforcement Opponent Network (DRON) [72]. The idea is to have two networks: one which evaluates Q values and a second one that *learns a representation of the opponent's policy*. Moreover, the authors proposed to have several expert networks to combine their predictions to get the estimated Q value, the idea being that each expert network captures one type of opponent strategy. DRON uses hand-crafted features to define the opponent network. In contrast, Deep Policy Inference Q-Network (DPIQN) and its recurrent version, DPIRQN [74] learn *policy features* directly from raw observations of the other agents. The way to learn these policy features is by means of auxiliary tasks [36] (see Section 4.1) that provide additional learning goals, in this case, the auxiliary task is to learn the opponents'

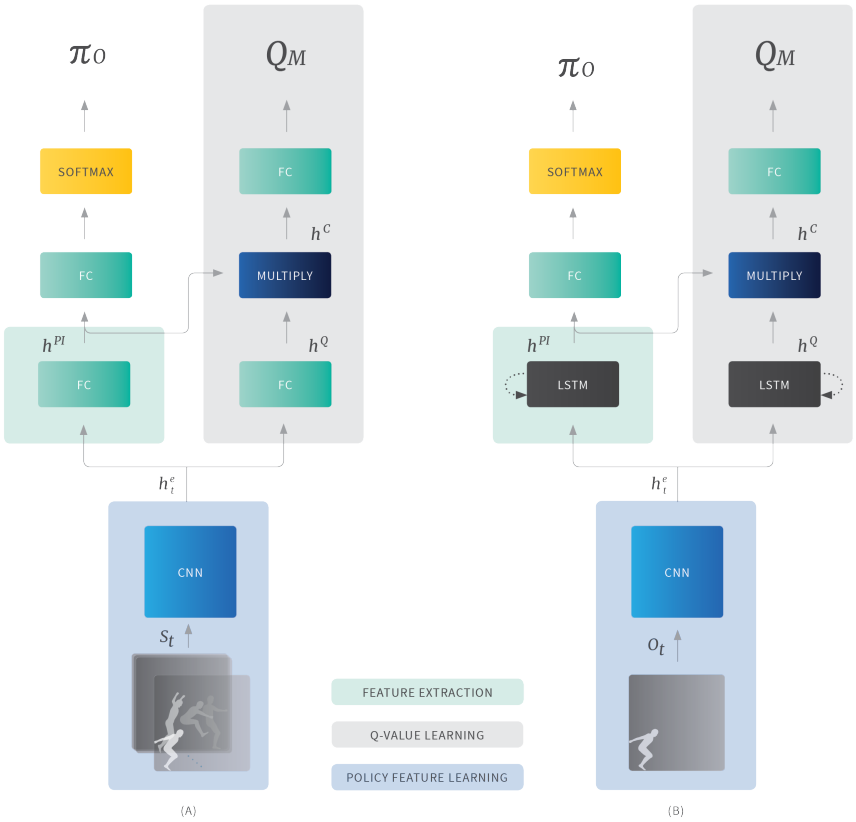


Figure 6: (a) Deep Policy Inference Q-Network: receives four frames as input. (b) Deep Policy Inference Recurrent Q-Network: receives one frame as input and has an LSTM layer instead of a fully connected layer (FC). Both approaches [74] condition the Q_M value outputs on the policy features, h^{PI}, which are also used to learn the opponent policy π_o.

policy. Then, the Q value function of the learning agent is conditioned on the policy features (see Figure 6), which aims to reduce the non-stationarity of the environment. The authors used an adaptive training procedure to adjust the *attention* (a weight on the loss function) to either emphasize learning the policy features (of the opponent) or the respective Q values of the agent. An advantage of these approaches is that modeling the agents can work for both opponents and teammates [74].

In many previous works an opponent model is learned from observations. Self Other Modeling (SOM) [73] proposed a different approach, this is, using *the agent's own policy as a means to predict the opponent's actions*. SOM aims to infer other agents' goals by using two networks, one used for computing the agents' policy and Q values, and a second one used to infer the opponent's policy. The idea is that these networks have the same input parameters but with a different values (the agent's or the opponent's). In contrast to previous approaches, SOM is not focused on learning the opponent strategy but rather on estimating the opponent's goal (hidden state).

There is a long-standing history of combining game theory and MAL [2, 6, 89]. From that context, some approaches were inspired by influential game theory approaches. Neural Fictitious Self-Play (NFSP) [75] builds on fictitious play [92] together with two deep networks to find *approximate Nash equilibria*, which is an efficient solution concept in certain types of games (e.g.,

two-player zero-sum games). One network learns an *approximate best response* to the historical behavior of other agents and the second one learns a model that averages over the agent's behavior. The agent behaves using a mixture of both networks. A second example that takes inspiration from behavioral game theory [93, 94] are Deep Cognitive Hierarchies (DCHs) [69]. Their goal is to find *approximate best responses* to the meta-strategy (obtained by empirical game theoretic analysis [95]) of other players. Their reasoning is that computing a standard best response can over-fit to a specific agent, causing it to fail to generalize. Therefore, they propose to add an *opponent/teammate regularization* by means of approximately best responding to a mixture of policies.

Previous approaches usually learned a model of the other agents as a way to predict their behavior. However, they do not explicitly *account for anticipated learning of the other agents*, which is the objective of Learning with Opponent-Learning Awareness (LOLA) [76]. LOLA optimizes the expected return *after the opponent updates its policy one step*. Therefore, a LOLA agent directly shapes the policy updates of other agents to maximize its own reward. One of LOLA's assumptions is having access to opponents' policy parameters.

Theory of Mind Network (ToMnet) [77] starts with a simple premise: when encountering a novel opponent, *the agent should already have a strong and rich prior about how the opponent should behave*. They propose an architecture composed of three networks: (i) a character network that learns from historical information, (ii) a mental state network that takes the character output and the recent trajectory, and (iii) the prediction network that takes the current state as well as the outputs of the other networks as its input. The output of the architecture is open for different problems but in general its goal is to predict the opponent's next action.

Deep Bayes-ToMoP (Bayesian Theory of Mind Policy) [78] is another algorithm that takes inspiration from theory of mind [96]. This algorithm is capable of best responding to changing stationary strategies of the opponent and to *higher-level reasoning strategies* (using theory of mind). It uses concepts from Bayesian policy reuse [97, 98] and efficient tracking and detection of opponent strategies [99, 100] together with deep networks.

4. Bridging MAL and MDRL

This section aims to provide directions to promote fruitful cooperations between sub-communities and reduce fractures in the MAL community. First, we present examples on how techniques from DRL and MAL can complement each other to solve problems in MDRL (see Section 4.1). Second, we outline *lessons learned* from the works analyzed in this survey (see Section 4.2). Third, we pose some open challenges and reflect on their relation with previous open questions in MAL [11] (see Section 4.3).

4.1. Examples of fruitful cooperation to solve MDRL problems

Here, we present 2 examples on how techniques from MAL can be used to solve problems in MDRL, and then 2 examples of DRL techniques extended naturally to MDRL.

Dealing with non-stationarity in independent learners. It is well known that using independent learners makes the environment non-stationary from the agent's point of view [4, 47]. This can be solved in different ways [10]. One example is *Hyper-Q* [101] which is a MAL approach that proposes to account for the (values of mixed) strategies of other agents and include that information in the state representation, which effectively turns the learning problem into a stationary one. Note that

in this way it is possible to even consider adaptive agents. Foerster et al. [59] make use of this insight to propose their *fingerprint* algorithm in an MDRL problem (see Section 3.5).

Multiagent credit assignment. In cooperative multiagent scenarios, it is common to use either *local rewards*, unique for each agent, or *global rewards*, which represent the entire group's performance [102]. However, local rewards are usually harder to obtain, therefore, it is common to rely only on the global ones. This raises the problem of *credit assignment*: how do a single agent's actions contribute to a system that involves the actions of many agents [23]. A solution that has proven successful in many scenarios is *difference rewards* [102, 103], which capture an agent's contribution to the system's global performance. COMA builds on this concept to propose an *advantage function* based on the contribution of the agent, which can be efficiently computed with deep neural networks [70] (see Section 3.5).

Multitask learning in MDRL. In the context of RL, multitask learning is an area that develops agents that can act in *several related tasks* rather than just in a single one [104]. Policy distillation [83] was proposed to extract DRL policies to train a much smaller NN and it also can be used to merge *several task-specific policies* into a single policy, i.e., for multitask learning. In the MDRL setting, HDRQNs [64] successfully adapted policy distillation to obtain a more general multitask multiagent network (see Section 3.5).

Auxiliary tasks for MDRL. Jaderberg et al. [36] introduced the auxiliary task concept with the insight that (single-agent) environments contain a variety of possible training signals (e.g., pixel changes, network activations). These can be treated as *pseudo-reward functions* and the agent can still use standard DRL to learn the optimal policy for those. One could think of extending these auxiliary tasks to modeling other agents' behaviors [57], this is one of the key ideas that DPIQN and DRPIQN [74] proposed in MDRL settings (see Section 3.6).

4.2. Lessons learned

We have exemplified how DRL and MAL can complement each other for MDRL settings. Now, we outline general *best practices* learned from the works analyzed throughout this paper.

- *Experience replay buffer in MDRL.* While some works removed the ER buffer in MDRL [59] it is an important component in many DRL and MDRL algorithms. However, using the standard buffer (i.e., keeping $\langle s, a, r, s' \rangle$) will probably fail. *Adding information in the experience tuple* that can help disambiguate the sample, is the solution adopted in many works, whether a value based method [62, 63, 64, 65] or a policy gradient method [71]. In this regard, it is an open question to consider new DRL ideas for the ER [105, 106, 107] and how those would fare in a MDRL setting.

- *Centralized learning with decentralized execution.* Many MAL works were either fully centralized or fully decentralized approaches. However, inspired by *decentralized partially observable Markov decison processes* (DEC-POMDPs) [108], in MDRL this new mixed paradigm has been commonly used [62, 63, 64, 68, 69, 70, 71]. Note that during learning *additional information can be used* (e.g., state, action, rewards) and during execution this information is removed.

- *Parameter sharing.* Another frequent component in many MDRL works is the idea of sharing parameters, i.e., training a single network in which agents share their weights. Note that, since agents could receive different observations (e.g., in partially observable scenarios), they can still behave differently [59, 60, 61, 62, 67, 68].

- *Recurrent networks.* LSTM networks [39, 109] are widely used recurrent neural networks that have memory capability for sequential data, in contrast to feed-forward or convolutional neural networks. In single-agent DRL, DRQN [37] initially proposed idea of using recurrent networks in single-agent partially observable environments. Then, Feudal Networks [85, 110] proposed *multiple LSTM networks with different time-scales*, i.e., the observation input schedule is different for each LSTM network, to create a temporal hierarchy so that it can better address the long-term credit assignment challenge for RL problems. Recently, the use of recurrent networks has been extended to MDRL [58, 59, 61, 64, 67, 68, 73, 74, 77, 111] for example, in FTW [66], depicted in Figure 5.

- *Ensemble policies.* Particularly when explicitly modeling opponents, models can over-fit to the behavior of other agents. In order to reduce this problem, a solution is to have a set of policies and learn from them or best respond to the mixture of them [69, 71, 72].

4.3. Open questions

Finally, here we present some open questions for MDRL.

- On the challenge of sparse and delayed rewards.

  Recent MAL competitions and environments (e.g., Pommerman [112], Capture the flag [86], MARLO [113], Starcraft II [114], and Dota 2 [16]) have complex scenarios where many actions are taken before a reward signal is available. This is already a challenge for RL [17, 26] — in MDRL this is even more problematic since the agents not only need to learn basic behaviors (like in DRL), but also to learn the strategic element (e.g., competitive/collaborative) embedded in the multiagent setting. To address this issue, recent MDRL approaches propose adding *dense* rewards at each step to allow the agents to learn basic motor skills and then decrease these *dense* rewards over time in favor of the environmental reward [58]. Recent works like OpenAI Five [16] uses hand-crafted intermediate rewards to accelerate the learning and FTW [66] lets the agents learn their internal rewards by a hierarchical two-tier optimization. In *single agent* domains, RUDDER [115] has been recently proposed for such delayed sparse reward problems. RUDDER generates a new MDP with *more intermediate rewards* whose optimal solution is still an optimal solution to the original MDP. This is achieved by using LSTM networks to redistribute the original sparse reward to earlier state-action pairs and automatically provide reward shaping. The extension of RUDDER to multiagent domains is an open question that might yield more efficient learning agents.

- On the role of self-play.

  Self-play is a cornerstone in MAL with impressive results [49, 116, 117, 118, 119]. While notable results had also been shown in MDRL [75, 89], recent works have also shown that *plain* self-play does not yield the best results. However, adding diversity, i.e., population-based or sampling-based methods, have shown good results [58, 66]. A drawback of these solutions is the additional computational requirements since they need either parallel training

(more CPU computation) or memory requirements. Then, it is still an open problem to work on *computationally efficient implementations* in MAL and MDRL (see [11, Section 5.5]).

- On the challenge of the combinatorial nature of MDRL.

  Monte Carlo tree search (MCTS) [120] has been the backbone of major breakthrough for AlphaGo [13] and AlphaGo Zero [14] that combined search and DRL. A recent work [121] has outlined how search and RL can be better combined for potentially new methods. However, for multiagent scenarios, there is an additional challenge of the exponential growth of all the agents' action spaces for centralized methods [122]. Amato et.al. [123] proposed to use factored value functions to exploit structure for multiagent settings. Search parallelization has also been employed for better scalability within multiagent scenarios [124, 125]. Given more scalable planners, there is room for research in combining these techniques in MDRL settings.

5. Conclusions

Deep reinforcement learning has shown recent success on many fronts [12, 13, 15] and a natural next step is to test multiagent scenarios. However, learning in multiagent environments is fundamentally more difficult due to non-stationarity, the increase of dimensionality, and the credit-assignment problem, among other factors [1, 5, 10, 49, 90, 126].

While previous MAL surveys have tried to contextualize most of the MAL literature within the framework of game theory [2], there is still an acknowledgement that some works lie outside that framework. Stone [127] takes this insight as a starting point to consider that *"there is no single correct multiagent learning algorithm — each problem must be considered individually."* Based on this reasoning, in this survey we have presented an overview on recent works in the emerging area of Multiagent Deep Reinforcement Learning (MDRL).

This work provides a comprehensive view of MDRL. First, we categorized recent works into four different topics: emergent behaviors, learning communication, learning cooperation, and agents modeling agents. Then, we exemplified how DRL and MAL can complement each other, we provided general lessons learned applicable to MDRL, and we reflected on open questions for MAL and MDRL.

While the number of works in DRL and MDRL are notable and represent important milestones for AI, at the same time we acknowledge there are also open questions in both (deep) single-agent learning [19, 25, 128, 129, 130, 131] and multiagent learning [10, 11, 132, 133, 134, 135, 136]. In this article, we have provided an outline of a recent active research area of MDRL, and at the same time our aim was to motivate future research to take advantage of the ample and existing literature to avoid having scattered sub-communities with little interaction.

References

[1] P. Stone, M. M. Veloso, Multiagent Systems - A Survey from a Machine Learning Perspective., Autonomous Robots 8 (2000) 345–383.

[2] Y. Shoham, R. Powers, T. Grenager, If multi-agent learning is the answer, what is the question?, Artificial Intelligence 171 (2007) 365–377.

[3] E. Alonso, M. D'inverno, D. Kudenko, M. Luck, J. Noble, Learning in multi-agent systems, Knowledge Engineering Review 16 (2002) 1–8.

[4] K. Tuyls, G. Weiss, Multiagent learning: Basics, challenges, and prospects, AI Magazine 33 (2012) 41–52.

[5] L. Busoniu, R. Babuska, B. De Schutter, A Comprehensive Survey of Multiagent Reinforcement Learning, IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 38 (2008) 156–172.
[6] A. Nowé, P. Vrancx, Y.-M. De Hauwere, Game theory and multi-agent reinforcement learning, in: Reinforcement Learning, Springer, 2012, pp. 441–470.
[7] L. Panait, S. Luke, Cooperative Multi-Agent Learning: The State of the Art, Autonomous Agents and Multi-Agent Systems 11 (2005).
[8] L. Matignon, G. J. Laurent, N. Le Fort-Piat, Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems, Knowledge Engineering Review 27 (2012) 1–31.
[9] D. Bloembergen, K. Tuyls, D. Hennes, M. Kaisers, Evolutionary Dynamics of Multi-Agent Learning: A Survey., Journal of Artificial Intelligence Research 53 (2015) 659–697.
[10] P. Hernandez-Leal, M. Kaisers, T. Baarslag, E. Munoz de Cote, A Survey of Learning in Multiagent Environments - Dealing with Non-Stationarity (2017). arXiv:1707.09183.
[11] S. V. Albrecht, P. Stone, Autonomous agents modelling other agents: A comprehensive survey and open problems, Artificial Intelligence 258 (2018) 66–95.
[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533.
[13] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[14] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of Go without human knowledge, Nature 550 (2017) 354.
[15] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, M. Bowling, DeepStack: Expert-level artificial intelligence in heads-up no-limit poker, Science 356 (2017) 508–513.
[16] Open AI Five, https://blog.openai.com/openai-five, 2018. [Online; accessed 7-September-2018].
[17] R. S. Sutton, A. G. Barto, Introduction to reinforcement learning, volume 135, MIT press Cambridge, 1998.
[18] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436.
[19] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, A Brief Survey of Deep Reinforcement Learning (2017). arXiv:1708.05866v2.
[20] B. M. Lake, T. D. Ullman, J. Tenenbaum, S. Gershman, Building machines that learn and think like people, Behavioral and Brain Sciences 40 (2016).
[21] A. Tamar, S. Levine, P. Abbeel, Y. Wu, G. Thomas, Value Iteration Networks., NIPS (2016) 2154–2162.
[22] G. Weiss (Ed.), Multiagent Systems, (Intelligent Robotics and Autonomous Agents series), 2nd ed., MIT Press, Cambridge, MA, USA, 2013.
[23] A. K. Agogino, K. Tumer, Unifying Temporal and Structural Credit Assignment Problems., in: Proceedings of 17th International Conference on Autonomous Agents and Multiagent Systems, 2004.
[24] E. Wei, S. Luke, Lenient Learning in Independent-Learner Stochastic Cooperative Games., Journal of Machine Learning Research (2016).
[25] A. Darwiche, Human-level intelligence or animal-like abilities?, Commun. ACM 61 (2018) 56–67.
[26] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, Journal of artificial intelligence research 4 (1996) 237–285.
[27] M. L. Puterman, Markov decision processes: Discrete stochastic dynamic programming, John Wiley & Sons, Inc., 1994.
[28] R. Bellman, A Markovian decision process, Journal of Mathematics and Mechanics 6 (1957) 679–684.
[29] J. Watkins, Learning from delayed rewards, Ph.D. thesis, King's College, Cambridge, UK, 1989.
[30] J. Tsitsiklis, Asynchronous stochastic approximation and Q-learning, Machine Learning 16 (1994) 185–202.
[31] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, Policy Gradient Methods for Reinforcement Learning with Function Approximation., in: Advances in Neural Information Processing Systems, 2000.
[32] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning 8 (1992) 229–256.
[33] V. R. Konda, J. Tsitsiklis, Actor-critic algorithms, in: Advances in Neural Information Processing Systems, 2000.
[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing Atari with Deep Reinforcement Learning (2013). arXiv:1312.5602v1.
[35] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, P. Moritz, Trust Region Policy Optimization., in: 31st

International Conference on Machine Learning, Lille, France, 2015.
[36] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, K. Kavukcuoglu, Reinforcement Learning with Unsupervised Auxiliary Tasks., in: International Conference on Learning Representations, 2017.
[37] M. Hausknecht, P. Stone, Deep Recurrent Q-Learning for Partially Observable MDPs, in: International Conference on Learning Representations, 2015.
[38] A. R. Cassandra, Exact and approximate algorithms for partially observable Markov decision processes, Ph.D. thesis, Computer Science Department, Brown University, 1998.
[39] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.
[40] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, 2016, pp. 1928–1937.
[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: International Conference on Learning Representations, 2016.
[42] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, Proceedings of the 32nd International Conference on Machine Learning (2015) 448–456.
[43] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al., IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures, in: International Conference on Machine Learning, 2018.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms (2017). arXiv:1707.06347.
[45] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, D. Silver, Emergence of Locomotion Behaviours in Rich Environments. (2017). arXiv:1707.02286v2.
[46] M. Tan, Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents, in: Machine Learning Proceedings 1993 Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, June 27–29, 1993, 1993, pp. 330–337.
[47] G. J. Laurent, L. Matignon, L. Fort-Piat, et al., The world of independent learners is not markovian, International Journal of Knowledge-based and Intelligent Engineering Systems 15 (2011) 55–64.
[48] M. L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Proceedings of the 11th International Conference on Machine Learning, New Brunswick, NJ, USA, 1994, pp. 157–163.
[49] M. Bowling, M. Veloso, Multiagent learning using a variable learning rate, Artificial Intelligence 136 (2002) 215–250.
[50] D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, T. Graepel, The mechanics of n-player differentiable games (2018). arXiv:1802.05642.
[51] N. Fulda, D. Ventura, Predicting and Preventing Coordination Problems in Cooperative Q-learning Systems, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, Hyderabad, India, 2007, pp. 780–785.
[52] Multiagent Learning, Foundations and Recent Trends, https://www.cs.utexas.edu/~larg/ijcai17_tutorial/multiagent_learning.pdf, 2017. [Online; accessed 7-September-2018].
[53] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, R. Vicente, Multiagent cooperation and competition with deep reinforcement learning, PLOS ONE 12 (2017) e0172395.
[54] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, Multi-agent Reinforcement Learning in Sequential Social Dilemmas, in: Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems, Sao Paulo, 2017.
[55] M. Raghu, A. Irpan, J. Andreas, R. Kleinberg, Q. Le, J. Kleinberg, Can Deep Reinforcement Learning solve Erdos-Selfridge-Spencer Games?, in: Proceedings of the 35th International Conference on Machine Learning, 2018.
[56] A. Lazaridou, A. Peysakhovich, M. Baroni, Multi-Agent Cooperation and the Emergence of (Natural) Language, in: International Conference on Learning Representations, 2017.
[57] I. Mordatch, P. Abbeel, Emergence of grounded compositional language in multi-agent populations (2017). arXiv:1703.04908.
[58] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, I. Mordatch, Emergent Complexity via Multi-Agent Competition., in: International Conference on Machine Learning, 2018.
[59] J. N. Foerster, Y. M. Assael, N. De Freitas, S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in: Advances in Neural Information Processing Systems, 2016, pp. 2145–2153.
[60] S. Sukhbaatar, A. Szlam, R. Fergus, Learning Multiagent Communication with Backpropagation, in: Advances

in Neural Information Processing Systems, 2016, pp. 2244–2252.
[61] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, J. Wang, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games. (2017). arXiv:1703.10069.
[62] J. N. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, S. Whiteson, Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning., in: International Conference on Machine Learning, 2017.
[63] G. Palmer, K. Tuyls, D. Bloembergen, R. Savani, Lenient Multi-Agent Deep Reinforcement Learning., in: International Conference on Autonomous Agents and Multiagent Systems, 2018.
[64] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, J. Vian, Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability, in: Proceedings of the 34th International Conference on Machine Learning, Sydney, 2017.
[65] Y. Zheng, J. Hao, Z. Zhang, Weighted double deep multiagent reinforcement learning in stochastic cooperative environments (2018). arXiv:1802.08534.
[66] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, T. Graepel, Human-level performance in first-person multiplayer games with population-based deep reinforcement learning (2018).
[67] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, T. Graepel, Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward., in: Proceedings of 17th International Conference on Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 2018.
[68] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, S. Whiteson, QMIX - Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning., in: International Conference on Machine Learning, 2018.
[69] M. Lanctot, V. F. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, T. Graepel, A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning., in: Advances in Neural Information Processing Systems, 2017.
[70] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual Multi-Agent Policy Gradients., in: 32nd AAAI Conference on Artificial Intelligence, 2017.
[71] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments., in: Advances in Neural Information Processing Systems, 2017, pp. 6379–6390.
[72] H. He, J. Boyd-Graber, K. Kwok, H. Daume, Opponent modeling in deep reinforcement learning, in: 33rd International Conference on Machine Learning, 2016, pp. 2675–2684.
[73] R. Raileanu, E. Denton, A. Szlam, R. Fergus, Modeling Others using Oneself in Multi-Agent Reinforcement Learning., in: International Conference on Machine Learning, 2018.
[74] Z.-W. Hong, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, C.-Y. Lee, A Deep Policy Inference Q-Network for Multi-Agent Systems, in: International Conference on Autonomous Agents and Multiagent Systems, 2018.
[75] J. Heinrich, D. Silver, Deep Reinforcement Learning from Self-Play in Imperfect-Information Games (2016). arXiv:1603.01121.
[76] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, I. Mordatch, Learning with Opponent-Learning Awareness., in: Proceedings of 17th International Conference on Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 2018.
[77] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. M. A. Eslami, M. Botvinick, Machine Theory of Mind., in: International Conference on Machine Learning, Stockholm, Sweden, 2018.
[78] T. Yang, Z. Meng, J. Hao, C. Zhang, Y. Zheng, Bayes-ToMoP: A Fast Detection and Best Reponse Algorithm Towards Sophisticated Opponents (2018). arXiv:1809.04240.
[79] E. Todorov, T. Erez, Y. Tassa, MuJoCo - A physics engine for model-based control., Intelligent Robots and Systems (2012).
[80] D. Fudenberg, J. Tirole, Game Theory, The MIT Press, 1991.
[81] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing 45 (1997) 2673–2681.
[82] D. Bloembergen, M. Kaisers, K. Tuyls, Lenient frequency adjusted Q-learning, in: Proceedings of the 22nd Belgian/Netherlands Artificial Intelligence Conference, 2010.
[83] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, R. Hadsell, Policy Distillation, in: International Confernece on Learning Representations, 2016.

[84] H. Van Hasselt, A. Guez, D. Silver, Deep Reinforcement Learning with Double Q-Learning, in: AAAI, Phoenix, AZ, 2016.
[85] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, K. Kavukcuoglu, FeUdal Networks for Hierarchical Reinforcement Learning., International Conference On Machine Learning (2017).
[86] Capture the Flag: the emergence of complex cooperative agents, `https://deepmind.com/blog/capture-the-flag/`, 2018. [Online; accessed 7-September-2018].
[87] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al., Population based training of neural networks (2017). `arXiv:1711.09846`.
[88] R. Herbrich, T. Minka, T. Graepel, TrueSkill: a Bayesian skill rating system, in: Advances in neural information processing systems, 2007, pp. 569–576.
[89] M. Bowling, N. Burch, M. Johanson, O. Tammelin, Heads-up limit hold'em poker is solved, Science 347 (2015) 145–149.
[90] K. Tumer, A. Agogino, Distributed agent-based air traffic flow management, in: Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems, Honolulu, Hawaii, 2007.
[91] F. A. Oliehoek, Interactive Learning and Decision Making - Foundations, Insights & Challenges., International Joint Conference on Artificial Intelligence (2018).
[92] G. W. Brown, Iterative solution of games by fictitious play, Activity analysis of production and allocation 13 (1951) 374–376.
[93] C. F. Camerer, T.-H. Ho, J.-K. Chong, A cognitive hierarchy model of games, The Quarterly Journal of Economics 119 (2004) 861.
[94] M. Costa Gomes, V. P. Crawford, B. Broseta, Cognition and Behavior in Normal–Form Games: An Experimental Study, Econometrica 69 (2001) 1193–1235.
[95] W. E. Walsh, R. Das, G. Tesauro, J. O. Kephart, Analyzing complex strategic interactions in multi-agent systems, AAAI-02 Workshop on Game-Theoretic and Decision-Theoretic Agents (2002) 109–118.
[96] H. de Weerd, R. Verbrugge, B. Verheij, How much does it help to know what she knows you know? An agent-based simulation study, Artificial Intelligence 199-200 (2013) 67–92.
[97] B. Rosman, M. Hawasly, S. Ramamoorthy, Bayesian Policy Reuse, Machine Learning 104 (2016) 99–127.
[98] P. Hernandez-Leal, M. E. Taylor, B. Rosman, L. E. Sucar, E. Munoz de Cote, Identifying and Tracking Switching, Non-stationary Opponents: a Bayesian Approach, in: Multiagent Interaction without Prior Coordination Workshop at AAAI, Phoenix, AZ, USA, 2016.
[99] P. Hernandez-Leal, Y. Zhan, M. E. Taylor, L. E. Sucar, E. Munoz de Cote, Efficiently detecting switches against non-stationary opponents, Autonomous Agents and Multi-Agent Systems 31 (2017) 767–789.
[100] P. Hernandez-Leal, M. Kaisers, Towards a Fast Detection of Opponents in Repeated Stochastic Games, in: G. Sukthankar, J. A. Rodriguez-Aguilar (Eds.), Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, Sao Paulo, Brazil, May 8-12, 2017, Revised Selected Papers, 2017, pp. 239–257.
[101] G. Tesauro, Extending Q-learning to general adaptive multi-agent systems, in: Advances in Neural Information Processing Systems, Vancouver, Canada, 2003, pp. 871–878.
[102] A. K. Agogino, K. Tumer, Analyzing and visualizing multiagent rewards in dynamic and stochastic domains., Autonomous Agents and Multi-Agent Systems (2008).
[103] S. Devlin, L. M. Yliniemi, D. Kudenko, K. Tumer, Potential-based difference rewards for multiagent reinforcement learning., in: 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014, Paris, France, 2014.
[104] M. E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: A survey, The Journal of Machine Learning Research 10 (2009) 1633–1685.
[105] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba, Hindsight experience replay, in: Advances in Neural Information Processing Systems, 2017.
[106] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized Experience Replay, in: International Confernece on Learning Representations, 2016.
[107] Z. C. Lipton, K. Azizzadenesheli, A. Kumar, L. Li, J. Gao, L. Deng, Combating Reinforcement Learning's Sisyphean Curse with Intrinsic Fear (2018). `arXiv:1611.01211v8`.
[108] F. A. Oliehoek, C. Amato, et al., A concise introduction to decentralized POMDPs, Springer, 2016.
[109] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, J. Schmidhuber, LSTM: A Search Space Odyssey, IEEE Transactions on Neural Networks and Learning Systems 28 (2017) 2222–2232.
[110] P. Dayan, G. E. Hinton, Feudal reinforcement learning, in: Advances in neural information processing systems, 1993, pp. 271–278.
[111] Y. Yang, J. Hao, M. Sun, Z. Wang, C. Fan, G. Strbac, Recurrent Deep Multiagent Q-Learning for Autonomous

Brokers in Smart Grid, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 2018.
[112] C. Resnick, W. Eldridge, D. Ha, D. Britz, J. Foerster, J. Togelius, K. Cho, J. Bruna, Pommerman: A Multi-Agent Playground (2018). arXiv:1809.07124.
[113] Multi-Agent Reinforcement Learning in Minecraft, https://www.crowdai.org/challenges/marlo-2018, 2018. [Online; accessed 7-September-2018].
[114] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, R. Tsing, StarCraft II: A New Challenge for Reinforcement Learning (2017). arXiv:1708.04782v1.
[115] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, S. Hochreiter, Rudder: Return decomposition for delayed rewards (2018). arXiv:1806.07857.
[116] J. Hu, M. P. Wellman, Nash Q-learning for general-sum stochastic games, Journal of Machine Learning Research 4 (2003) 1039–1069.
[117] M. Bowling, Convergence and no-regret in multiagent learning, in: Advances in Neural Information Processing Systems, Vancouver, Canada, 2004, pp. 209–216.
[118] V. Conitzer, T. Sandholm, AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents, Machine Learning 67 (2006) 23–43.
[119] A. Greenwald, K. Hall, Correlated Q-learning, in: Proceedings of 17th International Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA, 2003, pp. 242–249.
[120] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of Monte Carlo tree search methods, IEEE Transactions on Computational Intelligence and AI in games 4 (2012) 1–43.
[121] T. Vodopivec, S. Samothrakis, B. Ster, On Monte Carlo tree search and reinforcement learning, Journal of Artificial Intelligence Research 60 (2017) 881–936.
[122] B. Kartal, J. Godoy, I. Karamouzas, S. J. Guy, Stochastic tree search with useful cycles for patrolling problems, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, 2015, pp. 1289–1294.
[123] C. Amato, F. A. Oliehoek, Scalable Planning and Learning for Multiagent POMDPs, in: AAAI, 2015, pp. 1995–2002.
[124] B. Kartal, E. Nunes, J. Godoy, M. Gini, Monte Carlo tree search with branch and bound for multi-robot task allocation, in: The IJCAI-16 Workshop on Autonomous Mobile Service Robots, 2016.
[125] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, R. Fitch, Dec-MCTS: Decentralized planning for multi-robot active perception, The International Journal of Robotics Research (2018) 0278364918755924.
[126] E. Wei, D. Wicke, D. Freelan, S. Luke, Multiagent Soft Q-Learning (2018). arXiv:1804.09817.
[127] P. Stone, Multiagent learning is not the answer. It is the question, Artificial Intelligence 171 (2007) 402–405.
[128] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep Reinforcement Learning That Matters., in: 32nd AAAI Conference on Artificial Intelligence, 2018.
[129] P. Nagarajan, G. Warnell, P. Stone, Deterministic implementations for reproducibility in deep reinforcement learning (2018). arXiv:1809.05676.
[130] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, M. Bowling, Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents, Journal of Artificial Intelligence Research 61 (2018) 523–562.
[131] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, D. Perez-Liebana, Deep Reinforcement Learning for General Video Game AI (2018). arXiv:1806.02448.
[132] P. A. Ortega, S. Legg, Modeling friends and foes (2018). arXiv:1807.00196.
[133] G. Bono, J. S. Dibangoye, L. Matignon, F. Pereyron, O. Simonin, Cooperative multi-agent policy gradient, in: European Conference on Machine Learning, 2018.
[134] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, J. Wang, Mean field multi-agent reinforcement learning, in: Proceedings of the 35th International Conference on Machine Learning, Stockholm Sweden, 2018.
[135] A. Grover, M. Al-Shedivat, J. K. Gupta, Y. Burda, H. Edwards, Learning Policy Representations in Multiagent Systems., in: International Conference on Machine Learning, 2018.
[136] C. K. Ling, F. Fang, J. Z. Kolter, What game are we playing? end-to-end learning in normal and extensive form games, in: Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018.