Autonomous Agents and Multi-Agent Systems (2019) 33:750–797
https://doi.org/10.1007/s10458-019-09421-1

# A survey and critique of multiagent deep reinforcement learning

**Pablo Hernandez-Leal**[1] · **Bilal Kartal**[1] · **Matthew E. Taylor**[1]

Published online: 16 October 2019


**Abstract**
Deep reinforcement learning (RL) 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 learning (MAL) scenarios. Initial results report successes in complex multiagent domains, although there are several challenges to be addressed. The primary goal of this article is to provide a clear overview of current multiagent deep reinforcement learning (MDRL) literature. Additionally, we complement the overview with a broader analysis: (i) we revisit previous key components, originally presented in MAL and RL, and highlight how they have been adapted to multiagent deep reinforcement learning settings. (ii) We provide general guidelines to new practitioners in the area: describing lessons learned from MDRL works, pointing to recent benchmarks, and outlining open avenues of research. (iii) We take a more critical tone raising practical challenges of MDRL (e.g., implementation and computational demands). We expect this article will help unify and motivate future research to take advantage of the abundant literature that exists (e.g., RL 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 [305] laid the groundwork for defining the area of multiagent systems (MAS) and its open problems in the context of AI. About 10 years ago, Shoham et al. [289] noted that the literature on multiagent learning (MAL) was

✉ Pablo Hernandez-Leal
pablo.hernandez@borealisai.com

Bilal Kartal
bilal.kartal@borealisai.com

Matthew E. Taylor
matthew.taylor@borealisai.com

[1] Borealis AI, Edmonton, Canada

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 [7,55,333], to addressing specific subareas: game theory and MAL [233,289], cooperative scenarios [213,248], and evolutionary dynamics of MAL [38]. In just the last couple of years, three surveys related to MAL have been published: learning in non-stationary environments [141], agents modeling agents [6], and transfer learning in multiagent RL [290].

The research interest in MAL has been accompanied by successes in artificial intelligence, first, in single-agent video games [221]; more recently, in two-player games, for example, playing Go [291,293], poker [50,224], and games of two competing teams, e.g., DOTA 2 [235] and StarCraft II [339].

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) [315] and deep learning [184,281].

RL is an area of machine learning where an agent learns by interacting (i.e., taking actions) within a dynamic environment. However, one of the main challenges to RL, and traditional machine learning in general, is the need for manually designing quality features on which to learn. Deep learning enables efficient representation learning, thus allowing the automatic discovery of features [184,281]. In recent years, deep learning has had successes in different areas such as computer vision and natural language processing [184,281]. One of the key aspects of deep learning is the use of *neural networks* (NNs) that can find compact representations in high-dimensional data [13].

In deep reinforcement learning (DRL) [13,101] deep neural networks are trained to approximate the optimal policy and/or the value function. In this way the deep NN, serving as function approximator, enables powerful generalization. One of the key advantages of DRL is that it enables RL to scale to problems with high-dimensional state and action spaces. However, most existing successful DRL applications so far have been on visual domains (e.g., Atari games), and there is still a lot of work to be done for more realistic applications [359,364] with complex dynamics, which are not necessarily vision-based.

DRL has been regarded as an important component in constructing general AI systems [179] and has been successfully integrated with other techniques, e.g., search [291], planning [320], and more recently with multiagent systems, with an emerging area of *multiagent deep reinforcement learning (MDRL)* [232,251].[1]

Learning in multiagent settings is fundamentally more difficult than the single-agent case due to the presence of multiagent pathologies, e.g., the moving target problem (non-stationarity) [55,141,289], curse of dimensionality [55,289], multiagent credit assignment [2,355], global exploration [213], and relative overgeneralization [105,247,347]. Despite this complexity, top AI conferences like AAAI, ICML, ICLR, IJCAI and NeurIPS, 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 article contributes to the state of the art with a brief survey of the current works in MDRL in an effort to complement existing surveys on multiagent learning [56,141], cooperative learning [213,248], agents modeling agents [6], knowledge reuse in multiagent RL [290], and (single-agent) deep reinforcement learning [13,191].

[1] We have noted inconsistency in abbreviations such as: D-MARL, MADRL, deep-multiagent RL and MA-DRL.

First, we provide a short review of key algorithms in RL such as Q-learning and REINFORCE (see Sect. 2.1). Second, we review DRL highlighting the challenges in this setting and reviewing recent works (see Sect. 2.2). Third, we present the multiagent setting and give an overview of key challenges and results (see Sect. 3.1). Then, we present the identified four categories to group recent MDRL works (see Fig. 1):

- Analysis of emergent behaviors: evaluate single-agent DRL algorithms in multiagent scenarios (e.g., Atari games, social dilemmas, 3D competitive games).
- Learning communication: agents learn communication protocols to solve cooperative tasks.
- Learning cooperation: agents learn to cooperate using only actions and (local) observations.
- Agents modeling agents: agents reason about others to fulfill a task (e.g., best response learners).

For each category we provide a description as well as outline the recent works (see Sect. 3.2 and Tables 1, 2, 3, 4). Then, we take a step back and reflect on how these new works relate to the existing literature. In that context, first, we present examples on how methods and algorithms originally introduced in RL and MAL were successfully been scaled to MDRL (see Sect. 4.1). Second, we provide some pointers for new practitioners in the area by describing general *lessons learned* from the existing MDRL works (see Sect. 4.2) and point to recent multiagent benchmarks (see Sect. 4.3). Third, we take a more critical view and describe practical challenges in MDRL, such as reproducibility, hyperparameter tunning, and computational demands (see Sect. 4.4). Then, we outline some open research questions (see Sect. 4.5). Lastly, we present our conclusions from this work (see Sect. 5).

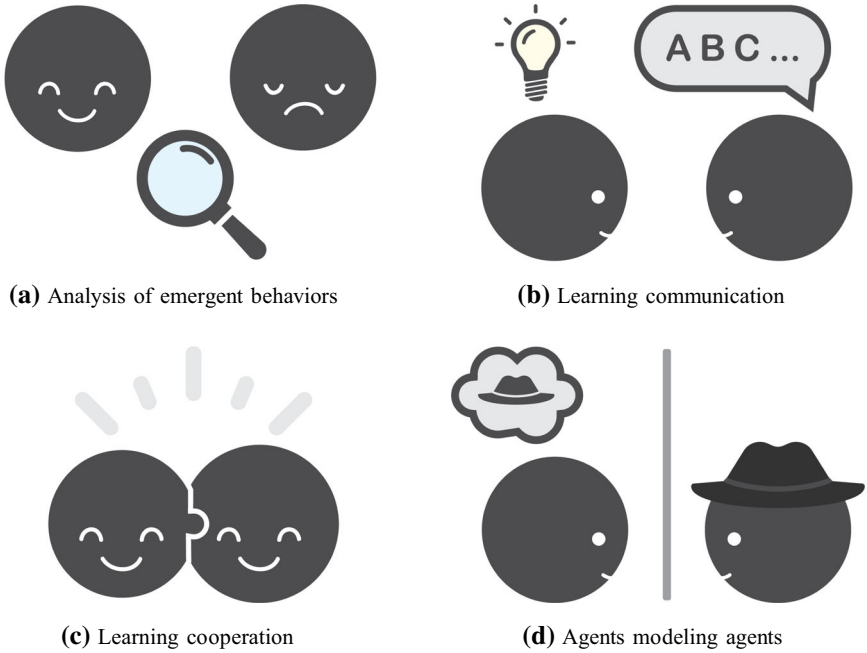
Our goal is to outline a recent and active area (i.e., MDRL), as well as to motivate future research to take advantage of the ample and existing literature in multiagent learning. We aim to enable researchers with experience in either DRL or MAL to gain a common understanding about recent works, and open problems in MDRL, and to avoid having scattered sub-communities with little interaction [6,81,141,289].

## 2 Single-agent learning

This section presents the formalism of reinforcement learning and its main components before outlining *deep* reinforcement learning along with its particular challenges and recent algorithms. For a more detailed description we refer the reader to excellent books and surveys on the area [13,101,164,315,353].

### 2.1 Reinforcement learning

RL formalizes the interaction of an agent with an environment using a Markov decision process (MDP) [261]. An MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$ where $\mathcal{S}$ represents a finite set of states. $\mathcal{A}$ represents a finite set of actions. The transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ determines the probability of a transition from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ given any possible action $a \in \mathcal{A}$. The reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ 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.




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

**(c)** Learning cooperation **(d)** Agents modeling agents

**Fig. 1** Categories of different MDRL works. **a** Analysis of emergent behaviors: evaluate single-agent 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 Sects. 3.3–3.6 and Tables 1, 2, 3 and 4

MDPs are adequate models to obtain optimal decisions in *single* agent fully observable environments.[2] Solving an MDP will yield a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which is a mapping from states to actions. An optimal policy $\pi^*$ 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 [33], which 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 [346]. 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:

[2] A Partially Observable Markov Decision Process (POMDP) [14,63] explicitly models environments where the agent no longer sees the true system state and instead receives an *observation* (generated from the underlying system state).

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha[(r + \gamma \max_{a'} \hat{Q}(s',a')) - \hat{Q}(s,a)] \quad (1)$$

with the learning rate $\alpha \in [0,1]$. Q-learning is proven to converge to $Q^*$ if state and action spaces are discrete and finite, the sum of the learning rates goes to infinity (so that each state-action pair is visited *infinitely* often) and that the sum of the squares of the learning rates is finite (which is required to show that the convergence is with probability one) [94,154,168,318,319,329,346]. The convergence of single-step on-policy RL algorithms, i.e, SARSA ($\lambda = 0$), for both decaying exploration (greedy in the limit with infinite exploration) and persistent exploration (selecting actions probabilistically according to the ranks of the Q values) was demonstrated by Singh et al. [294]. Furthermore, Van Seijen [337] has proven convergence for Expected SARSA (see Sect. 3.1 for convergence results in multiagent domains).

*REINFORCE (Monte Carlo policy gradient)* In contrast to value-based methods, which do not try to optimize directly over a policy space [175], policy gradient methods can learn parameterized policies without using intermediate value estimates.

Policy parameters are learned by following the gradient of some performance measure with gradient descent [316]. For example, REINFORCE [354] uses estimated return by Monte Carlo (MC) methods with full episode trajectories to learn policy parameters $\theta$, with $\pi(a;s,\theta) \approx \pi(a;s)$, as follows

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t;S_t,\theta_t)}{\pi(A_t;S_t,\theta_t)} \quad (2)$$

where $G_t$ represents the return, $\alpha$ is the learning rate, and $A_t \sim \pi$. A main limitation is that policy gradient methods can have high variance [175].

The policy gradient update can be generalized to include a comparison to an arbitrary *baseline* of the state [354]. The baseline, $b(s)$, can be any function, as long as it does not vary with the action; the baseline leaves the expected value of the update unchanged, but it can have an effect on its variance [315]. A natural choice for the baseline is a learned state-value function, this reduces the variance, and it is bias-free if learned by MC.[3] Moreover, when using the state-value function for bootstrapping (updating the value estimate for a state from the estimated values of subsequent states) it assigns credit (reducing the variance but introducing bias), i.e., criticizes the policy's action selections. Thus, in actor-critic methods [175], the actor represents the policy, i.e., action-selection mechanism, whereas a critic is used for the value function learning. In the case when the critic learns a state-action function ($Q$ function) and a state value function ($V$ function), an *advantage function* can be computed by subtracting state values from the state-action values [283,315]. The advantage function indicates the relative quality of an action compared to other available actions computed from the baseline, i.e., state value function. An example of an actor-critic algorithm is Deterministic Policy Gradient (DPG) [292]. In DPG [292] the critic follows the standard Q-learning and the actor is updated following the gradient of the policy's performance [128], DPG was later extended to DRL (see Sect. 2.2) and MDRL (see Sect. 3.5). For multiagent learning settings the variance is further increased as all the agents' rewards depend on the rest of the agents, and it is formally shown that as the number of agents increase, the probability of taking a correct gradient direction decreases exponentially [206]. Recent MDRL works addressed this high variance issue, e.g., COMA [97] and MADDPG [206] (see Sect. 3.5).

[3] Action-dependant baselines had been proposed [117,202], however, a recent study by Tucker et al. [331] found that in many works the reason of good performance was because of bugs or errors in the code, rather than the proposed method itself.

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 or values for continuous control [192], and whose weights are the policy parameters.

### 2.2 Deep reinforcement learning

While tabular RL methods such as Q-learning are successful in domains that do not suffer from the curse of dimensionality, there are many limitations: learning in large state spaces can be prohibitively slow, methods do not generalize (across the state space), and state representations need to be hand-specified [315]. Function approximators tried to address those limitations, using for example, decision trees [262], tile coding [314], radial basis functions [177], and locally weighted regression [46] to approximate the value function.

Similarly, these challenges can be addressed by using deep learning, i.e., neural networks [46,262] as function approximators. For example, $Q(s, a; \theta)$ can be used to approximate the state-action values with $\theta$ representing the neural network weights. This has two advantages, 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 [184,281].

However, extending deep learning to RL problems comes with additional challenges including non-i.i.d. (not independently and identically distributed) data. Many supervised learning methods assume that training data is from an i.i.d. stationary distribution [36, 269,281]. However, in RL, training data consists of highly correlated sequential agent-environment interactions, which violates the *independence* condition. Moreover, RL training data distribution is non-stationary as the agent actively learns while exploring different parts of the state space, violating the condition of sampled data being *identically distributed* [220].

In practice, using function approximators in RL requires making crucial representational decisions and poor design choices can result in estimates that diverge from the optimal value function [1,21,46,112,334,351]. In particular, function approximation, bootstrapping, and off-policy learning are considered the three main properties that when combined, can make the learning to diverge and are known as *the deadly triad* [315,334]. Recently, some works have shown that non-linear (i.e., deep) function approximators poorly estimate the value function [104,151,331] and another work found problems with Q-learning using function approximation (over/under-estimation, instability and even divergence) due to the *delusional* bias: "delusional bias occurs whenever a backed-up value estimate is derived from action choices that are not realizable in the underlying policy class"[207]. Additionally, convergence results for reinforcement learning using function approximation are still scarce [21,92,207,217,330]; in general, stronger convergence guarantees are available for policy-gradient methods [316] than for value-based methods [315].

Below we mention how the existing DRL methods aim to address these challenges when briefly reviewing value-based methods, such as DQN [221]; policy gradient methods, like Proximal Policy Optimization (PPO) [283]; and actor-critic methods like Asynchronous Advantage Actor-Critic (A3C) [158]. We refer the reader to recent surveys on single-agent DRL [13,101,191] for a more detailed discussion of the literature.

*Value-based methods* The major breakthrough work combining deep learning with Q-learning was the Deep Q-Network (DQN) [221]. DQN uses a deep neural network for function approx-

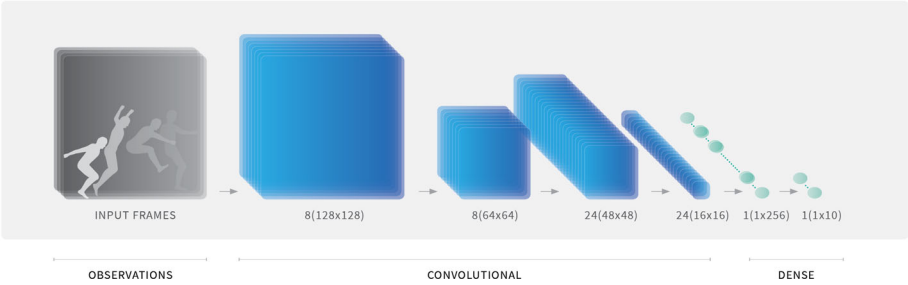



**Fig. 2** Deep Q-Network (DQN) [221]: Inputs are four stacked frames; the network is composed of several layers: *Convolutional* layers employ filters to learn features from high-dimensional data with a much smaller number of neurons and *Dense* layers are fully-connected layers. The last layer represents the actions the agent can take (in this case, 10 possible actions). Deep Recurrent Q-Network (DRQN) [131], which extends DQN to partially observable domains [63], is identical to this setup except the penultimate layer ($1 \times 256$ Dense layer) is replaced with a recurrent LSTM layer [147]

imation [268][4] (see Fig. 2) and maintains an *experience replay* (ER) buffer [193,194] to store interactions $\langle s, a, r, s' \rangle$. DQN keeps an additional copy of neural network parameters, $\theta^-$, for the target network in addition to the $\theta$ parameters to stabilize the learning, i.e., to alleviate the non-stationary data distribution.[5] For each training iteration $i$, DQN minimizes the mean-squared error (MSE) between the Q-network and its target network using the loss function:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2] \quad (3)$$

where target network parameters $\theta^-$ are set to Q-network parameters $\theta$ periodically and mini-batches of $\langle s, a, r, s' \rangle$ tuples are sampled from the ER buffer, as depicted in Fig. 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 higher memory requirements and computation per real interaction [219]. The ER buffer is mainly used for off-policy RL methods as it can cause a mismatch between buffer content from earlier policy and from the current policy for on-policy methods [219]. Extending the ER buffer for the multiagent case is not trivial, see Sects. 3.5, 4.1 and 4.2. Recent works were designed to reduce the problem of catastrophic forgetting (this occurs when the trained neural network performs poorly on previously learned tasks due to a non-stationary training distribution [111,214]) and the ER buffer, in DRL [153] and MDRL [246].

DQN has been extended in many ways, for example, by using double estimators [130] to reduce the overestimation bias with Double DQN [336] (see Sect. 4.1) and by decomposing the Q-function with a *dueling*-DQN architecture [345], where two streams are learned, one estimates state values and another one advantages, those are combined in the final layer to form $Q$ values (this method improved over Double DQN).

In practice, DQN is trained using an input of four stacked frames (last four frames the agent has encountered). If a game requires a memory of more than four frames it will appear non-Markovian to DQN because the future game states (and rewards) do not depend only on the input (four frames) but rather on the history [132]. Thus, DQN's performance declines

[4] Before DQN, many approaches used neural networks for representing the Q-value function [74], such as Neural Fitted Q-learning [268] and NEAT+Q [351].

[5] Double Q-learning [130] originally proposed keeping two $Q$ functions (estimators) to reduce the overestimation bias in RL, while still keeping the convergence guarantees, later it was extended to DRL in Double DQN [336] (see Sect. 4.1).


**Fig. 3** Representation of a DQN agent that uses an experience replay buffer [193,194] to keep $\langle s, a, r, s' \rangle$ tuples for minibatch updates. The Q-values are parameterized with a NN and a policy is obtained by selecting (greedily) over those at every timestep

when given incomplete state observations (e.g., one input frame) since DQN assumes full state observability.

Real-world tasks often feature incomplete and noisy state information resulting from *partial observability* (see Sect. 2.1). Deep Recurrent Q-Networks (DRQN) [131] proposed using *recurrent neural networks*, in particular, Long Short-Term Memory (LSTMs) cells [147] in DQN, for this setting. Consider the architecture in Fig. 2 with the first dense layer after convolution replaced by a layer of LSTM cells. With this addition, DRQN has memory capacity so that it can even work with only one input frame rather than a stacked input of consecutive frames. This idea has been extended to MDRL, see Fig. 6 and Sect. 4.2. There are also other approaches to deal with partial observability such as finite state controllers [218] (where action selection is performed according to the complete observation history) and using an initiation set of options conditioned on the previously employed option [302].

*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) [192] is a model-free off-policy actor-critic algorithm for such domains, based on the DPG algorithm [292] (see Sect. 2.1). Additionally, it proposes a new method for updating the networks, i.e., the target network parameters slowly change (this could also be applicable to DQN), in contrast to the hard reset (direct weight copy) used 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 [152] to ensure generalization across many different tasks without performing manual normalizations. However, note that other works have shown batch normalization can cause divergence in DRL [274,335].

Asynchronous Advantage Actor-Critic (A3C) [219] 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 Fig. 4). There is also the Advantage Actor-Critic (A2C) method [234] that combines all the gradients from all the workers to update the global NN *synchronously*. The loss function for A3C is composed of two terms: policy loss (actor), $\mathcal{L}_\pi$, and value loss (critic), $\mathcal{L}_v$. A3C parameters are updated using the *advantage* function $A(s_t, a_t; \theta_v) = Q(s, a) - V(s)$, commonly used to reduce variance (see Sect. 2.1). An entropy loss for the policy, $H(\pi)$, is also commonly added, which helps to improve exploration by discouraging premature con-

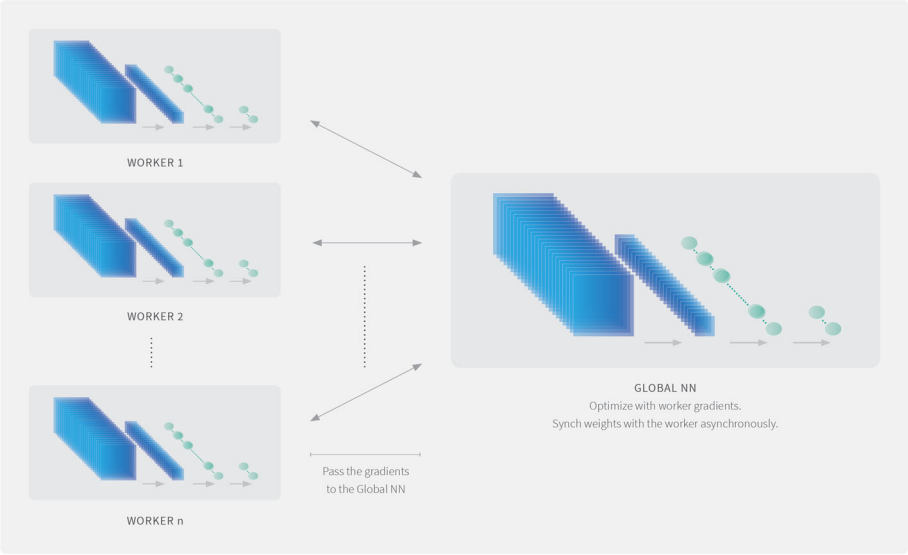



**Fig. 4** Asynchronous Advantage Actor-Critic (A3C) employs multiple (CPUs) workers without needing an ER buffer. Each worker has its own NN and independently interacts with the environment to compute the loss and gradients. Workers then pass computed gradients to the global NN that optimizes the parameters and synchronizes with the worker *asynchronously*. This distributed system is designed for single-agent deep RL. Compared to different DQN variants, A3C obtains better performance on a variety of Atari games using substantially less training time with multiple CPU cores of standard laptops without a GPU [219]. However, we note that more recent approaches use both multiple CPU cores for more efficient training data generation and GPUs for more efficient learning

vergence to suboptimal deterministic policies [219]. Thus, the loss function is given by: $\mathcal{L}_{\text{A3C}} = \lambda_v \mathcal{L}_v + \lambda_\pi \mathcal{L}_\pi - \lambda_H \mathbb{E}_{s\sim\pi}[H(\pi(s,\cdot,\theta)]$ with $\lambda_v$, $\lambda_\pi$, and $\lambda_H$, being weighting terms on the individual loss components. Wang et al. [344] took A3C's framework but used off-policy learning to create the Actor-critic with experience replay (ACER) algorithm. Gu et al. [118] introduced the Interpolated Policy Gradient (IPG) algorithm and showed a connection between ACER and DDPG: they are a pair of reparametrization terms (they are special cases of IPG) when they are put under the same stochastic policy setting, and when the policy is deterministic they collapse into DDPG.

Jaderberg et al. [158] built the Unsupervised Reinforcement and Auxiliary Learning (UNREAL) framework on top of A3C and introduced unsupervised *auxiliary tasks* (e.g., reward prediction) to speed up the learning process. Auxiliary tasks in general are not used for anything other than shaping the features of the agent, i.e., facilitating and regularizing the representation learning process [31,288]; their formalization in RL is related to the concept of *general value functions* [315,317]. The UNREAL framework optimizes a combined loss function $\mathcal{L}_{\text{UNREAL}} \approx \mathcal{L}_{\text{A3C}} + \sum_i \lambda_{AT_i} \mathcal{L}_{AT_i}$, that combines the A3C loss, $\mathcal{L}_{\text{A3C}}$, together with auxiliary task losses $\mathcal{L}_{AT_i}$, where $\lambda_{AT_i}$ are weight terms (see Sect. 4.1 for use of auxiliary tasks in MDRL). In contrast to A3C, UNREAL uses a prioritized ER buffer, in which transitions with positive reward are given higher probability of being sampled. This approach can be viewed as a simple form of prioritized replay [278], which was in turn inspired by model-based RL algorithms like prioritized sweeping [10,223].

Another distributed architecture is the Importance Weighted Actor-Learner Architecture (IMPALA) [93]. 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.

Trust Region Policy Optimization (TRPO) [283] and Proximal Policy Optimization (PPO) [284] are recently proposed policy gradient algorithms where the latter represents the state-of-the art with advantages such as being simpler to implement and having better empirical sample complexity. Interestingly, a recent work [151] studying PPO and TRPO arrived at the surprising conclusion that these methods often deviate from what the theoretical framework would predict: gradient estimates are poorly correlated with the true gradient and value networks tend to produce inaccurate predictions for the true value function. Compared to vanilla policy gradient algorithms, PPO prevents abrupt changes in policies during training through the loss function, similar to early work by Kakade [166]. Another advantage of PPO is that it can be used in a distributed fashion, i.e, Distributed PPO (DPPO) [134]. Note that *distributed approaches* like DPPO or A3C use parallelization only to improve the learning by more efficient training data generation through multiple CPU cores for single agent DRL and they should not be considered multiagent approaches (except for recent work which tries to exploit this parallelization in a multiagent environment [19]).

Lastly, there's a connection between policy gradient algorithms and Q-learning [282] within the framework of entropy-regularized reinforcement learning [126] where the value and $Q$ functions are slightly altered to consider the entropy of the policy. In this vein, Soft Actor-Critic (SAC) [127] is a recent algorithm that concurrently learns a stochastic policy, two Q-functions (taking inspiration from Double Q-learning) and a value function. SAC alternates between collecting experience with the current policy and updating from batches sampled from the ER buffer.

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 [55]. The *independent* learners, a.k.a. *decentralized* learners approach [323] directly uses single-agent algorithms in the multi-agent setting despite the underlying assumptions of these algorithms being violated (each agent independently learns its own policy, treating other agents as part of the environment). In particular the *Markov property* (the future dynamics, transitions, and rewards depend only on the current state) becomes invalid since the environment is no longer stationary [182,233,333]. 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 [289]. Despite the lack of guarantees, independent learners have been used in practice, providing advantages with regards to scalability while often achieving good results [213].

To understand why multiagent domains are non-stationary from agents' local perspectives, consider a simple stochastic (also known as Markov) game $(\mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, which can be seen as an extension of an MDP to multiple agents [198,200]. One key distinction is that the transition, $\mathcal{T}$, and reward function, $\mathcal{R}$, depend on the actions $\mathcal{A} = A_1 \times \cdots \times A_{\mathcal{N}}$ of all, $\mathcal{N}$, agents, this means, $\mathcal{R} = R_1 \times \cdots \times R_{\mathcal{N}}$ and $\mathcal{T} = \mathcal{S} \times A_1 \times \cdots \times A_{\mathcal{N}}$.

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 $\boldsymbol{\pi}(s, \boldsymbol{a}) = \prod_j \pi_j(s, a_j)$[6]:

$$V_i^{\boldsymbol{\pi}}(s) = \sum_{\boldsymbol{a} \in \mathcal{A}} \boldsymbol{\pi}(s, \boldsymbol{a}) \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a_i, \boldsymbol{a_{-i}}, s')[R_i(s, a_i, \boldsymbol{a_{-i}}, s') + \gamma V_i(s')]. \tag{4}$$

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

$$\begin{aligned} \pi_i^*(s, a_i, \boldsymbol{\pi_{-i}}) &= \arg\max_{\pi_i} V_i^{(\pi_i, \boldsymbol{\pi_{-i}})}(s) \\ &= \arg\max_{\pi_i} \sum_{\boldsymbol{a} \in \mathcal{A}} \pi_i(s, a_i) \boldsymbol{\pi_{-i}}(s, \boldsymbol{a_{-i}}) \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a_i, \boldsymbol{a_{-i}}, s')[R_i(s, a_i, \boldsymbol{a_{-i}}, s') \\ &\quad + \gamma V_i^{(\pi_i, \boldsymbol{\pi_{-i}})}(s')]. \end{aligned} \tag{5}$$

Specifically, the opponents' joint policy $\boldsymbol{\pi_{-i}}(s, \boldsymbol{a_{-i}})$ can be non-stationary, i.e., changes as the opponents' policies change over time, for example with learning opponents.

*Convergence results* Littman [200] studied convergence properties of reinforcement learning joint action agents [70] in Markov games with the following conclusions: in adversarial environments (zero-sum games) an optimal play can be guaranteed against an arbitrary opponent, i.e., Minimax Q-learning [198]. In coordination environments (e.g., in cooperative games all agents share the same reward function), strong assumptions need be made about other agents to guarantee convergence to optimal behavior [200], e.g., Nash Q-learning [149] and Friend-or-Foe Q-learning [199]. In other types of environments no value-based RL algorithms with guaranteed convergence properties are known [200].

Recent work on MDRL have addressed scalability and have focused significantly less on convergence guarantees, with few exceptions [22,40,255,297]. One notable work has shown a connection between update rules for actor-critic algorithms for multiagent partially observable settings and (counterfactual) regret minimization[7]: the advantage values are scaled counterfactual regrets. This lead to new convergence properties of independent RL algorithms in zero-sum games with imperfect information [300]. The result is also used to support policy gradient optimization against worst-case opponents, in a new algorithm called Exploitability Descent [204].[8]

We refer the interested reader to seminal works about convergence in multiagent domains [23,41,42,45,113,165,167,277,295,357,367]. Note that instead of convergence,

[6] In this setting each agent independently executes a policy, however, there are other cases where this does not hold, for example when agents have a coordinated exploration strategy.

[7] Counterfactual regret minimization is a technique for solving large games based on regret minimization [230, 368] due to a well-known connection between regret and Nash equilibria [39]. It has been one of the reasons of successes in Poker [50,224].

[8] This algorithm is similar to CFR-BR [159] and has the main advantage that the current policy convergences rather than the average policy, so there is no need to learn the average strategy, which requires large reservoir buffers or many past networks.

some MAL algorithms have proved learning a best response against classes of opponents [66,326,349].

There are other common problems in MAL, including action shadowing [105,347], the curse of dimensionality [55], and multiagent credit assignment [2]. Describing each problem is out of the scope of this survey. However, we refer the interested reader to excellent resources on general MAL [209,333,350], as well as surveys in specific areas: game theory and multiagent reinforcement learning [55,233], cooperative scenarios [213,248], evolutionary dynamics of multiagent learning [38], learning in non-stationary environments [141], agents modeling agents [6], and transfer learning in multiagent RL [290].

### 3.2 MDRL categorization

In Sect. 2.2 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 [13,191,251]. One natural extension to DRL is to test whether these approaches could be applied in a multiagent environment.

We analyzed the most recent works (that are not covered by previous MAL surveys [6,141] and we do not consider genetic algorithms or swarm intelligence in this survey) that have a clear connection with MDRL. We propose 4 categories which take inspiration from previous surveys [6,55,248,305] and that conveniently describe and represent current works. Note that some of these works fit into more than one category (they are not mutually exclusive), therefore their summaries are presented in all applicable Tables 1, 2, 3 and 4, however, for the ease of exposition when describing them in the text we only do so in one category. Additionally, for each work we present its learning type, either a value-based method (e.g., DQN) or a policy gradient method (e.g., actor-critic); also, we mention if the setting is evaluated in a fully cooperative, fully competitive or mixed environment (both cooperative and competitive).

- *Analysis of emergent behaviors* These works, in general, do not propose learning algorithms—their main focus is to analyze and evaluate DRL algorithms, e.g., DQN [188,264,322], PPO [24,264] and others [187,225,264], in a multiagent environment. In this category we found works which analyze behaviors in the three major settings: cooperative, competitive and mixed scenarios; see Sect. 3.3 and Table 1.
- *Learning communication* [96,183,225,253,256,312]. These works explore a sub-area in which agents can share information with communication protocols, for example through direct messages [96] or via a shared memory [256]. This area is attracting attention and it had not been explored much in the MAL literature. See Sect. 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 [213,248]. In this category the analyzed works are evaluated in either cooperative or mixed settings. Some works in this category take inspiration from MAL (e.g., leniency, hysteresis, and difference rewards concepts) and extend them to the MDRL setting [98,244,247]. A notable exception [99] takes a key component from RL (i.e., experience replay buffer) and adapts it for MDRL. See Sect. 3.5 and Table 3.
- *Agents modeling agents* Albrecht and Stone [6] 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 [133,148,265], and others from MAL [97,136,180,263,358]. Modeling agents is helpful not only to cooperate, but also for modeling opponents [133,136,148,180], inferring goals [265], and accounting for the learning behavior of other

**Table 1** These papers analyze *emergent behaviors* in MDRL

| Work | Summary | Learning | Setting |
|---|---|---|---|
| Tampuu et al. [322] | Train DQN agents to play Pong. | VB | CO and CMP |
| Leibo et al. [188] | Train DQN agents to play sequential social dilemmas. | VB | Mixed |
| Lerer and Peysakhovich [189] | Propose DRL agents able to cooperate in social dilemmas. | VB | Mixed |
| Leibo et al. [187] | Propose Malthusian reinforcement learning which extends self-play to population dynamics. | VB | Mixed |
| Bansal et al. [24] | Train PPO agents in competitive MuJoCo scenarios. | PG | CMP |
| Raghu et al. [264] | Train PPO, A3C, and DQN agents in attacker-defender games. | VB, PG | CMP |
| Lazaridou et al. [183] | Train agents represented with NN to learn a communication language. | PG | CO |
| Mordatch and Abbeel [225] | Learn communication with an end-to-end differentiable model to train with backpropagation. | PG | CO |

Learning type is either value-based (VB) or policy gradient (PG). Setting where experiments were performed: cooperative (CO), competitive (CMP) or mixed. A detailed description is given in Sect. 3.3

agents [97]. In this category the analyzed algorithms present their results in either a competitive setting or a mixed one (cooperative and competitive). See Sect. 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

Some recent works have analyzed the previously mentioned *independent* DRL agents (see Sect. 3.1) from the perspective of types of emerging behaviors (e.g., cooperative or competitive).

One of the earliest MDRL works is by Tampuu et al. [322], which had two independent DQN learning agents to play the Atari 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 al. [188] meanwhile studied independent DQNs in the context of *sequential social dilemmas*: a Markov game that satisfies certain inequalities [188]. 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). In the related setting of one shot Markov social dilemmas, Lerer and Peysakhovich [189] extended the famous Tit-for-Tat (TFT)[9] strategy [15] for DRL (using function approximators) and showed (theoretically and experimentally) that such agents can maintain cooperation. To construct the agents they used self-play and two reward schemes: selfish and cooperative. Previously, different MAL

[9] TFT originated in an iterated prisoner's dilemma tournament and later inspired different strategies in MAL [258], its generalization, Godfather, is a representative of *leader strategies* [201].

**Table 2** These papers propose algorithms for *learning communication*

| Algorithm | Summary | Learning | Setting |
|---|---|---|---|
| Lazaridou et al. [183] | Train agents represented with NN to learn a communication language. | PG | CO |
| Mordatch and Abbeel [225] | Learn communication with an end-to-end differentiable model to train with backpropagation. | PG | CO |
| RIAL [96] | Use a single network (parameter sharing) to train agents that take environmental and communication actions. | VB | CO |
| DIAL [96] | Use gradient sharing during learning and communication actions during execution. | VB | CO |
| CommNet [312] | Use a continuous vector channel for communication on a single network. | PG | CO |
| BiCNet [253] | Use the actor-critic paradigm where communication occurs in the latent space. | PG | Mixed |
| MD-MADDPG [256] | Use of a shared memory as a means to multiagent communication. | PG | CO |
| MADDPG-MD [173] | Extend dropout technique to robustify communication when applied in multiagent scenarios with direct communication. | PG | CO |

Learning type is either value-based (VB) or policy gradient (PG). Setting were experiments were performed: cooperative (CO) or mixed. A more detailed description is given in Sect. 3.4

algorithms were designed to foster cooperation in social dilemmas with Q-learning agents [77, 303].

Self-play is a useful concept for learning algorithms (e.g., fictitious play [49]) since under certain classes of games it can guarantee convergence[10] and it has been used as a standard technique in previous RL and MAL works [43,291,325]. Despite its common usage self-play can be brittle to forgetting past knowledge [180,186,275] (see Sect. 4.5 for a note on the role of self-play as an open question in MDRL). To overcome this issue, Leibo et al. [187] proposed Malthusian reinforcement learning as an extension of self-play to population dynamics. The approach can be thought of as community coevolution and has been shown to produce better results (avoiding local optima) than independent agents with intrinsic motivation [30]. A limitation of this work is that it does not place itself within the state of the art in evolutionary and genetic algorithms. Evolutionary strategies have been employed for solving reinforcement learning problems [226] and for evolving function approximators [351]. Similarly, they have been used multiagent scenarios to compute approximate Nash equilibria [238] and as metaheuristic optimization algorithms [53,54,150,248].

Bansal et al. [24] explored the emergent behaviors in competitive scenarios using the MuJoCo simulator [327]. They trained independent learning agents with PPO and incorporated two main modifications to deal with the MAL nature of the problem. First, they used

[10] The average strategy profile of fictitious players converges to a Nash equilibrium in certain classes of games, e.g., two-player zero-sum and potential games [222].

**Table 3** These papers aim to *learn cooperation*

| Algorithm | Summary | Learning | Setting |
|---|---|---|---|
| Lerer and Peysakhovich [189] | Propose DRL agents able to cooperate in social dilemmas | VB | Mixed |
| MD-MADDPG [256] | Use of a shared memory as a means to multiagent communication | PG | CO |
| MADDPG-MD [173] | Extend dropout technique to robustify communication when applied in multiagent scenarios with direct communication | PG | CO |
| RIAL [96] | Use a single network (parameter sharing) to train agents that take environmental and communication actions | VB | CO |
| DIAL [96] | Use gradient sharing during learning and communication actions during execution | VB | CO |
| DCH/PSRO [180] | Policies can overfit to opponents: better compute approximate best responses to a mixture of policies | VB | CO and CMP |
| Fingerprints [99] | Deal with ER problems in MDRL by conditioning the value function on a fingerprint that disambiguates the age of the sampled data | VB | CO |
| Lenient-DQN [247] | Achieve cooperation by leniency, optimism in the value function by forgiving suboptimal (low-rewards) actions | VB | CO |
| Hysteretic-DRQN [244] | Achieve cooperation by using two learning rates, depending on the updated values together with multitask learning via policy distillation | VB | CO |
| WDDQN [365] | Achieve cooperation by leniency, weighted double estimators, and a modified prioritized experience replay buffer | VB | CO |
| FTW [156] | Agents act in a mixed environment (composed of teammates and opponents), it proposes a two-level architecture and population-based learning | PG | Mixed |
| VDN [313] | Decompose the team action-value function into pieces across agents, where the pieces can be easily added | VB | Mixed |
| QMIX [266] | Decompose the team action-value function together with a mixing network that can recombine them | VB | Mixed |
| COMA [98] | Use a centralized critic and a counter-factual advantage function based on solving the multiagent credit assignment | PG | Mixed |
| PS-DQN, PS-TRPO, PS-A3C [123] | Propose parameter sharing for learning cooperative tasks | VB, PG | CO |
| MADDPG [206] | Use an actor-critic approach where the critic is augmented with information from other agents, the actions of all agents | PG | Mixed |

Learning type is either value-based (VB) or policy gradient (PG). Setting where experiments were performed: cooperative (CO), competitive (CMP) or mixed. A more detailed description is given in Sect. 3.5

**Table 4** These papers consider *agents modeling agents*

| Algorithm | Summary | Learning | Setting |
|---|---|---|---|
| MADDPG [206] | Use an actor-critic approach where the critic is augmented with information from other agents, the actions of all agents. | PG | Mixed |
| DRON [133] | Have a network to infer the opponent behavior together with the standard DQN architecture. | VB | Mixed |
| DPIQN, DPIRQN [148] | Learn policy features from raw observations that represent high-level opponent behaviors via auxiliary tasks. | VB | Mixed |
| SOM [265] | 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. | PG | Mixed |
| NFSP [136] | Compute approximate Nash equilibria via self-play and two neural networks. | VB | CMP |
| PSRO/DCH [180] | Policies can overfit to opponents: better compute approximate best responses to a mixture of policies. | PG | CO and CMP |
| M3DDPG [190] | Extend MADDPG with minimax objective to robustify the learned policy. | PG | Mixed |
| LOLA [97] | Use a learning rule where the agent accounts for the parameter update of other agents to maximize its own reward. | PG | Mixed |
| ToMnet [263] | Use an architecture for end-to-end learning and inference of diverse opponent types. | PG | Mixed |
| Deep Bayes-ToMoP [358] | Best respond to opponents using Bayesian policy reuse, theory of mind, and deep networks. | VB | CMP |
| Deep BPR+[366] | Bayesian policy reuse and policy distillation to quickly best respond to opponents. | VB | CO and CMP |

Learning type is either value-based (VB) or policy gradient (PG). Setting where experiments were performed: cooperative (CO), competitive (CMP) or mixed. A more detailed description is given in Sect. 3.6

*exploration rewards* [122] 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. Exploration rewards come from early work in robotics [212] and single-agent RL [176], and their goal is to provide dense feedback for the learning algorithm to improve sample efficiency (Ng et al. [231] studied the theoretical conditions under which modifications of the reward function of an MDP preserve the optimal policy). For multiagent scenarios, these dense rewards help agents in the beginning phase of the training to learn basic non-competitive skills, increasing the probability of random actions from the agent yielding a positive reward. The second contribution was *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. [264] 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 [91,299]. Their reasoning is threefold: (i) these games provide a parameterized 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. [183] proposed a framework for language learning that relies on multiagent communication. The agents, represented by (feed-forward) neural networks, need to develop an *emergent language* to solve a task. The task is formalized as a *signaling game* [103] 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. The results show that agents can coordinate for the experimented visual-based domain. To analyze the semantic properties[11] of the learned communication protocol they looked whether symbol usage reflects the semantics of the visual space, and that despite some variation, many high level objects groups correspond to the same learned symbols using a t-SNE [210] based analysis (t-SNE is a visualization technique for high-dimensional data and it has also been used to better understand the behavior of trained DRL agents [29,362]). A key objective of this work was to determine if the agent's language could be human-interpretable. To achieve this, learned symbols were grounded with natural language by extending the signaling game with a supervised image labelling task (the sender will be encouraged to use conventional names, making communication more transparent to humans). To measure the interpretability of the extended game, a crowdsourced survey was performed, and in essence, the trained agent receiver was replaced with a human. The results showed that 68% of the cases, human participants picked the correct image.

Similarly, Mordatch and Abbeel [225] investigated the emergence of language with the difference that in their setting there were no explicit roles for the agents (i.e., sender or 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 [183,225]. This setting usually considers a set of *cooperative agents* in a *partially observable* environment (see Sect. 2.2) where agents need to maximize their shared utility by means of communicating information.

Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL) are two methods using deep networks to learn to communicate [96]. 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.

*Memory-driven* (MD) communication was proposed on top of the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [206] method. In MD-MADDPG [256], the agents use a shared memory as a communication channel: before taking an action, the agent first reads the memory, then writes a response. In this case the agent's policy becomes dependent on its private observation and its interpretation of the collective memory. Experiments were performed with two agents in cooperative scenarios. The results highlighted the fact

[11] The vocabulary that agents use was arbitrary and had no initial meaning. To understand its emerging semantics they looked at the relationship between symbols and the sets of images they referred to [183].

that the communication channel was used differently in each environment, e.g., in simpler tasks agents significantly decrease their memory activity near the end of the task as there are no more changes in the environment; in more complex environments, the changes in memory usage appear at a much higher frequency due to the presence of many sub-tasks.

Dropout [301] is a technique to prevent overfitting (in supervised learning this happens when the learning algorithm achieves good performance only on a specific data set and fails to generalize) in neural networks which is based on randomly dropping units and their connections during training time. Inspired by dropout, Kim et al. [173] proposed a similar approach in multiagent environments where direct communication through messages is allowed. In this case, the messages of other agents are dropped out at training time, thus the authors proposed the Message-Dropout MADDPG algorithm [173]. This method is expected to work in fully or limited communication environments. The empirical results show that with properly chosen message dropout rate, the proposed method both significantly improves the training speed and the robustness of learned policies (by introducing communication errors) during execution time. This capability is important as MDRL agents trained in simulated or controlled environments will be less fragile when transferred to more realistic environments.

While RIAL and DIAL used a discrete communication channel, CommNet [312] used 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) [253], communication takes place in the latent space (i.e., in the hidden layers). It also uses parameter sharing, however, it proposes bidirectional recurrent neural networks [285] 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.

Learning communication is an active area in MDRL with many open questions, in this context, we refer the interested reader to a recent work by Lowe et al. [205] where it discusses common pitfalls (and recommendations to avoid those) while measuring communication in multiagent environments.

### 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[12] that do not involve communication [213,248]. Therefore, it was a natural starting point for many recent MDRL works.

Foerster et al. [99] studied the simple scenario of *cooperation with independent Q-learning agents* (see Sect. 3.1), where the agents use the standard DQN architecture of neural networks and an experience replay buffer (see Fig. 3). However, for the ER to work, the data distribution needs to follow certain assumptions (see Sect. 2.2) which are no loger valid due to the multiagent nature of the world: the dynamics that generated the data in the ER no longer reflect the current dynamics, making the experience obsolete [99,194]. Their solution is to add information to the experience tuple that can help to *disambiguate the age of the*

[12] There is a large body of research on coordinating multiagent teams by specifying communication protocols [115,321]: these expect agents to know the team's goal as well as the tasks required to accomplish the goal.

*sampled data* from the replay memory. Two approaches were proposed. The first is Multiagent Importance Sampling which adds the probability of the joint action so an importance sampling correction [36,260] can computed when the tuple is later sampled for training. This was similar to previous works in adaptive importance sampling [4,102] and off-environment RL [68]. The second approach is Multiagent Fingerprints which adds the estimate (i.e., fingerprint) of other agents' policies (loosely inspired by Hyper-Q [326], see Sect. 4.1). For the practical implementation, good results were obtained by using the training iteration number and exploration rate as the fingerprint.

Gupta et al. [123] tackled cooperative environments in partially observable domains without explicit communication. They proposed *parameter sharing* (PS) as a way to improve learning in homogeneous multiagent environments (where agents have the same set of actions). The idea is to have one globally shared learning network that can still behave differently in execution time, i.e., because its inputs (individual agent observation and agent index) will be different. They tested three variations of this approach with parameter sharing: PS-DQN, PS-DDPG and PS-TRPO, which extended single-agent DQN, DDPG and TRPO algorithms, respectively. The results showed that PS-TRPO outperformed the other two. Note that Foerster et al. [96] concurrently proposed a similar concept, see Sect. 3.4.

Lenient-DQN (LDQN) [247] took the *leniency* concept [37] (originally presented in MAL) and extended their use to MDRL. The purpose of leniency is to overcome a pathology called relative overgeneralization [249,250,347]. Similar to other approaches designed to overcome relative overgeneralization (e.g., distributed Q-learning [181] and hysteretic Q-learning [213]) lenient learners initially maintain an optimistic disposition to mitigate the noise from transitions resulting in miscoordination, preventing agents from being drawn towards sub-optimal but wide peaks in the reward search space [246]. However, similar to other MDRL works [99], the LDQN 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, Decentralized-Hysteretic Deep Recurrent Q-Networks (DEC-HDRQNs) [244] were proposed for fostering cooperation among independent learners. The motivation is similar to LDQN, making an optimistic value update, however, their solution is different. Here, the authors took inspiration from Hysteretic Q-learning [213], originally presented in MAL, where two learning rates were used. A difference between lenient agents and hysteretic Q-learning is that lenient agents are only *initially* forgiving towards teammates. Lenient learners over time apply less leniency towards updates that would lower utility values, taking into account how frequently observation-action pairs have been encountered. The idea being that the transition from optimistic to average reward learner will help make lenient learners more robust towards misleading stochastic rewards [37]. Additionally, in DEC-HDRQNs the ER buffer is also extended 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, i.e., multi-task learning[62], DEC-HDRQNs make use of policy distillation [146,273] (see Sect. 4.1). In contrast to other approaches, DEC-HDRQNS are fully decentralized during learning and execution.

Weighted Double Deep Q-Network (WDDQN) [365] is based on having double estimators. This idea was originally introduced in Double Q-learning [130] and aims to remove the existing overestimation bias caused by using the maximum action value as an approximation for the maximum expected action value (see Sect. 4.1). It also uses a *lenient* reward [37] to be optimistic during initial phase of coordination and proposes a *scheduled* replay strategy in

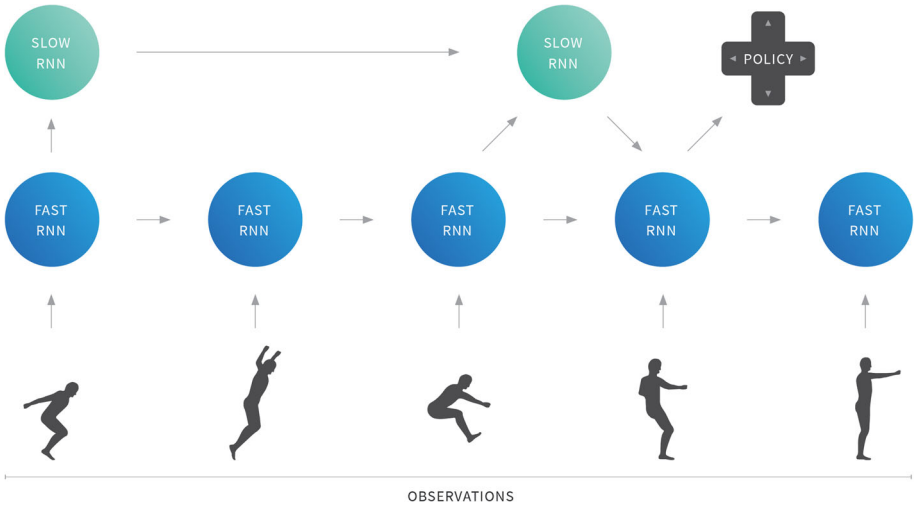



**Fig. 5** A schematic view of the architecture used in FTW (For the Win) [156]: two unrolled recurrent neural networks (RNNs) operate at different time-scales, the idea is that the *Slow RNN* helps with long term temporal correlations. Observations are latent space output of some convolutional neural network to learn non-linear features. Feudal Networks [338] is another work in single-agent DRL that also maintains a multi-time scale hierarchy where the slower network sets the goal, and the faster network tries to achieve them. Fedual Networks were in turn, inspired by early work in RL which proposed a hierarchy of Q-learners [82,296]

which samples closer to the terminal states are heuristically given higher priority; this strategy might not be applicable for any domain. For other works extending the ER to multiagent settings see MADDPG [206], Sects. 4.1 and 4.2.

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) [156] agent which is based on the actor-learner structure of IMPALA [93] (see Sect. 2.2). The authors test FTW in a game where two opposing teams compete to capture each other's flags [57]. To deal with the MAL problem they propose two main additions: a *hierarchical two-level representation* with recurrent neural networks operating at different timescales, as depicted in Fig. 5, and a *population based training* [157,185,271] where 30 agents were trained in parallel together with a stochastic matchmaking scheme that biases agents to be of similar *skills*. The Elo rating system [90] was originally devised to rate chess player skills,[13] TrueSkill [138] extended Elo by tracking uncertainty in skill rating, supporting draws, and matches beyond 1 vs 1; $\alpha-$Rank is a more recent alternative to ELO [243]. FTW did not use TrueSkill but a simpler extension of Elo for $n$ vs $n$ games (by adding individual agent ratings to compute the team skill). Hierarchical approaches were previously proposed in RL, e.g., Feudal RL [82,296], and were later extended to DRL in Feudal networks [338]; population based training can be considered analogous to evolutionary strategies that employ self-adaptive hyperparameter tuning to modify how the genetic algorithm itself operates [20,85,185]. An interesting result from FTW is that the population-based training obtained better results than training via self-play [325], which was a standard concept in previous works [43,291]. FTW used heavy compute resources, it used 30 agents

[13] Elo uses a normal distribution for each player skill, and after each match, both players' distributions are updated based on measure of *surprise*, i.e., if a user with previously lower (predicted) skill beats a high skilled one, the low-skilled player is significantly increased.

(processes) in parallel where every training game lasted 4500 agent steps ($\approx$ 5 min) and agents were trained for two billion steps ($\approx$ 450K games).

Lowe et al. [206] noted that using standard policy gradient methods (see Sect. 2.1) on multiagent environments yields high variance and performs poorly. This occurs because the variance is further increased as all the agents' rewards depend on the rest of the agents, and it is formally shown that as the number of agents increase, the probability of taking a correct gradient direction decreases exponentially [206]. Therefore, to overcome this issue Lowe et al. proposed the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [206], building on DDPG [192] (see Sect. 2.2), to train a centralized critic per agent that is given all agents' policies during training to reduce the variance by removing the non-stationarity caused by the concurrently learning agents. Here, the actor only has local information (turning the method into a centralized training with decentralized execution) and the ER buffer records experiences of *all* agents. MADDPG was tested in both cooperative and competitive scenarios, experimental results show that it performs better than several decentralized methods (such as DQN, DDPG, and TRPO). The authors mention that traditional RL methods do not produce consistent gradient signals. This is exemplified in a challenging competitive scenarios where agents continuously adapt to each other causing the learned best-response policies oscillate—for such a domain, MADDPG is shown to learn more robustly than DDPG.

Another approach based on policy gradients is the Counterfactual Multi-Agent Policy Gradients (COMA) [98]. COMA was designed for the fully centralized setting and the *multi-agent credit assignment problem* [332], i.e., how the agents should deduce their contributions when learning in a cooperative setting in the presence of only 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. This counterfactual baseline has its roots in *difference rewards*, which is a method for obtaining the individual contribution of an agent in a cooperative multiagent team [332]. In particular, the *aristocrat* utility aims to measure the difference between an agent's actual action and the average action [355]. The intention would be equivalent to sideline the agent by having the agent perform an action where the reward does not depend on the agent's actions, i.e., to consider the reward that would have arisen assuming a world without that agent having ever existed (see Sect. 4.2).

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 [119,174]. Value Decomposition Networks (VDNs) [313] decompose a team value function into an additive decomposition of the individual value functions. Similarly, QMIX [266] 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 monotonic action-value functions. While the mentioned approaches have obtained good empirical results, the factorization of value-functions in multiagent scenarios using function approximators (MDRL) is an ongoing research topic, with open questions such as how well factorizations capture complex coordination problems and how to learn those factorizations [64] (see Sect. 4.4).

### 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 [6]. An early work for

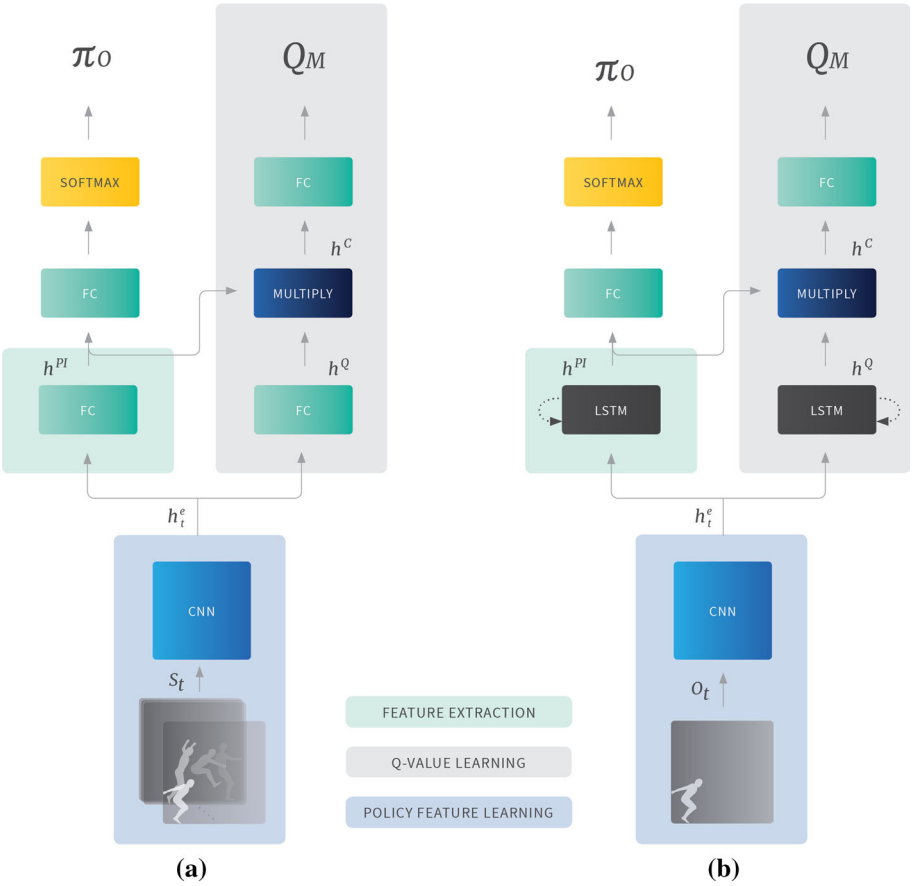



**Fig. 6** **a** Deep policy inference Q-network: receives four stacked frames as input (similar to DQN, see Fig. 2). **b** Deep Policy Inference Recurrent Q-Network: receives one frame as input and has an LSTM layer instead of a fully connected layer (FC). Both approaches [148] condition the $Q_M$ value outputs on the policy features, $h^{PI}$, which are also used to learn the opponent policy $\pi_o$

modeling agents while using deep neural networks was the Deep Reinforcement Opponent Network (DRON) [133]. 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 [109]. This is related to previous works in type-based reasoning from game theory [129,167] later applied in AI [6,26,109]. The mixture of experts idea was presented in supervised learning where each expert handled a subset of the data (a subtask), and then a gating network decided which of the experts should be used [155].

DRON uses hand-crafted features to define the opponent network. In contrast, Deep Policy Inference Q-Network (DPIQN) and its recurrent version, DPIRQN [148] learn *policy features* directly from raw observations of the other agents. The way to learn these policy features is by means of auxiliary tasks [158,317] (see Sects. 2.2 and 4.1) that provide additional learning goals, in this case, the auxiliary task is to learn the opponents' policies. This auxiliary task

modifies the loss function by computing an auxiliary loss: the cross entropy loss between the inferred opponent policy and the ground truth (one-hot action vector) of the opponent. Then, the $Q$ value function of the learning agent is conditioned on the opponent's policy features (see Fig. 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 [148].

In many previous works an opponent model is learned from observations. Self Other Modeling (SOM) [265] proposed a different approach, this is, using *the agent's own policy as a means to predict the opponent's actions*. SOM can be used in cooperative and competitive settings (with an arbitrary number of agents) and infers other agents' goals. This is important because in the evaluated domains, the reward function depends on the goal of the agents. SOM uses two networks, one used for computing the agents' own policy, and a second one used to infer the opponent's goal. The idea is that these networks have the same input parameters but with different values (the agent's or the opponent's). In contrast to previous approaches, SOM is not focused on learning the opponent policy, i.e., a probability distribution over next actions, but rather on estimating the opponent's goal. SOM is expected to work best when agents share a set of goals from which each agent gets assigned one at the beginning of the episode and the reward structure depends on both of their assigned goals. Despite its simplicity, training takes longer as an additional optimization step is performed given the other agent's observed actions.

There is a long-standing history of combining game theory and MAL [43,233,289]. From that context, some approaches were inspired by influential game theory approaches. Neural Fictitious Self-Play (NFSP) [136] builds on fictitious (self-) play [49,135], together with two deep networks to find *approximate Nash equilibria*[14] in two-player imperfect information games [341] (for example, consider Poker: when it is an agent's turn to move it does not have access to all information about the world). One network learns an *approximate best response* ($\epsilon-$greedy over $Q$ values) to the historical behavior of other agents and the second one (called the average network) learns to imitate its own past best response behaviour using supervised classification. The agent behaves using a mixture of the average and the best response networks depending on the probability of an anticipatory parameter [287]. Comparisons with DQN in Leduc Hold'em Poker revealed that DQN's deterministic strategy is highly exploitable. Such strategies are sufficient to behave optimally in single-agent domains, i.e., MDPs for which DQN was designed. However, imperfect-information games generally require stochastic strategies to achieve optimal behaviour [136]. DQN learning experiences are both highly correlated over time, and highly focused on a narrow state distribution. In contrast to NFSP agents whose experience varies more smoothly, resulting in a more stable data distribution, more stable neural networks and better performance.

The (N)FSP concept was further generalized in Policy-Space Response Oracles (PSRO) [180], where it was shown that fictitious play is one specific meta-strategy distribution over a set of previous (approximate) best responses (summarized by a meta-game obtained by empirical game theoretic analysis [342]), but there are a wide variety to choose from. One reason to use mixed meta-strategies is that it prevents overfitting[15] the responses to one specific policy,

[14] Nash equilibrium [229] is a solution concept in game theory in which no agent would choose to deviate from its strategy (they are a best response to others' strategies). This concept has been explored in seminal MAL algorithms like Nash-Q learning [149] and Minimax-Q learning [198,199].

[15] Johanson et al. [160] also found "overfitting" when solving large extensive games (e.g., poker)—the performance in an abstract game improved but it was worse in the full game.

and hence provides a form of opponent/teammate regularization. An approximate scalable version of the algorithm leads to a graph of agents best-responding independently called Deep Cognitive Hierarchies (DCHs) [180] due to its similarity to behavioral game-theoretic models [59,72].

Minimax is a paramount concept in game theory that is roughly described as minimizing the worst case scenario (maximum loss) [341]. Li et al. [190] took the minimax idea as an approach to robustify learning in multiagent environments so that the learned robust policy should be able to behave well even with strategies not seen during training. They extended the MADDPG algorithm [206] to Minimax Multiagent Deep Deterministic Policy Gradients (M3DDPG), which updates policies considering a worst-case scenario: assuming that all other agents act adversarially. This yields a minimax learning objective which is computationally intractable to directly optimize. They address this issue by taking ideas from robust reinforcement learning [227] which implicitly adopts the minimax idea by using the *worst noise* concept [257]. In MAL different approaches were proposed to assess the robustness of an algorithm, e.g., guarantees of safety [66,259], security [73] or exploitability [80,161,215].

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) [97]. 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. LOLA builds on previous ideas by Zhang and Lesser [363] where the learning agent predicts the opponent's policy parameter update but only uses it to learn a best response (to the anticipated updated parameters).

Theory of mind is part of a group of *recursive reasoning* approaches[60,61,109,110] in which agents have explicit beliefs about the mental states of other agents. The mental states of other agents may, in turn, also contain beliefs and mental states of other agents, leading to a nesting of beliefs [6]. Theory of Mind Network (ToMnet) [263] 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*. ToMnet has 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. A main advantage of ToMnet is that it can predict general behavior, for all agents; or specific, for a particular agent.

Deep Bayesian Theory of Mind Policy (Bayes-ToMoP) [358] is another algorithm that takes inspiration from theory of mind [76]. The algorithm assumes the opponent has different stationary strategies to act and changes among them over time [140]. Earlier work in MAL dealt with this setting, e.g., BPR+ [143] extends the Bayesian policy reuse[16] framework [272] to multiagent settings (BPR assumes a single-agent environment; BPR+ aims to best respond to the opponent in a multiagent game). A limitation of BPR+ is that it behaves poorly against itself (self-play), thus, Deep Bayes-ToMoP uses theory of mind to provide a higher-level reasoning strategy which provides an optimal behavior against BPR+ agents.

Deep BPR+ [366] is another work inspired by BPR+ which uses neural networks as value-function approximators. It not only uses the environment reward but also uses the

[16] Bayesian policy reuse assumes an agent with prior experience in the form of a library of policies. When a novel task instance occurs, the objective is to reuse a policy from its library based on observed signals which correlate to policy performance [272].

online learned opponent model [139,144] to construct a rectified belief over the opponent strategy. Additionally, it leverages ideas from policy distillation [146,273] and extends them to the multiagent case to create a distilled policy network. In this case, whenever a new acting policy is learned, distillation is applied to consolidate the new updated library which improves in terms of storage and generalization (over opponents).

## 4 Bridging RL, MAL and MDRL

This section aims to provide directions to promote fruitful cooperations between subcommunities. First, we address the pitfall of *deep learning amnesia*, roughly described as missing citations to the original works and not exploiting the advancements that have been made in the past. We present examples on how ideas originated earlier, for example in RL and MAL, were successfully extended to MDRL (see Sect. 4.1). Second, we outline *lessons learned* from the works analyzed in this survey (see Sect. 4.2). Then we point the readers to recent benchmarks for MDRL (see Sect. 4.3) and we discuss the practical challenges that arise in MDRL like high computational demands and reproducibility (see Sect. 4.4). Lastly, we pose some open research challenges and reflect on their relation with previous open questions in MAL [6] (see Sect. 4.5).

### 4.1 Avoiding deep learning amnesia: examples in MDRL

This survey focuses on recent *deep* works, however, in previous sections, when describing recent algorithms, we also point to original works that inspired them. Schmidhuber said "Machine learning is the science of credit assignment. The machine learning community itself profits from proper credit assignment to its members" [280]. In this context, we want to avoid committing the pitfall of not giving credit to original ideas that were proposed earlier, a.k.a. *deep learning amnesia*. Here, we provide some specific examples of research milestones that were studied earlier, e.g., RL or MAL, and that now became highly relevant for MDRL. Our purpose is to highlight that existent literature contains pertinent ideas and algorithms that should not be ignored. On the contrary, they should be examined and cited [58,79] to understand recent developments [343].

*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 [182,333]. Many MAL algorithms tried to solve this problem in different ways [141]. One example is *Hyper-Q* [326] which accounts for the (values of mixed) strategies of other agents and includes 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. [96] make use of this insight to propose their *fingerprint* algorithm in an MDRL problem (see Sect. 3.5). Other examples include the leniency concept [37] and Hysteretic Q-learning [213] originally presented in MAL, which now have their "deep" counterparts, LDQNs [247] and DEC-HDRQNs[244], see Sect. 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 [3]. 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 does a

single agent's actions contribute to a system that involves the actions of many agents [2]. A solution that came from MAL research that has proven successful in many scenarios is *difference rewards* [3,86,332], which aims to capture an agent's contribution to the system's global performance. In particular the *aristocrat* utility aims to measure the difference between an agent's actual action and the average action [355], however, it has a self-consistency problem and in practice it is more common to compute the *wonderful life utility* [355,356], which proposes to use a clamping operation that would be equivalent to removing that player from the team. COMA [98] builds on these concepts to propose an *advantage function* based on the contribution of the agent, which can be efficiently computed with deep neural networks (see Sect. 3.5).

*Multitask learning* In the context of RL, multitask learning [62] is an area that develops agents that can act in *several related tasks* rather than just in a single one [324]. *Distillation*, roughly defined as transferring the knowledge from a large model to a small model, was a concept originally introduced for supervised learning and model compression [52,146]. Inspired by those works, Policy distillation [273] was extended to the DRL realm. Policy distillation was used to train a much smaller network and to merge *several task-specific policies* into a single policy, i.e., for multitask learning. In the MDRL setting, Omidshafiei et al. [244] successfully adapted policy distillation within Dec-HDRQNs to obtain a more general multitask multiagent network (see Sect. 3.5). Another example is Deep BPR+ [366] which uses distillation to generalize over multiple opponents (see Sect. 3.6).

*Auxiliary tasks* Jaderberg et al. [158] introduced the term auxiliary task with the insight that (single-agent) environments contain a variety of possible training signals (e.g., pixel changes). These tasks are naturally implemented in DRL in which the last layer is split into multiple parts (heads), each working on a different task. All heads propagate errors into the same shared preceding part of the network, which would then try to form representations, in its next-to-last layer, to support all the heads [315]. However, the idea of multiple predictions about arbitrary signals was originally suggested for RL, in the context of general value functions [315,317] and there still open problems, for example, better theoretical understanding [31,88]. In the context of neural networks, early work proposed *hints* that improved the network performance and learning time. Suddarth and Kergosien [311] presented a minimal example of a small neural network where it was shown that adding an auxiliary task effectively removed local minima. One could think of extending these auxiliary tasks to modeling other agents' behaviors [142,225], which is one of the key ideas that DPIQN and DRPIQN [148] proposed in MDRL settings (see Sect. 3.6).

*Experience replay* Lin [193,194] proposed the concept of experience replay to speed up the credit assignment propagation process in single agent RL. This concept became central to many DRL works [220] (see Sect. 2.2). However, Lin stated that a condition for the ER to be useful is that "the environment should not change over time because this makes past experiences irrelevant or even harmful" [194]. This is a problem in domains where many agents are learning since the environment becomes non-stationary from the point of view of each agent. Since DRL relies heavily on experience replay, this is an issue in MDRL: the non-stationarity introduced means that the dynamics that generated the data in the agent's replay memory no longer reflect the current dynamics in which it is learning [96]. To overcome this problem different methods have been proposed [99,244,247,365], see Sect. 4.2.

*Double estimators* Double Q-learning [130] proposed to reduce the overestimation of action values in Q-learning, this is caused by using the maximum action value as an approximation for the maximum expected action value. Double Q-learning works by keeping two Q functions and was proven to convergence to the optimal policy [130]. Later this idea was applied to arbitrary function approximators, including deep neural networks, i.e., Double DQN [336], which were naturally applied since two networks were already used in DQN (see Sect. 2.2). These ideas have also been recently applied to MDRL [365].

### 4.2 Lessons learned

We have exemplified how RL and MAL can be extended 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 [96] 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 due to a lack of theoretical guarantees under this setting, see Sects. 2.2 and 4.1. *Adding information in the experience tuple* that can help disambiguate the sample is the solution adopted in many works, whether a value based method [99,244,247,365] or a policy gradient method [206]. In this regard, it is an open question to consider how new DRL ideas could be best integrated into the ER [11,83,153,196,278] and how those ideas 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) [34,237],[17] in MDRL this new mixed paradigm has been commonly used [98,99,180,206,247,266] (a notable exception are DEC-HDRQNs [244] which perform learning and execution in a decentralized manner, see Sect. 3.5). Note that not all real-world problems fit into this paradigm and it is more common for robotics or games where a simulator is generally available [96]. The main benefit is that during learning *additional information can be used* (e.g., global state, action, or 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. This method was proposed concurrently in different works [96,124] and later it has been successfully applied in many others [99, 253,266,312,313].
- *Recurrent networks* Recurrent neural networks (RNNs) enhanced neural networks with a memory capability, however, they suffer from the vanishing gradient problem, which renders them inefficient for long-term dependencies [252]. However, RNN variants such as LSTMs [114,147] and GRUs (Gated Recurrent Unit) [67] addressed this challenge. In single-agent DRL, DRQN [131] initially proposed idea of using recurrent networks in single-agent *partially observable* environments. Then, Feudal Networks [338] proposed a hierarchical approach [82], *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 to

[17] Centralized planning and decentralized execution is also a standard paradigm for multiagent planning [239].

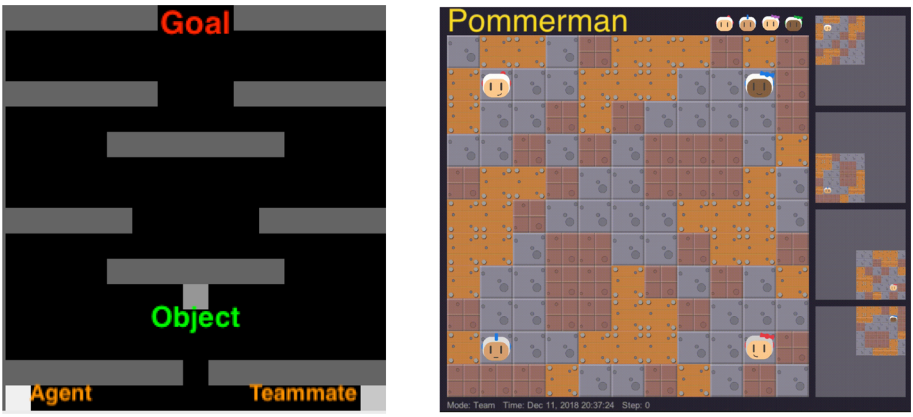
address the challenge of partially observability [24,96,148,244,253,263,265,266,313] for example, in FTW [156], depicted in Fig. 5 and DRPIRQN [148] depicted in Fig. 6. See Sect. 4.4 for practical challenges (e.g., training issues) of recurrent networks in MDRL.
– *Overfitting in MAL* In single-agent RL, agents can overfit to the environment [352]. A similar problem can occur in multiagent settings [160], agents can overfit, i.e., an agent's policy can easily get stuck in a local optima and the learned policy may be only locally optimal to other agents' current policies [190]. This has the effect of limiting the generalization of the learned policies [180]. To reduce this problem, a solution is to have a set of policies (an ensemble) and learn from them or best respond to the mixture of them [133,180,206]. Another solution has been to robustify algorithms—a robust policy should be able to behave well even with strategies different from its training (better generalization) [190].

### 4.3 Benchmarks for MDRL

Standardized environments such as the Arcade Learning Environment (ALE) [32,211] and OpenAI Gym [48] have allowed single-agent RL to move beyond toy domains. For DRL there are open-source frameworks that provide compact and reliable implementations of some state-of-the-art DRL algorithms [65]. Even though MDRL is a recent area, there are now a number of open sourced simulators and benchmarks to use with different characteristics, which we describe below.

– Fully Cooperative Multiagent Object Transporation Problems (CMOTPs)[18] were originally presented by Busoniu et al. [56] as a simple two-agent coordination problem in MAL. Palmer et al. [247] proposed two pixel-based extensions to the original setting which include narrow passages that test the agents' ability to master fully-cooperative sub-tasks, stochastic rewards and noisy observations, see Fig. 7a.
– The Apprentice Firemen Game[19] (inspired by the classic climb game [70]) is another two-agent pixel-based environment that simultaneously confronts learners with four pathologies in MAL: relative overgeneralization, stochasticity, the moving target problem, and alter exploration problem [246].
– Pommerman [267] is a multiagent benchmark useful for testing cooperative, competitive and mixed (cooperative and competitive) scenarios. It supports partial observability and communication among agents, see Fig. 7b. Pommerman is a very challenging domain from the exploration perspective as the rewards are very sparse and delayed [107]. A recent competition was held during NeurIPS-2018[20] and the top agents from that competition are available for training purposes.
– Starcraft Multiagent Challenge [276] is based on the real-time strategy game StarCraft II and focuses on micromanagement challenges,[21] that is, fine-grained control of individual units, where each unit is controlled by an independent agent that must act based on local observations. It is accompanied by a MDRL framework including state-of-the-art algorithms (e.g., QMIX and COMA).[22]

[18] https://github.com/gjp1203/nui_in_madrl.
[19] https://github.com/gjp1203/nui_in_madrl.
[20] https://www.pommerman.com/.
[21] https://github.com/oxwhirl/smac.
[22] https://github.com/oxwhirl/pymarl.




**(a)** Multiagent object transportation **(b)** Pommerman

**Fig. 7** **a** A fully cooperative benchmark with two agents, Multiagent Object Trasportation. **b** A mixed cooperative-competitive domain with four agents, Pommerman. For more MDRL benchmarks see Sect. 4.3

- The Multi-Agent Reinforcement Learning in Malmö (MARLÖ) competition [254] is another multiagent challenge with multiple cooperative 3D games[23] within Minecraft. The scenarios were created with the open source Malmö platform [162], providing examples of how a wider range of multiagent cooperative, competitive and mixed scenarios can be experimented on within Minecraft.
- Hanabi is a cooperative multiplayer card game (two to five players). The main characteristic of the game is that players do not observe their own cards but other players can reveal information about them. This makes an interesting challenge for learning algorithms in particular in the context of self-play learning and ad-hoc teams [5,44,304]. The Hanabi Learning Environment [25] was recently released[24] and it is accompanied with a baseline (deep RL) agent [145].
- Arena [298] is platform for multiagent research[25] based on the Unity engine [163]. It has 35 multiagent games (e.g., social dilemmas) and supports communication among agents. It has basseline implementations of recent DRL algorithms such as independent PPO learners.
- MuJoCo Multiagent Soccer [203] uses the MuJoCo physics engine [327]. The environment simulates a 2 vs. 2 soccer game with agents having a 3-dimensional action space.[26]
- Neural MMO [308] is a research platform[27] inspired by the human game genre of Massively Multiplayer Online (MMO) Role-Playing Games. These games involve a large, variable number of players competing to survive.

[23] https://github.com/crowdAI/marlo-single-agent-starter-kit/.

[24] https://github.com/deepmind/hanabi-learning-environment.

[25] https://github.com/YuhangSong/Arena-BuildingToolkit.

[26] https://github.com/deepmind/dm_control/tree/master/dm_control/locomotion/soccer.

[27] https://github.com/openai/neural-mmo.

### 4.4 Practical challenges in MDRL

In this section we take a more critical view with respect to MDRL and highlight different practical challenges that already happen in DRL and that are likely to occur in MDRL such as reproducibility, hyperparameter tuning, the need of computational resources and conflation of results. We provide pointers on how we think those challenges could be (partially) addressed.

*Reproducibility, troubling trends and negative results* Reproducibility is a challenge in RL which is only aggravated in DRL due to different sources of stochasticity: baselines, hyperparameters, architectures [137,228] and random seeds [69]. Moreover, DRL does not have common practices for statistical testing [100] which has led to bad practices such as only reporting the results when algorithms perform well, sometimes referred as *cherry picking* [16] (Azizzadenesheli also describes *cherry planting* as adapting an environment to a specific algorithm [16]). We believe that together with following the advice on how to design experiments and report results [197], the community would also benefit from reporting *negative results* [100,108,270,286] for carefully designed hypothesis and experiments.[28] However, we found very few papers with this characteristic[17,170,208] — we note that this is not encouraged in the ML community; moreover, negative results reduce the chance of paper acceptance [197]. In this regard, we ask the community to reflect on these practices and find ways to remove these obstacles.

*Implementation challenges and hyperparameter tuning* One problem is that canonical implementations of DRL algorithms often contain additional non-trivial optimizations—these are sometimes necessary for the algorithms to achieve good performance [151]. A recent study by Tucker et al. [331] found that several published works on action-dependant baselines contained bugs and errors—those were the real reason of the high performance in the experimental results, not the proposed method. Melis et al. [216] compared a series of works with increasing innovations in network architectures and the vanilla LSTMs [147] (originally proposed in 1997). The results showed that, when properly tuned, LSTMs outperformed the more recent models. In this context, Lipton and Steinhardt noted that the community may have benefited more by learning the details of the hyperparameter tuning [197]. A partial reason for this surprising result might be that this type of networks are known for being difficult to train [252] and there are recent works in DRL that report problems when using recurrent networks [78,95,106,123]. Another known complication is catastrophic forgetting (see Sect. 2.2) with recent examples in DRL [264,336]—we expect that these issues would likely occur in MDRL. The effects of hyperparameter tuning were analyzed in more detail in DRL by Henderson et al. [137], who arrived at the conclusion that hyperparameters can have significantly different effects across algorithms (they tested TRPO, DDPG, PPO and ACKTR) and environments since there is an intricate interplay among them [137]. The authors urge the community to report *all* parameters used in the experimental evaluations for accurate comparison—we encourage a similar behavior for MDRL. Note that hyperparameter tuning is related to the troubling trend of cherry picking in that it can show a carefully picked set of

[28] This idea was initially inspired by the Workshop "Critiquing and Correcting Trends in Machine Learning" at NeurIPS 2018 where it was possible to submit *Negative results* papers: "Papers which show failure modes of existing algorithms or suggest new approaches which one might expect to perform well but which do not. The aim is to provide a venue for work which might otherwise go unpublished but which is still of interest to the community." https://ml-critique-correct.github.io/.

parameters that make an algorithm work (see previous challenge). Lastly, note that hyperparameter tuning is computationally very expensive, which relates to the connection with the following challenge of computational demands.

*Computational resources* Deep RL usually requires millions of interactions for an agent to learn [9], i.e., low sample efficiency [361], which highlights the need for large computational infrastructure in general. The original A3C implementation [219] uses 16 CPU workers for 4 days to learn to play an Atari game with a total of 200M training frames[29] (results are reported for 57 Atari games). Distributed PPO used 64 workers (presumably one CPU per worker, although this is not clearly stated in the paper) for 100 hours (more than 4 days) to learn locomotion tasks [134]. In MDRL, for example, the Atari Pong game, agents were trained for 50 epochs, 250k time steps each, for a total of 1.25M training frames [322]. The FTW agent [156] uses 30 agents (processes) in parallel and every training game lasts for 5 min; FTW agents were trained for approximately 450K games $\approx$ 4.2 years. These examples highlight the computational demands sometimes needed within DRL and MDRL.

Recent works have reduced the learning of an Atari game to minutes (Stooke and Abbeel [306] trained DRL agents in less than one hour with hardware consisting of 8 GPUs and 40 cores). However, this is (for now) the exception and computational infrastructure is a major bottleneck for doing DRL and MDRL, especially for those who do not have such large compute power (e.g., most companies and most academic research groups) [29,286].[30] Within this context we propose two ways to address this problem. (1) Raising awareness: For DRL we found few works that study the computational demands of recent algorithms [9,18]. For MDRL most published works do not provide information regarding computational resources used such as CPU/GPU usage, memory demands, and wall-clock computation. Therefore, the first way to tackle this issue is by raising awareness and encouraging authors to report metrics about computational demands for accurately comparison and evaluation. (2) Delve into algorithmic contributions. Another way to address these issues is to prioritize the algorithmic contribution for the new MDRL algorithms rather than the computational resources spent. Indeed, for this to work, it needs to be accompanied with high-quality reviewers.

We have argued to raise awareness on the computational demands and report results, however, there is still the open question on *how* and *what* to measure/report. There are several dimensions to measure efficiency: sample efficiency is commonly measured by counting state-action pairs used for training; computational efficiency could be measured by number of CPUs/GPUs and days used for training. How do we measure the impact of other resources, such as external data sources or annotations?[31] Similarly, do we need to differentiate the computational needs of the algorithm itself versus the environment it is run in? We do not have the answers, however, we point out that current standard metrics might not be entirely comprehensive.

In the end, we believe that high compute based methods act as a frontier to showcase benchmarks [235,339], i.e., they show what results are possible as data and compute is scaled up (e.g., OpenAI Five generates 180 years of gameplay data each day using 128,000

[29] It is sometimes unclear in the literature what is the meaning of frame due to the "frame skip" technique. It is therefore suggested to refer to "game frames" and "training frames" [310].

[30] One recent effort by Beeching et al. [29] proposes to use only "mid-range hardware" (8 CPUs and 1 GPU) to train deep RL agents.

[31] NeurIPS 2019 hosts the "MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors" where the primary goal of the competition is to foster the development of algorithms which can efficiently leverage human demonstrations to drastically reduce the number of samples needed to solve complex, hierarchical, and sparse environments [125].

CPU cores and 256 GPUs [235]; AlphaStar uses 200 years of Starcraft II gameplay [339]); however, lighter compute based algorithmic methods can also yield significant contributions to better tackle real-world problems.

*Occam's razor and ablative analysis* Finding the simplest context that exposes the innovative research idea remains challenging, and if ignored it leads to a conflation of fundamental research (working principles in the most abstract setting) and applied research (working systems as complete as possible). In particular, some deep learning papers are presented as learning from pixels without further explanation, while object-level representations would have already exposed the algorithmic contribution. This still makes sense to remain comparable with established benchmarks (e.g., OpenAI Gym [48]), but less so if custom simulations are written without open source access, as it introduces unnecessary variance in pixel-level representations and artificially inflates computational resources (see previous point about *computational resources*).[32] In this context there are some notable exceptions where the algorithmic contribution is presented in a minimal setting and then results are scaled into complex settings: LOLA [97] first presented a minimalist setting with a two-player two-action game and then with a more complex variant; similarly, QMIX [266] presented its results in a two-step (matrix) game and then in the more involved Starcraft II micromanagement domain [276].

### 4.5 Open questions

Finally, here we present some open questions for MDRL and point to suggestions on how to approach them. We believe that there are solid ideas in earlier literature and we refer the reader to Sect. 4.1 to avoid deep learning amnesia.

- On the challenge of sparse and delayed rewards.
  Recent MDRL competitions and environments have complex scenarios where many actions are taken before a reward signal is available (see Sect. 4.3). This sparseness is already a challenge for RL [89,315] where approaches such as count-based exploration/intrinsic motivation [27,30,47,279,307] and hierarchical learning [87,178,278] have been proposed to address it—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 applied *dense* rewards [176,212,231] (a concept originated in RL) 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 [24], see Sect. 3.3. Recent works like OpenAI Five [235] uses hand-crafted intermediate rewards to accelerate the learning and FTW [156] lets the agents learn their internal rewards by a hierarchical two-tier optimization. In *single agent* domains, RUDDER [12] 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. How to best extend RUDDER to multiagent domains is an open avenue of research.
- On the role of self-play.
  Self-play is a cornerstone in MAL with impressive results [42,45,71,113,149]. While

[32] Cuccu, Togelius and Cudré-Mauroux achieved state-of-the-art policy learning in Atari games with only 6 to 18 neurons [75]. The main idea was to decouple image processing from decision-making.

notable results had also been shown in MDRL [43,136], recent works have also shown that *plain* self-play does not yield the best results. However, adding diversity, i.e., evolutionary methods [20,85,185,271] or sampling-based methods, have shown good results [24,156,187]. 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 improve the computational efficiency of these previously proposed successful methods, i.e., achieving similar training stability with smaller population sizes that uses fewer CPU workers in MAL and MDRL (see Sect. 4.4 and Albrecht et al. [6, Section 5.5]).

– On the challenge of the combinatorial nature of MDRL.
Monte Carlo tree search (MCTS) [51] has been the backbone of the major breakthroughs behind AlphaGo [291] and AlphaGo Zero [293] that combined search and DRL. A recent work [340] 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 [169]. One way to tackle this challenge within multiagent scenarios is the use of search parallelization [35,171]. Given more scalable planners, there is room for research in combining these techniques in MDRL settings.
To learn complex multiagent interactions some type of abstraction [84] is often needed, for example, factored value functions [8,119–121,174,236] (see QMIX and VDN in Sect. 3.5 for recent work in MDRL) try to exploit independence among agents through (factored) structure; however, in MDRL there are still open questions such as understanding their representational power [64] (e.g., the accuracy of the learned Q-function approximations) and how to learn those factorizations, where ideas from transfer planning techniques could be useful [240,335]. In transfer planning the idea is to define a simpler "source problem" (e.g., with fewer agents), in which the agent(s) can plan [240] or learn [335]; since it is less complex than the real multiagent problem, issues such as the non-stationarity of the environment can be reduced/removed. Lastly, another related idea are *influence* abstractions [28,141,241], where instead of learning a complex multiagent model, these methods try to build smaller models based on the influence agents can exert on one another. While this has not been sufficiently explored in actual multiagent settings, there is some evidence that these ideas can lead to effective inductive biases, improving effectiveness of DRL in such local abstractions [309].

## 5 Conclusions

Deep reinforcement learning has shown recent success on many fronts [221,224,291] 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 [45,55,141,246,305,332,348].

This survey provides a broad overview of recent works in the emerging area of Multiagent Deep Reinforcement Learning (MDRL). First, we categorized recent works into four different topics: emergent behaviors, learning communication, learning cooperation, and agents modeling agents. Then, we exemplified how key components (e.g., experience replay and difference rewards) originated in RL and MAL need to be adapted to work in MDRL. We provided general lessons learned applicable to MDRL, pointed to recent multiagent benchmarks

and highlighted some open research problems. Finally, we also reflected on the practical challenges such as computational demands and reproducibility in MDRL.

Our conclusions of this work are that 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 [81,151,211,328] and multiagent learning [116,172,195,242,245,360]. Our view is that there are practical issues within MDRL that hinder its scientific progress: the necessity of high compute power, complicated reproducibility (e.g., hyperparameter tuning), and the lack of sufficient encouragement for publishing negative results. However, we remain highly optimistic of the multiagent community and hope this work serves to raise those issues, encounter good solutions, and ultimately take advantage of the existing literature and resources available to move the area in the right direction.

**Acknowledgements** We would like to thank Chao Gao, Nidhi Hegde, Gregory Palmer, Felipe Leno Da Silva and Craig Sherstan for reading earlier versions of this work and providing feedback, to April Cooper for her visual designs for the figures in the article, to Frans Oliehoek, Sam Devlin, Marc Lanctot, Nolan Bard, Roberta Raileanu, Angeliki Lazaridou, and Yuhang Song for clarifications in their areas of expertise, to Baoxiang Wang for his suggestions on recent deep RL works, to Michael Kaisers, Daan Bloembergen, and Katja Hofmann for their comments about the practical challenges of MDRL, and to the editor and three anonymous reviewers whose comments and suggestions increased the quality of this work.

## References

1. Achiam, J., Knight, E., & Abbeel, P. (2019). *Towards characterizing divergence in deep Q-learning*. CoRR arXiv:1903.08894.
2. Agogino, A. K., & Tumer, K. (2004). Unifying temporal and structural credit assignment problems. In *Proceedings of 17th international conference on autonomous agents and multiagent systems*.
3. Agogino, A. K., & Tumer, K. (2008). Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, *17*(2), 320–338.
4. Ahamed, T. I., Borkar, V. S., & Juneja, S. (2006). Adaptive importance sampling technique for markov chains using stochastic approximation. *Operations Research*, *54*(3), 489–504.
5. Albrecht, S. V., & Ramamoorthy, S. (2013). A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 12th international conference on autonomous agents and multi-agent systems*. Saint Paul, MN, USA.
6. Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, *258*, 66–95.
7. Alonso, E., D'inverno, M., Kudenko, D., Luck, M., & Noble, J. (2002). Learning in multi-agent systems. *Knowledge Engineering Review*, *16*(03), 1–8.
8. Amato, C., & Oliehoek, F. A. (2015). Scalable planning and learning for multiagent POMDPs. In *AAAI* (pp. 1995–2002).
9. Amodei, D., & Hernandez, D. (2018). AI and compute. https://blog.openai.com/ai-and-compute.
10. Andre, D., Friedman, N., & Parr, R. (1998). Generalized prioritized sweeping. In *Advances in neural information processing systems* (pp. 1001–1007).
11. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., & Zaremba, W. (2017). Hindsight experience replay. In *Advances in neural information processing systems*.
12. Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., & Hochreiter, S. (2018). *RUDDER: Return decomposition for delayed rewards*. arXiv:1806.07857.
13. Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). *A brief survey of deep reinforcement learning*. arXiv:1708.05866v2.
14. Astrom, K. J. (1965). Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, *10*(1), 174–205.
15. Axelrod, R., & Hamilton, W. D. (1981). The evolution of cooperation. *Science*, *211*(27), 1390–1396.
16. Azizzadenesheli, K. (2019). Maybe a few considerations in reinforcement learning research? In *Reinforcement learning for real life workshop*.

17. Azizzadenesheli, K., Yang, B., Liu, W., Brunskill, E., Lipton, Z., & Anandkumar, A. (2018). Surprising negative results for generative adversarial tree search. In *Critiquing and correcting trends in machine learning workshop*.
18. Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., & Kautz, J. (2017). Reinforcement learning through asynchronous advantage actor-critic on a GPU. In *International conference on learning representations*.
19. Bacchiani, G., Molinari, D., & Patander, M. (2019). Microscopic traffic simulation by cooperative multi-agent deep reinforcement learning. In *AAMAS*.
20. Back, T. (1996). *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Oxford: Oxford University Press.
21. Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. *Machine Learning Proceedings*, *1995*, 30–37.
22. Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., & Graepel, T. (2018). The mechanics of n-player differentiable games. In *Proceedings of the 35th international conference on machine learning, proceedings of machine learning research* (pp. 354–363). Stockholm, Sweden.
23. Banerjee, B., & Peng, J. (2003). Adaptive policy gradient in multiagent learning. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (pp. 686–692). ACM.
24. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., & Mordatch, I. (2018). Emergent complexity via multi-agent competition. In *International conference on machine learning*.
25. Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., & Song, H. F., et al. (2019). *The Hanabi challenge: A new frontier for AI research*. arXiv:1902.00506.
26. Barrett, S., Stone, P., Kraus, S., & Rosenfeld, A. (2013). Teamwork with Limited Knowledge of Teammates. In Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, pp. 102–108. Bellevue, WS, USA.
27. Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. In M. Mirolli & G. Baldassarre (Eds.), *Intrinsically motivated learning in natural and artificial systems* (pp. 17–47). Berlin: Springer.
28. Becker, R., Zilberstein, S., Lesser, V., & Goldman, C. V. (2004). Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, *22*, 423–455.
29. Beeching, E., Wolf, C., Dibangoye, J., & Simonin, O. (2019). *Deep reinforcement learning on a budget: 3D Control and reasoning without a supercomputer*. CoRR arXiv:1904.01806.
30. Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems* (pp. 1471–1479).
31. Bellemare, M. G., Dabney, W., Dadashi, R., Taïga, A. A., Castro, P. S., & Roux, N. L., et al. (2019). *A geometric perspective on optimal representations for reinforcement learning*. CoRR arXiv:1901.11530.
32. Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.
33. Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, *6*(5), 679–684.
34. Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*(4), 819–840.
35. Best, G., Cliff, O. M., Patten, T., Mettu, R. R., & Fitch, R. (2019). Dec-MCTS: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research*, *38*(2–3), 316–337.
36. Bishop, C. M. (2006). *Pattern recognition and machine learning*. Berlin: Springer.
37. Bloembergen, D., Kaisers, M., & Tuyls, K. (2010). Lenient frequency adjusted Q-learning. In *Proceedings of the 22nd Belgian/Netherlands artificial intelligence conference*.
38. Bloembergen, D., Tuyls, K., Hennes, D., & Kaisers, M. (2015). Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, *53*, 659–697.
39. Blum, A., & Monsour, Y. (2007). Learning, regret minimization, and equilibria. Chap. 4. In N. Nisan (Ed.), *Algorithmic game theory*. Cambridge: Cambridge University Press.
40. Bono, G., Dibangoye, J. S., Matignon, L., Pereyron, F., & Simonin, O. (2018). Cooperative multi-agent policy gradient. In *European conference on machine learning*.
41. Bowling, M. (2000). Convergence problems of general-sum multiagent reinforcement learning. In International conference on machine learning (pp. 89–94).
42. Bowling, M. (2004). Convergence and no-regret in multiagent learning. *Advances in neural information processing systems* (pp. 209–216). Canada: Vancouver.
43. Bowling, M., Burch, N., Johanson, M., & Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, *347*(6218), 145–149.
44. Bowling, M., & McCracken, P. (2005). Coordination and adaptation in impromptu teams. *Proceedings of the nineteenth conference on artificial intelligence* (Vol. 5, pp. 53–58).

45. Bowling, M., & Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, *136*(2), 215–250.
46. Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems*, pp. 369–376.
47. Brafman, R. I., & Tennenholtz, M. (2002). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*(Oct), 213–231.
48. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI gym*. arXiv preprint arXiv:1606.01540.
49. Brown, G. W. (1951). Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, *13*(1), 374–376.
50. Brown, N., & Sandholm, T. (2018). Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, *359*(6374), 418–424.
51. Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., et al. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–43.
52. Bucilua, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 535–541). ACM.
53. Bull, L. (1998). Evolutionary computing in multi-agent environments: Operators. In *International conference on evolutionary programming* (pp. 43–52). Springer.
54. Bull, L., Fogarty, T. C., & Snaith, M. (1995). Evolution in multi-agent systems: Evolving communicating classifier systems for gait in a quadrupedal robot. In *Proceedings of the 6th international conference on genetic algorithms* (pp. 382–388). Morgan Kaufmann Publishers Inc.
55. Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156–172.
56. Busoniu, L., Babuska, R., & De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. In D. Srinivasan & L. C. Jain (Eds.), *Innovations in multi-agent systems and applications - 1* (pp. 183–221). Berlin: Springer.
57. Capture the Flag: The emergence of complex cooperative agents. (2018). [Online]. Retrieved September 7, 2018, https://deepmind.com/blog/capture-the-flag/.
58. Collaboration & Credit Principles, How can we be good stewards of collaborative trust? (2019). [Online]. Retrieved May 31, 2019, http://colah.github.io/posts/2019-05-Collaboration/index.html.
59. Camerer, C. F., Ho, T. H., & Chong, J. K. (2004). A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, *119*(3), 861.
60. Camerer, C. F., Ho, T. H., & Chong, J. K. (2004). Behavioural game theory: Thinking, learning and teaching. In *Advances in understanding strategic behavior* (pp. 120–180). New York.
61. Carmel, D., & Markovitch, S. (1996). Incorporating opponent models into adversary search. *AAAI/IAAI*, *1*, 120–125.
62. Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.
63. Cassandra, A. R. (1998). *Exact and approximate algorithms for partially observable Markov decision processes*. Ph.D. thesis, Computer Science Department, Brown University.
64. Castellini, J., Oliehoek, F. A., Savani, R., & Whiteson, S. (2019). The representational capacity of action-value networks for multi-agent reinforcement learning. In *18th International conference on autonomous agents and multiagent systems*.
65. Castro, P. S., Moitra, S., Gelada, C., Kumar, S., Bellemare, M. G. (2018). *Dopamine: A research framework for deep reinforcement learning*. arXiv:1812.06110.
66. Chakraborty, D., & Stone, P. (2013). Multiagent learning in the presence of memory-bounded agents. *Autonomous Agents and Multi-Agent Systems*, *28*(2), 182–213.
67. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Deep learning and representation learning workshop*.
68. Ciosek, K. A., & Whiteson, S. (2017). Offer: Off-environment reinforcement learning. In *Thirty-first AAAI conference on artificial intelligence*.
69. Clary, K., Tosch, E., Foley, J., & Jensen, D. (2018). Let's play again: Variability of deep reinforcement learning agents in Atari environments. In *NeurIPS critiquing and correcting trends workshop*.
70. Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th national conference on artificial intelligence* (pp. 746–752). Madison, Wisconsin, USA.
71. Conitzer, V., & Sandholm, T. (2006). AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, *67*(1–2), 23–43.

72. Costa Gomes, M., Crawford, V. P., & Broseta, B. (2001). Cognition and behavior in normal-form games: An experimental study. *Econometrica*, *69*(5), 1193–1235.
73. Crandall, J. W., & Goodrich, M. A. (2011). Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. *Machine Learning*, *82*(3), 281–314.
74. Crites, R. H., & Barto, A. G. (1998). Elevator group control using multiple reinforcement learning agents. *Machine Learning*, *33*(2–3), 235–262.
75. Cuccu, G., Togelius, J., & Cudré-Mauroux, P. (2019). Playing Atari with six neurons. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 998–1006). International Foundation for Autonomous Agents and Multiagent Systems.
76. de Weerd, H., Verbrugge, R., & Verheij, B. (2013). How much does it help to know what she knows you know? An agent-based simulation study. *Artificial Intelligence*, *199–200*(C), 67–92.
77. de Cote, E. M., Lazaric, A., & Restelli, M. (2006). Learning to cooperate in multi-agent social dilemmas. In *Proceedings of the 5th international conference on autonomous agents and multiagent systems* (pp. 783–785). Hakodate, Hokkaido, Japan.
78. Deep reinforcement learning: Pong from pixels. (2016). [Online]. Retrieved May 7, 2019, https://karpathy.github.io/2016/05/31/rl/.
79. Do I really have to cite an arXiv paper? (2017). [Online]. Retrieved May 21, 2019, http://approximatelycorrect.com/2017/08/01/do-i-have-to-cite-arxiv-paper/.
80. Damer, S., & Gini, M. (2017). Safely using predictions in general-sum normal form games. In *Proceedings of the 16th conference on autonomous agents and multiagent systems*. Sao Paulo.
81. Darwiche, A. (2018). Human-level intelligence or animal-like abilities? *Communications of the ACM*, *61*(10), 56–67.
82. Dayan, P., & Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in neural information processing systems* (pp. 271–278).
83. De Bruin, T., Kober, J., Tuyls, K., & Babuška, R. (2018). Experience selection in deep reinforcement learning for control. *The Journal of Machine Learning Research*, *19*(1), 347–402.
84. De Hauwere, Y. M., Vrancx, P., & Nowe, A. (2010). Learning multi-agent state space representations. In *Proceedings of the 9th international conference on autonomous agents and multiagent systems* (pp. 715–722). Toronto, Canada.
85. De Jong, K. A. (2006). *Evolutionary computation: A unified approach*. Cambridge: MIT press.
86. Devlin, S., Yliniemi, L. M., Kudenko, D., & Tumer, K. (2014). Potential-based difference rewards for multiagent reinforcement learning. In *13th International conference on autonomous agents and multiagent systems, AAMAS 2014*. Paris, France.
87. Dietterich, T. G. (2000). Ensemble methods in machine learning. In *MCS proceedings of the first international workshop on multiple classifier systems* (pp. 1–15). Springer, Berlin Heidelberg, Cagliari, Italy.
88. Du, Y., Czarnecki, W. M., Jayakumar, S. M., Pascanu, R., & Lakshminarayanan, B. (2018). *Adapting auxiliary losses using gradient similarity*. arXiv preprint arXiv:1812.02224.
89. Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., & Clune, J. (2019). Go-explore: A new approach for hard-exploration problems. arXiv preprint arXiv:1901.10995.
90. Elo, A. E. (1978). *The rating of chessplayers, past and present*. Nagoya: Arco Pub.
91. Erdös, P., & Selfridge, J. L. (1973). On a combinatorial game. *Journal of Combinatorial Theory, Series A*, *14*(3), 298–301.
92. Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*(Apr), 503–556.
93. Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., & Dunning, I., et al. (2018). IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International conference on machine learning*.
94. Even-Dar, E., & Mansour, Y. (2003). Learning rates for Q-learning. *Journal of Machine Learning Research*, *5*(Dec), 1–25.
95. Firoiu, V., Whitney, W. F., & Tenenbaum, J. B. (2017). *Beating the World's best at super smash Bros. with deep reinforcement learning*. CoRR arXiv:1702.06230.
96. Foerster, J. N., Assael, Y. M., De Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems* (pp. 2145–2153).
97. Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., & Mordatch, I. (2018). Learning with opponent-learning awareness. In *Proceedings of 17th international conference on autonomous agents and multiagent systems*. Stockholm, Sweden.
98. Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2017). Counterfactual multi-agent policy gradients. In *32nd AAAI conference on artificial intelligence*.

99. Foerster, J. N., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H. S., Kohli, P., & Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*.
100. Forde, J. Z., & Paganini, M. (2019). The scientific method in the science of machine learning. In *ICLR debugging machine learning models workshop*.
101. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J., et al. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, *11*(3–4), 219–354.
102. Frank, J., Mannor, S., & Precup, D. (2008). Reinforcement learning in the presence of rare events. In *Proceedings of the 25th international conference on machine learning* (pp. 336–343). ACM.
103. Fudenberg, D., & Tirole, J. (1991). *Game theory*. Cambridge: The MIT Press.
104. Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*.
105. Fulda, N., & Ventura, D. (2007). Predicting and preventing coordination problems in cooperative Q-learning systems. In *Proceedings of the twentieth international joint conference on artificial intelligence* (pp. 780–785). Hyderabad, India.
106. Gao, C., Hernandez-Leal, P., Kartal, B., & Taylor, M. E. (2019). Skynet: A top deep RL agent in the inaugural pommerman team competition. In *4th multidisciplinary conference on reinforcement learning and decision making*.
107. Gao, C., Kartal, B., Hernandez-Leal, P., & Taylor, M. E. (2019). On hard exploration for reinforcement learning: A case study in pommerman. In *AAAI conference on artificial intelligence and interactive digital entertainment*.
108. Gencoglu, O., van Gils, M., Guldogan, E., Morikawa, C., Süzen, M., Gruber, M., Leinonen, J., & Huttunen, H. (2019). *Hark side of deep learning–from grad student descent to automated machine learning*. arXiv preprint arXiv:1904.07633.
109. Gmytrasiewicz, P. J., & Doshi, P. (2005). A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, *24*(1), 49–79.
110. Gmytrasiewicz, P. J., & Durfee, E. H. (2000). Rational coordination in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, *3*(4), 319–350.
111. Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio, Y. (2013). *An empirical investigation of catastrophic forgetting in gradient-based neural networks*. arXiv:1312.6211
112. Gordon, G. J. (1999). *Approximate solutions to Markov decision processes*. Technical report, Carnegie-Mellon University.
113. Greenwald, A., & Hall, K. (2003). Correlated Q-learning. In *Proceedings of 17th international conference on autonomous agents and multiagent systems* (pp. 242–249). Washington, DC, USA.
114. Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232.
115. Grosz, B. J., & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, *86*(2), 269–357.
116. Grover, A., Al-Shedivat, M., Gupta, J. K., Burda, Y., & Edwards, H. (2018). Learning policy representations in multiagent systems. In *International conference on machine learning*.
117. Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., & Levine, S. (2017). Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International conference on learning representations*.
118. Gu, S. S., Lillicrap, T., Turner, R. E., Ghahramani, Z., Schölkopf, B., & Levine, S. (2017). Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in neural information processing systems* (pp. 3846–3855).
119. Guestrin, C., Koller, D., & Parr, R. (2002). Multiagent planning with factored MDPs. In *Advances in neural information processing systems* (pp. 1523–1530).
120. Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, *19*, 399–468.
121. Guestrin, C., Lagoudakis, M., & Parr, R. (2002). Coordinated reinforcement learning. In *ICML* (Vol. 2, pp. 227–234).
122. Gullapalli, V., & Barto, A. G. (1992). Shaping as a method for accelerating reinforcement learning. In *Proceedings of the 1992 IEEE international symposium on intelligent control* (pp. 554–559). IEEE.
123. Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In G. Sukthankar & J. A. Rodriguez-Aguilar (Eds.), *Autonomous agents and multiagent systems* (pp. 66–83). Cham: Springer.
124. Gupta, J. K., Egorov, M., & Kochenderfer, M. J. (2017). Cooperative Multi-agent Control using deep reinforcement learning. In *Adaptive learning agents at AAMAS*. Sao Paulo.

125. Guss, W. H., Codel, C., Hofmann, K., Houghton, B., Kuno, N., Milani, S., Mohanty, S. P., Liebana, D. P., Salakhutdinov, R., Topin, N., Veloso, M., & Wang, P. (2019). *The MineRL competition on sample efficient reinforcement learning using human priors*. CoRR arXiv:1904.10079.
126. Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 1352–1361).
127. Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*.
128. Hafner, R., & Riedmiller, M. (2011). Reinforcement learning in feedback control. *Machine Learning*, *84*(1–2), 137–169.
129. Harsanyi, J. C. (1967). Games with incomplete information played by "Bayesian" players, I–III part I. The basic model. *Management Science*, *14*(3), 159–182.
130. Hasselt, H. V. (2010). Double Q-learning. In *Advances in neural information processing systems* (pp. 2613–2621).
131. Hausknecht, M., & Stone, P. (2015). Deep recurrent Q-learning for partially observable MDPs. In *International conference on learning representations*.
132. Hauskrecht, M. (2000). Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, *13*(1), 33–94.
133. He, H., Boyd-Graber, J., Kwok, K., Daume, H. (2016). Opponent modeling in deep reinforcement learning. In *33rd international conference on machine learning* (pp. 2675–2684).
134. Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M. A., & Silver, D. (2017). *Emergence of locomotion behaviours in rich environments*. arXiv:1707.02286v2
135. Heinrich, J., Lanctot, M., & Silver, D. (2015). Fictitious self-play in extensive-form games. In *International conference on machine learning* (pp. 805–813).
136. Heinrich, J., & Silver, D. (2016). *Deep reinforcement learning from self-play in imperfect-information games*. arXiv:1603.01121.
137. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In *32nd AAAI conference on artificial intelligence*.
138. Herbrich, R., Minka, T., & Graepel, T. (2007). TrueSkill$^{TM}$: a Bayesian skill rating system. In *Advances in neural information processing systems* (pp. 569–576).
139. Hernandez-Leal, P., & Kaisers, M. (2017). Learning against sequential opponents in repeated stochastic games. In *The 3rd multi-disciplinary conference on reinforcement learning and decision making*. Ann Arbor.
140. Hernandez-Leal, P., & Kaisers, M. (2017). 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, 8–12 May, 2017, Revised selected papers (pp. 239–257).
141. Hernandez-Leal, P., Kaisers, M., Baarslag, T., & Munoz de Cote, E. (2017). *A survey of learning in multiagent environments—dealing with non-stationarity*. arXiv:1707.09183.
142. Hernandez-Leal, P., Kartal, B., & Taylor, M. E. (2019). Agent modeling as auxiliary task for deep reinforcement learning. In *AAAI conference on artificial intelligence and interactive digital entertainment*.
143. Hernandez-Leal, P., Taylor, M. E., Rosman, B., Sucar, L. E., & Munoz de Cote, E. (2016). Identifying and tracking switching, non-stationary opponents: A Bayesian approach. In *Multiagent interaction without prior coordination workshop at AAAI*. Phoenix, AZ, USA.
144. Hernandez-Leal, P., Zhan, Y., Taylor, M. E., Sucar, L. E., & Munoz de Cote, E. (2017). Efficiently detecting switches against non-stationary opponents. *Autonomous Agents and Multi-Agent Systems*, *31*(4), 767–789.
145. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence.*
146. Hinton, G., Vinyals, O., & Dean, J. (2014). Distilling the knowledge in a neural network. In *NIPS deep learning workshop*.
147. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.
148. Hong, Z. W., Su, S. Y., Shann, T. Y., Chang, Y. H., & Lee, C. Y. (2018). A deep policy inference Q-network for multi-agent systems. In *International conference on autonomous agents and multiagent systems*.
149. Hu, J., & Wellman, M. P. (2003). Nash Q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, *4*, 1039–1069.

150. Iba, H. (1996). Emergent cooperation for multiple agents using genetic programming. In *International conference on parallel problem solving from nature* (pp. 32–41). Springer.
151. Ilyas, A., Engstrom, L., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., & Madry, A. (2018). *Are deep policy gradient algorithms truly policy gradient algorithms?* CoRR arXiv:1811.02553.
152. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd international conference on machine learning* (pp. 448–456).
153. Isele, D., & Cosgun, A. (2018). Selective experience replay for lifelong learning. In *Thirty-second AAAI conference on artificial intelligence*.
154. Jaakkola, T., Jordan, M. I., & Singh, S. P. (1994). Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems* (pp. 703–710)
155. Jacobs, R. A., Jordan, M. I., Nowlan, S. J., Hinton, G. E., et al. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.
156. Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., et al. (2019). Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, *364*(6443), 859–865. https://doi.org/10.1126/science.aau6249.
157. Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., & Simonyan, K., et al. (2017). *Population based training of neural networks*. arXiv:1711.09846.
158. Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., & Kavukcuoglu, K. (2017). Reinforcement learning with unsupervised auxiliary tasks. In *International conference on learning representations*.
159. Johanson, M., Bard, N., Burch, N., & Bowling, M. (2012). Finding optimal abstract strategies in extensive-form games. In *Twenty-sixth AAAI conference on artificial intelligence*.
160. Johanson, M., Waugh, K., Bowling, M., & Zinkevich, M. (2011). Accelerating best response calculation in large extensive games. In *Twenty-second international joint conference on artificial intelligence*.
161. Johanson, M., Zinkevich, M. A., & Bowling, M. (2007). Computing robust counter-strategies. In *Advances in neural information processing systems* (pp. 721–728). Vancouver, BC, Canada.
162. Johnson, M., Hofmann, K., Hutton, T., & Bignell, D. (2016). The Malmo platform for artificial intelligence experimentation. In *IJCAI* (pp. 4246–4247).
163. Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M., & Lange, D. (2018). *Unity: A general platform for intelligent agents*. CoRR arXiv:1809.02627.
164. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.
165. Kaisers, M., & Tuyls, K. (2011). FAQ-learning in matrix games: demonstrating convergence near Nash equilibria, and bifurcation of attractors in the battle of sexes. In *AAAI Workshop on Interactive Decision Theory and Game Theory* (pp. 309–316). San Francisco, CA, USA.
166. Kakade, S. M. (2002). A natural policy gradient. In *Advances in neural information processing systems* (pp. 1531–1538).
167. Kalai, E., & Lehrer, E. (1993). Rational learning leads to Nash equilibrium. *Econometrica: Journal of the Econometric Society*, *61*, 1019–1045.
168. Kamihigashi, T., & Le Van, C. (2015). *Necessary and sufficient conditions for a solution of the bellman equation to be the value function: A general principle*. https://halshs.archives-ouvertes.fr/halshs-01159177
169. Kartal, B., Godoy, J., Karamouzas, I., & Guy, S. J. (2015). Stochastic tree search with useful cycles for patrolling problems. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 1289–1294). IEEE.
170. Kartal, B., Hernandez-Leal, P., & Taylor, M. E. (2019). Using Monte Carlo tree search as a demonstrator within asynchronous deep RL. In *AAAI workshop on reinforcement learning in games*.
171. Kartal, B., Nunes, E., Godoy, J., & Gini, M. (2016). Monte Carlo tree search with branch and bound for multi-robot task allocation. In *The IJCAI-16 workshop on autonomous mobile service robots*.
172. Khadka, S., Majumdar, S., & Tumer, K. (2019). *Evolutionary reinforcement learning for sample-efficient multiagent coordination*. arXiv e-prints arXiv:1906.07315.
173. Kim, W., Cho, M., & Sung, Y. (2019). Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In *33rd AAAI conference on artificial intelligence*.
174. Kok, J. R., & Vlassis, N. (2004). Sparse cooperative Q-learning. In *Proceedings of the twenty-first international conference on Machine learning* (p. 61). ACM.
175. Konda, V. R., & Tsitsiklis, J. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*.

176. Konidaris, G., & Barto, A. (2006). Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on machine learning* (pp. 489–496). ACM.
177. Kretchmar, R. M., & Anderson, C. W. (1997). Comparison of CMACs and radial basis functions for local function approximators in reinforcement learning. In *Proceedings of international conference on neural networks (ICNN'97)* (Vol. 2, pp. 834–837). IEEE.
178. Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems* (pp. 3675–3683).
179. Lake, B. M., Ullman, T. D., Tenenbaum, J., & Gershman, S. (2016). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*, 1–72.
180. Lanctot, M., Zambaldi, V. F., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., & Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in neural information processing systems*.
181. Lauer, M., & Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the seventeenth international conference on machine learning*.
182. Laurent, G. J., Matignon, L., Fort-Piat, L., et al. (2011). The world of independent learners is not Markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems*, *15*(1), 55–64.
183. Lazaridou, A., Peysakhovich, A., & Baroni, M. (2017). Multi-agent cooperation and the emergence of (natural) language. In *International conference on learning representations*.
184. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.
185. Lehman, J., & Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE* (pp. 329–336).
186. Leibo, J. Z., Hughes, E., Lanctot, M., & Graepel, T. (2019). *Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research*. CoRR arXiv:1903.00742.
187. Leibo, J. Z., Perolat, J., Hughes, E., Wheelwright, S., Marblestone, A. H., Duéñez-Guzmán, E., Sunehag, P., Dunning, I., & Graepel, T. (2019). Malthusian reinforcement learning. In *18th international conference on autonomous agents and multiagent systems*.
188. Leibo, J. Z., Zambaldi, V., Lanctot, M., & Marecki, J. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th conference on autonomous agents and multiagent systems*. Sao Paulo.
189. Lerer, A., & Peysakhovich, A. (2017). *Maintaining cooperation in complex social dilemmas using deep reinforcement learning*. CoRR arXiv:1707.01068.
190. Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., & Russell, S. (2019). Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *AAAI conference on artificial intelligence*.
191. Li, Y. (2017). *Deep reinforcement learning: An overview*. CoRR arXiv:1701.07274.
192. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *International conference on learning representations*.
193. Lin, L. J. (1991). Programming robots using reinforcement learning and teaching. In *AAAI* (pp. 781–786).
194. Lin, L. J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*(3–4), 293–321.
195. Ling, C. K., Fang, F., & Kolter, J. Z. (2018). What game are we playing? End-to-end learning in normal and extensive form games. In *Twenty-seventh international joint conference on artificial intelligence*.
196. Lipton, Z. C., Azizzadenesheli, K., Kumar, A., Li, L., Gao, J., & Deng, L. (2018). *Combating reinforcement learning's Sisyphean curse with intrinsic fear*. arXiv:1611.01211v8.
197. Lipton, Z. C., & Steinhardt, J. (2018). Troubling trends in machine learning scholarship. In *ICML Machine Learning Debates workshop*.
198. Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th international conference on machine learning* (pp. 157–163). New Brunswick, NJ, USA.
199. Littman, M. L. (2001). Friend-or-foe Q-learning in general-sum games. In *Proceedings of 17th international conference on autonomous agents and multiagent systems* (pp. 322–328). Williamstown, MA, USA.
200. Littman, M. L. (2001). Value-function reinforcement learning in Markov games. *Cognitive Systems Research*, *2*(1), 55–66.
201. Littman, M. L., & Stone, P. (2001). Implicit negotiation in repeated games. In *ATAL '01: revised papers from the 8th international workshop on intelligent agents VIII*.
202. Liu, H., Feng, Y., Mao, Y., Zhou, D., Peng, J., & Liu, Q. (2018). Action-depedent control variates for policy optimization via stein's identity. In *International conference on learning representations*.

203. Liu, S., Lever, G., Merel, J., Tunyasuvunakool, S., Heess, N., & Graepel, T. (2019). Emergent coordination through competition. In *International conference on learning representations*.
204. Lockhart, E., Lanctot, M., Pérolat, J., Lespiau, J., Morrill, D., Timbers, F., & Tuyls, K. (2019). *Computing approximate equilibria in sequential adversarial games by exploitability descent*. CoRR arXiv:1903.05614.
205. Lowe, R., Foerster, J., Boureau, Y. L., Pineau, J., & Dauphin, Y. (2019). On the pitfalls of measuring emergent communication. In *18th international conference on autonomous agents and multiagent systems*.
206. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems* (pp. 6379–6390).
207. Lu, T., Schuurmans, D., & Boutilier, C. (2018). Non-delusional Q-learning and value-iteration. In *Advances in neural information processing systems* (pp. 9949–9959).
208. Lyle, C., Castro, P. S., & Bellemare, M. G. (2019). A comparative analysis of expected and distributional reinforcement learning. In *Thirty-third AAAI conference on artificial intelligence*.
209. Multiagent Learning, Foundations and Recent Trends. (2017). [Online]. Retrieved September 7, 2018, https://www.cs.utexas.edu/~larg/ijcai17_tutorial/multiagent_learning.pdf.
210. Maaten, Lvd, & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605.
211. Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., & Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, *61*, 523–562.
212. Mahadevan, S., & Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, *55*(2–3), 311–365.
213. Matignon, L., Laurent, G. J., & Le Fort-Piat, N. (2012). Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems. *Knowledge Engineering Review*, *27*(1), 1–31.
214. McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (Ed.), *Psychology of learning and motivation* (Vol. 24, pp. 109–165). Amsterdam: Elsevier.
215. McCracken, P., & Bowling, M. (2004) Safe strategies for agent modelling in games. In *AAAI fall symposium* (pp. 103–110).
216. Melis, G., Dyer, C., & Blunsom, P. (2018). On the state of the art of evaluation in neural language models. In *International conference on learning representations*.
217. Melo, F. S., Meyn, S. P., & Ribeiro, M. I. (2008). An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th international conference on Machine learning* (pp. 664–671). ACM.
218. Meuleau, N., Peshkin, L., Kim, K. E., & Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the fifteenth conference on uncertainty in artificial intelligence* (pp. 427–436).
219. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).
220. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing atari with deep reinforcement learning*. arXiv:1312.5602v1.
221. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.
222. Monderer, D., & Shapley, L. S. (1996). Fictitious play property for games with identical interests. *Journal of Economic Theory*, *68*(1), 258–265.
223. Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, *13*(1), 103–130.
224. Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., et al. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, *356*(6337), 508–513.
225. Mordatch, I., & Abbeel, P. (2018). Emergence of grounded compositional language in multi-agent populations. In *Thirty-second AAAI conference on artificial intelligence*.
226. Moriarty, D. E., Schultz, A. C., & Grefenstette, J. J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, *11*, 241–276.
227. Morimoto, J., & Doya, K. (2005). Robust reinforcement learning. *Neural Computation*, *17*(2), 335–359.
228. Nagarajan, P., Warnell, G., & Stone, P. (2018). *Deterministic implementations for reproducibility in deep reinforcement learning*. arXiv:1809.05676

229. Nash, J. F. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, *36*(1), 48–49.
230. Neller, T. W., & Lanctot, M. (2013). An introduction to counterfactual regret minimization. In *Proceedings of model AI assignments, the fourth symposium on educational advances in artificial intelligence (EAAI-2013)*.
231. Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the sixteenth international conference on machine learning* (pp. 278–287).
232. Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2018). *Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications*. arXiv preprint arXiv:1812.11794.
233. Nowé, A., Vrancx, P., & De Hauwere, Y. M. (2012). Game theory and multi-agent reinforcement learning. In M. Wiering & M. van Otterlo (Eds.), *Reinforcement learning* (pp. 441–470). Berlin: Springer.
234. OpenAI Baselines: ACKTR & A2C. (2017). [Online]. Retrieved April 29, 2019, https://openai.com/blog/baselines-acktr-a2c/.
235. Open AI Five. (2018). [Online]. Retrieved September 7, 2018, https://blog.openai.com/openai-five.
236. Oliehoek, F. A. (2018). Interactive learning and decision making - foundations, insights & challenges. In *International joint conference on artificial intelligence*.
237. Oliehoek, F. A., Amato, C., et al. (2016). *A concise introduction to decentralized POMDPs*. Berlin: Springer.
238. Oliehoek, F. A., De Jong, E. D., & Vlassis, N. (2006). The parallel Nash memory for asymmetric games. In *Proceedings of the 8th annual conference on genetic and evolutionary computation* (pp. 337–344). ACM.
239. Oliehoek, F. A., Spaan, M. T., & Vlassis, N. (2008). Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, *32*, 289–353.
240. Oliehoek, F. A., Whiteson, S., & Spaan, M. T. (2013). Approximate solutions for factored Dec-POMDPs with many agents. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 563–570). International Foundation for Autonomous Agents and Multiagent Systems.
241. Oliehoek, F. A., Witwicki, S. J., & Kaelbling, L. P. (2012). Influence-based abstraction for multiagent systems. In *Twenty-sixth AAAI conference on artificial intelligence.*
242. Omidshafiei, S., Hennes, D., Morrill, D., Munos, R., Perolat, J., Lanctot, M., Gruslys, A., Lespiau, J. B., & Tuyls, K. (2019). *Neural replicator dynamics*. arXiv e-prints arXiv:1906.00190.
243. Omidshafiei, S., Papadimitriou, C., Piliouras, G., Tuyls, K., Rowland, M., Lespiau, J. B., et al. (2019). $\alpha$-rank: Multi-agent evaluation by evolution. *Scientific Reports*, *9*, 9937.
244. Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th international conference on machine learning*. Sydney.
245. Ortega, P. A., & Legg, S. (2018). *Modeling friends and foes*. arXiv:1807.00196
246. Palmer, G., Savani, R., & Tuyls, K. (2019). Negative update intervals in deep multi-agent reinforcement learning. In *18th International conference on autonomous agents and multiagent systems*.
247. Palmer, G., Tuyls, K., Bloembergen, D., & Savani, R. (2018). Lenient multi-agent deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*.
248. Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, *11*(3), 387–434.
249. Panait, L., Sullivan, K., & Luke, S. (2006). Lenience towards teammates helps in cooperative multiagent learning. In *Proceedings of the 5th international conference on autonomous agents and multiagent systems*. Hakodate, Japan.
250. Panait, L., Tuyls, K., & Luke, S. (2008). Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *JMLR*, *9*(Mar), 423–457.
251. Papoudakis, G., Christianos, F., Rahman, A., & Albrecht, S. V. (2019). *Dealing with non-stationarity in multi-agent deep reinforcement learning*. arXiv preprint arXiv:1906.04737.
252. Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310–1318).
253. Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., & Wang, J. (2017). *Multiagent bidirectionally-coordinated nets for learning to play StarCraft combat games*. arXiv:1703.10069
254. Pérez-Liébana, D., Hofmann, K., Mohanty, S. P., Kuno, N., Kramer, A., Devlin, S., Gaina, R. D., & Ionita, D. (2019). *The multi-agent reinforcement learning in Malmö (MARLÖ) competition*. CoRR arXiv:1901.08129.
255. Pérolat, J., Piot, B., & Pietquin, O. (2018). Actor-critic fictitious play in simultaneous move multistage games. In *21st international conference on artificial intelligence and statistics*.

256. Pesce, E., & Montana, G. (2019). *Improving coordination in multi-agent deep reinforcement learning through memory-driven communication*. CoRR arXiv:1901.03887.
257. Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. In *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 2817–2826). JMLR. org
258. Powers, R., & Shoham, Y. (2005). Learning against opponents with bounded memory. *In Proceedings of the 19th international joint conference on artificial intelligence* (pp. 817–822). Edinburg, Scotland, UK.
259. Powers, R., Shoham, Y., & Vu, T. (2007). A general criterion and an algorithmic framework for learning in multi-agent systems. *Machine Learning*, *67*(1–2), 45–76.
260. Precup, D., Sutton, R. S., & Singh, S. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the seventeenth international conference on machine learning*.
261. Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: Wiley.
262. Pyeatt, L. D., Howe, A. E., et al. (2001). Decision tree function approximation in reinforcement learning. In *Proceedings of the third international symposium on adaptive systems: Evolutionary computation and probabilistic graphical models* (Vol. 2, pp. 70–77). Cuba.
263. Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S. M. A., & Botvinick, M. (2018). Machine theory of mind. In *International conference on machine learning*. Stockholm, Sweden.
264. Raghu, M., Irpan, A., Andreas, J., Kleinberg, R., Le, Q., & Kleinberg, J. (2018). Can deep reinforcement learning solve Erdos–Selfridge-spencer games? In *Proceedings of the 35th international conference on machine learning*.
265. Raileanu, R., Denton, E., Szlam, A., & Fergus, R. (2018). Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*.
266. Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J. N., & Whiteson, S. (2018). QMIX - monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*.
267. Resnick, C., Eldridge, W., Ha, D., Britz, D., Foerster, J., Togelius, J., Cho, K., & Bruna, J. (2018). *Pommerman: A multi-agent playground*. arXiv:1809.07124.
268. Riedmiller, M. (2005). Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning* (pp. 317–328). Springer.
269. Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., & Tesauro, G. (2018). *Learning to learn without forgetting by maximizing transfer and minimizing interference*. CoRR arXiv:1810.11910.
270. Rosenthal, R. (1979). The file drawer problem and tolerance for null results. *Psychological Bulletin*, *86*(3), 638.
271. Rosin, C. D., & Belew, R. K. (1997). New methods for competitive coevolution. *Evolutionary Computation*, *5*(1), 1–29.
272. Rosman, B., Hawasly, M., & Ramamoorthy, S. (2016). Bayesian policy reuse. *Machine Learning*, *104*(1), 99–127.
273. Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., & Hadsell, R. (2016). Policy distillation. In *International conference on learning representations*.
274. Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems* (pp. 901–909).
275. Samothrakis, S., Lucas, S., Runarsson, T., & Robles, D. (2013). Coevolving game-playing agents: Measuring performance and intransitivities. *IEEE Transactions on Evolutionary Computation*, *17*(2), 213–226.
276. Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C., Torr, P. H. S., Foerster, J. N., & Whiteson, S. (2019). *The StarCraft multi-agent challenge*. CoRR arXiv:1902.04043.
277. Sandholm, T. W., & Crites, R. H. (1996). Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, *37*(1–2), 147–166.
278. Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. In *International conference on learning representations*.
279. Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the international conference on simulation of adaptive behavior: From animals to animats* (pp. 222–227).
280. Schmidhuber, J. (2015). *Critique of Paper by "Deep Learning Conspiracy"* (Nature 521 p 436). http://people.idsia.ch/~juergen/deep-learning-conspiracy.html.
281. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

282. Schulman, J., Abbeel, P., & Chen, X. (2017) *Equivalence between policy gradients and soft Q-learning*. CoRR arXiv:1704.06440.
283. Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., & Moritz, P. (2015). Trust region policy optimization. In *31st international conference on machine learning*. Lille, France.
284. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms*. arXiv:1707.06347.
285. Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.
286. Sculley, D., Snoek, J., Wiltschko, A., & Rahimi, A. (2018). Winner's curse? On pace, progress, and empirical rigor. In *ICLR workshop*.
287. Shamma, J. S., & Arslan, G. (2005). Dynamic fictitious play, dynamic gradient play, and distributed convergence to Nash equilibria. *IEEE Transactions on Automatic Control*, *50*(3), 312–327.
288. Shelhamer, E., Mahmoudieh, P., Argus, M., & Darrell, T. (2017). Loss is its own reward: Self-supervision for reinforcement learning. In *ICLR workshops*.
289. Shoham, Y., Powers, R., & Grenager, T. (2007). If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, *171*(7), 365–377.
290. Silva, F. L., & Costa, A. H. R. (2019). A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, *64*, 645–703.
291. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.
292. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.
293. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354.
294. Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, *38*(3), 287–308.
295. Singh, S., Kearns, M., & Mansour, Y. (2000). Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the sixteenth conference on uncertainty in artificial intelligence* (pp. 541–548). Morgan Kaufmann Publishers Inc.
296. Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, *8*(3–4), 323–339.
297. Song, X., Wang, T., & Zhang, C. (2019). Convergence of multi-agent learning with a finite step size in general-sum games. In *18th International conference on autonomous agents and multiagent systems*.
298. Song, Y., Wang, J., Lukasiewicz, T., Xu, Z., Xu, M., Ding, Z., & Wu, L. (2019). *Arena: A general evaluation platform and building toolkit for multi-agent intelligence*. CoRR arXiv:1905.08085.
299. Spencer, J. (1994). Randomization, derandomization and antirandomization: three games. *Theoretical Computer Science*, *131*(2), 415–429.
300. Srinivasan, S., Lanctot, M., Zambaldi, V., Pérolat, J., Tuyls, K., Munos, R., & Bowling, M. (2018). Actor-critic policy optimization in partially observable multiagent environments. In *Advances in neural information processing systems* (pp. 3422–3435).
301. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.
302. Steckelmacher, D., Roijers, D. M., Harutyunyan, A., Vrancx, P., Plisnier, H., & Nowé, A. (2018). Reinforcement learning in pomdps with memoryless options and option-observation initiation sets. In *Thirty-second AAAI conference on artificial intelligence*.
303. Stimpson, J. L., & Goodrich, M. A. (2003). Learning to cooperate in a social dilemma: A satisficing approach to bargaining. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (pp. 728–735).
304. Stone, P., Kaminka, G., Kraus, S., & Rosenschein, J. S. (2010). Ad Hoc autonomous agent teams: Collaboration without pre-coordination. In *32nd AAAI conference on artificial intelligence* (pp. 1504–1509). Atlanta, Georgia, USA.
305. Stone, P., & Veloso, M. M. (2000). Multiagent systems - a survey from a machine learning perspective. *Autonomous Robots*, *8*(3), 345–383.
306. Stooke, A., & Abbeel, P. (2018). *Accelerated methods for deep reinforcement learning*. CoRR arXiv:1803.02811.
307. Strehl, A. L., & Littman, M. L. (2008). An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, *74*(8), 1309–1331.
308. Suarez, J., Du, Y., Isola, P., & Mordatch, I. (2019). *Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents*. CoRR arXiv:1903.00784.

309. Suau de Castro, M., Congeduti, E., Starre, R. A., Czechowski, A., & Oliehoek, F. A. (2019). Influence-based abstraction in deep reinforcement learning. In *Adaptive, learning agents workshop*.
310. Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., & Clune, J. (2017). *Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning*. CoRR arXiv:1712.06567.
311. Suddarth, S. C., & Kergosien, Y. (1990). Rule-injection hints as a means of improving network performance and learning time. In *Neural networks* (pp. 120–129). Springer.
312. Sukhbaatar, S., Szlam, A., & Fergus, R. (2016). Learning multiagent communication with backpropagation. In *Advances in neural information processing systems* (pp. 2244–2252).
313. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V. F., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., & Graepel, T. (2018). 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.
314. Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems* (pp. 1038–1044).
315. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). Cambridge: MIT Press.
316. Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*.
317. Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., & Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th international conference on autonomous agents and multiagent systems* (Vol. 2, pp. 761–768). International Foundation for Autonomous Agents and Multiagent Systems.
318. Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *4*(1), 1–103.
319. Szepesvári, C., & Littman, M. L. (1999). A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Computation*, *11*(8), 2017–2060.
320. Tamar, A., Levine, S., Abbeel, P., Wu, Y., & Thomas, G. (2016). Value iteration networks. In *NIPS* (pp. 2154–2162).
321. Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, *7*, 83–124.
322. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., et al. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE*, *12*(4), e0172395.
323. Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Machine learning proceedings 1993 proceedings of the tenth international conference*, University of Massachusetts, Amherst, 27–29 June, 1993 (pp. 330–337).
324. Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, *10*, 1633–1685.
325. Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*(3), 58–68.
326. Tesauro, G. (2003). Extending Q-learning to general adaptive multi-agent systems. In *Advances in neural information processing systems* (pp. 871–878). Vancouver, Canada.
327. Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo - A physics engine for model-based control. In *Intelligent robots and systems*( pp. 5026–5033).
328. Torrado, R. R., Bontrager, P., Togelius, J., Liu, J., & Perez-Liebana, D. (2018). *Deep reinforcement learning for general video game AI*. arXiv:1806.02448
329. Tsitsiklis, J. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, *16*(3), 185–202.
330. Tsitsiklis, J. N., & Van Roy, B. (1997). Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems* (pp. 1075–1081).
331. Tucker, G., Bhupatiraju, S., Gu, S., Turner, R. E., Ghahramani, Z., & Levine, S. (2018). The mirage of action-dependent baselines in reinforcement learning. In *International conference on machine learning*.
332. Tumer, K., & Agogino, A. (2007). Distributed agent-based air traffic flow management. In *Proceedings of the 6th international conference on autonomous agents and multiagent systems*. Honolulu, Hawaii.
333. Tuyls, K., & Weiss, G. (2012). Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, *33*(3), 41–52.
334. van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., & Modayil, J. (2018). *Deep reinforcement learning and the deadly triad*. CoRR arXiv:1812.02648.
335. Van der Pol, E., & Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. In *Proceedings of learning, inference and control of multi-agent systems at NIPS*.

336. Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *Thirtieth AAAI conference on artificial intelligence*.
337. Van Seijen, H., Van Hasselt, H., Whiteson, S., & Wiering, M. (2009). A theoretical and empirical analysis of Expected Sarsa. *In IEEE symposium on adaptive dynamic programming and reinforcement learning* (pp. 177–184). Nashville, TN, USA.
338. Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., & Kavukcuoglu, K. (2017). FeUdal networks for hierarchical reinforcement learning. In *International conference on machine learning*.
339. Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., & Silver, D. (2019). AlphaStar: Mastering the real-time strategy game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/
340. Vodopivec, T., Samothrakis, S., & Ster, B. (2017). On Monte Carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, *60*, 881–936.
341. Von Neumann, J., & Morgenstern, O. (1945). *Theory of games and economic behavior* (Vol. 51). New York: Bulletin of the American Mathematical Society.
342. Walsh, W. E., Das, R., Tesauro, G., & Kephart, J. O. (2002). Analyzing complex strategic interactions in multi-agent systems. In *AAAI-02 workshop on game-theoretic and decision-theoretic agents* (pp. 109–118).
343. Wang, H., Raj, B., & Xing, E. P. (2017). *On the origin of deep learning*. CoRR arXiv:1702.07800.
344. Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & de Freitas, N. (2016). *Sample efficient actor-critic with experience replay*. arXiv preprint arXiv:1611.01224.
345. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*.
346. Watkins, J. (1989). Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge, UK
347. Wei, E., & Luke, S. (2016). Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research*, *17*, 1–42.
348. Wei, E., Wicke, D., Freelan, D., & Luke, S. (2018). *Multiagent soft Q-learning*. arXiv:1804.09817
349. Weinberg, M., & Rosenschein, J. S. (2004). Best-response multiagent learning in non-stationary environments. In *Proceedings of the 3rd international conference on autonomous agents and multiagent systems* (pp. 506–513). New York, NY, USA.
350. Weiss, G. (Ed.). (2013). *Multiagent systems. Intelligent robotics and autonomous agents series* (2nd ed.). Cambridge, MA: MIT Press.
351. Whiteson, S., & Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, *7*(May), 877–917.
352. Whiteson, S., Tanner, B., Taylor, M. E., & Stone, P. (2011). Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)* (pp. 120–127). IEEE.
353. Wiering, M., & van Otterlo, M. (Eds.) (2012). *Reinforcement learning*. Adaptation, learning, and optimization (Vol. 12). Springer-Verlag Berlin Heidelberg.
354. Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3–4), 229–256.
355. Wolpert, D. H., & Tumer, K. (2002). Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems* (pp. 355–369).
356. Wolpert, D. H., Wheeler, K. R., & Tumer, K. (1999). General principles of learning-based multi-agent systems. In *Proceedings of the third international conference on autonomous agents*.
357. Wunder, M., Littman, M. L., & Babes, M. (2010). Classes of multiagent Q-learning dynamics with epsilon-greedy exploration. In *Proceedings of the 35th international conference on machine learning* (pp. 1167–1174). Haifa, Israel.
358. Yang, T., Hao, J., Meng, Z., Zhang, C., & Zheng, Y. Z. Z. (2019). Towards efficient detection and optimal response against sophisticated opponents. In *IJCAI*.
359. Yang, Y., Hao, J., Sun, M., Wang, Z., Fan, C., & Strbac, G. (2018). 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.
360. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., & Wang, J. (2018). Mean field multi-agent reinforcement learning. In *Proceedings of the 35th international conference on machine learning*. Stockholm Sweden.
361. Yu, Y. (2018). Towards sample efficient reinforcement learning. In *IJCAI* (pp. 5739–5743).

362. Zahavy, T., Ben-Zrihem, N., & Mannor, S. (2016). Graying the black box: Understanding DQNs. In *International conference on machine learning* (pp. 1899–1908).
363. Zhang, C., & Lesser, V. (2010). Multi-agent learning with policy prediction. In *Twenty-fourth AAAI conference on artificial intelligence*.
364. Zhao, J., Qiu, G., Guan, Z., Zhao, W., & He, X. (2018). Deep reinforcement learning for sponsored search real-time bidding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1021–1030). ACM.
365. Zheng, Y., Hao, J., & Zhang, Z. (2018). *Weighted double deep multiagent reinforcement learning in stochastic cooperative environments*. arXiv:1802.08534.
366. Zheng, Y., Meng, Z., Hao, J., Zhang, Z., Yang, T., & Fan, C. (2018). A deep bayesian policy reuse approach against non-stationary agents. In *Advances in Neural Information Processing Systems* (pp. 962–972).
367. Zinkevich, M., Greenwald, A., & Littman, M. L. (2006). Cyclic equilibria in Markov games. In *Advances in neural information processing systems* (pp. 1641–1648).
368. Zinkevich, M., Johanson, M., Bowling, M., & Piccione, C. (2008). Regret minimization in games with incomplete information. In *Advances in neural information processing systems* (pp. 1729–1736).