Remember and Forget Experience Replay for Multi-Agent Reinforcement Learning

Pascal Weber, Daniel Wälchli, Mustafa Zeqiri, and Petros Koumoutsakos

Computational Science and Engineering Laboratory
ETH Zürich, Switzerland
and
John A. Paulson School of Engineering and Applied Sciences
Harvard University, USA

Abstract

We present the extension of the Remember and Forget for Experience Replay (ReF-ER) algorithm to Multi-Agent Reinforcement Learning (MARL). ReF-ER was shown to outperform state of the art algorithms for continuous control in problems ranging from the OpenAI Gym to complex fluid flows. In MARL, the dependencies between the agents are included in the state-value estimator and the environment dynamics are modeled via the importance weights used by ReF-ER. In collaborative environments, we find the best performance when the value is estimated using individual rewards and we ignore the effects of other actions on the transition map. We benchmark the performance of ReF-ER MARL on the Stanford Intelligent Systems Laboratory (SISL) environments. We find that employing a single feed-forward neural network for the policy and the value function in ReF-ER MARL, outperforms state of the art algorithms that rely on complex neural network architectures.

1 Introduction

State of the art deep reinforcement learning (RL) algorithms approximate the optimal value function and policy using deep neural networks. This approach has been showcased in the playing of Atari games [17], board games like Shogi, Chess, and Go [27]. More recently Multi-Agent Reinforcement Learning (MARL) has been applied to multiplayer games such as poker [2] and prominent computer games such as Dota 2 and StarCraftII [22, 35]. Tasks that require multiple agents to collaborate often rely on a generalization of single agent RL algorithms and employ complex neural network architectures for the value estimation. Deep MARL presents several algorithmic challenges. Learning individual policies is hard in environments with multiple learning agents that can be non-stationary. Moreover, the inclusion of collaborative behaviour by an averaging of the rewards yields a credit assignment problem, hindering the reinforcement of beneficial behaviour. Finally, most games are restricted to a discrete action space. More recently the extension of MARL to scientific computing and complex systems (scientific multi-agent reinforcement learning (SciMARL) [20, 31]) has showcase the importance of control in continuous, high-dimensional action spaces. To the best of our knowledge, generalizations of well-established algorithms for continuous action RL is limited [8, 33].

We address these challenges by revisiting the relationships and interactions between multiple agents. We follow the centralized training with decentralized execution paradigm [21, 13] (CTDE). Agents learn using the experiences from all their peers, while at execution time, the learned policy is used to make decisions based on a single agent’s state information. The inclusion of the observations from all agents addresses the non-stationary issues, and the adoption of ReF-ER handles effectively far-policy
experiences \cite{19}. Furthermore, the learning rule can be modified to systematically examine the credit assignment problem. Lastly, ReF-ER allows to model the interaction strength between agents via the importance weight.

2 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) aims to find the optimal policy in a Multi-Agent Markov Decision Process (MAMDP). A MAMDP is a tuple \((\mathcal{S}, \mathcal{A}, \mathcal{r}, \mathcal{N}, \mathcal{D})\) consisting of the states \(s \in \mathcal{S}^N\), actions \(a \in \mathcal{A}^N\), and rewards \(r \in \mathcal{R}^N\) observed by the \(N \in \mathbb{N}_+\) agents in the environment. Here we consider a homogeneous setting, where all agents have the same state space, action space, and reward function. The agents’ individual states, actions, and rewards are denoted by \(s^{(i)} \in \mathcal{S}, a^{(i)} \in \mathcal{A}\), and \(r^{(i)} \in \mathcal{R}\) for \(i = 1, \ldots, N\). The transition map is given by \(r, s' = D(s, a)\), i.e. it depends on the states and actions from all agents. This is contrary to environments in which agents take actions sequentially and the state is updated after each action. In MAMDP we specify the initial state distribution \(p(s)\) and the stochastic policy \(\pi(a|s)\). The later is the probability distribution over the actions of the agents given the states. We assume that the policy factorizes as

\[
\pi(a|s) = \prod_{i=1}^{N} \pi(a^{(i)}|s^{(i)}).
\]

This allows for decentralized execution, where the agent samples an action solely based on its state. The vector-valued state-action-value function and state-value are defined as

\[
Q^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right], \quad V^\pi(s) = \mathbb{E}_\pi \left[ Q^\pi(s, a) \right].
\]

where \(\gamma \in [0, 1)\) is the discount factor and \(r_t = r(s_t, a_t)\). Assuming equal importance for all agents yields the scalar state-value that is expressed in terms of the individual state value functions \(V^\pi(s^{(i)})\)

\[
P[V^\pi(s)] = \frac{1}{N} \sum_{i=1}^{N} V^\pi(s^{(i)}).
\]

The optimal policy \(\pi^*\) in the MAMDP maximizes the average of the state-values for all agents

\[
\pi^* = \arg \max_{\pi} \frac{1}{N} \sum_{i=1}^{N} V^\pi(s^{(i)}), \quad \forall s \in \mathcal{S}^N.
\]

In MARL the agents interact with the environment as depicted in fig. 1. At timestep \(t\), the agents take actions \(a_t\) based on the states \(s_t\). Thereby the environment transitions into new states \(s_{t+1}\) and returns rewards \(r_t\). An experience, consists of states, actions, and rewards \((s_t, a_t, r_t)\). Upon termination of the episode \(E_k = \{(s_t, a_t, r_t)\}_{t=1}^{T_k}\), the experiences are stored in the replay memory \(\mathcal{R}M = \{E_k\}_{k=1}^{K}\). In the following \(d\) denotes the distribution of the experiences in the replay memory.

We extend V-RACER with ReF-ER \cite{19} for MARL, training a neural network that approximates the state-value and the policy parameters. Using the experiences in the replay memory, we train the neural network with Adam \cite{12}. In the following, \(\pi_\omega(a|s)\) and \(V^\pi_\vartheta(s)\) denote the policy and state-value function with the respective weights \(\omega\) and \(\vartheta\). Note, that the parameters \(\omega\) and \(\vartheta\) only differ in the output layer.
2.1 Value Learning

The relationships between the agents are included in the learning process by introducing a scalarization function \( f : \mathbb{R}^N \rightarrow \mathbb{R} \) in the on-policy returns estimator Retrace \([13]\)

\[
Q_{t,f}^{\text{ret}} = f(r_t) + \gamma \bar{V}^\pi_t(s_t) + \gamma \bar{V}^\pi_{t+1,f} - Q_f^\pi(s_t, a_t).
\] \hspace{1cm} (5)

V-RACER approximates \( Q^\pi_t(s_t, a_t) \approx \bar{V}^\pi_t(s_t) \) and hence the V-trace estimator \([36]\) becomes

\[
\hat{V}^{\text{the}}_{t,f} = \bar{V}^\pi_t(s_t) + \rho_t \left[ f(r_t) + \gamma \hat{V}^{\text{the}}_{t+1,f} - \bar{V}^\pi_t(s_t) \right],
\] \hspace{1cm} (6)

and it can be related to the on-policy returns estimator Retrace via

\[
Q_{t,f}^{\text{ret}} = f(r_t) + \gamma \hat{V}^{\text{the}}_{t,f}.
\] \hspace{1cm} (7)

The truncated importance weight \( \bar{\rho}_t \) will be defined in the following section. It balances the impact from off-policy data on the current estimator of the state-value.

For the function \( f \) we distinguish two cases. The first case, which we refer to as \textit{individual}, assumes that the value estimate depends solely on the reward observed by agent \( i \)

\[
f^{(i)}_{\text{individual}}(r) = r^{(i)}, \quad V^\pi_{\text{individual},(i)}(s) = V^\pi(s^{(i)}).
\] \hspace{1cm} (8)

In the second case, referred to as \textit{cooperative}, the reward and the state-values are averaged

\[
f^{(i)}_{\text{cooperative}}(r) = \frac{1}{N} \sum_{i=1}^{N} r^{(i)}, \quad V^\pi_{\text{cooperative}}(s) = \frac{1}{N} \sum_{i=1}^{N} V^\pi(s^{(i)}).
\] \hspace{1cm} (9)

The weights \( \theta \) of the neural network \( V^\pi_{\theta}(s) \) are updated to minimize the loss

\[
\mathcal{L}(\theta) = \mathbb{E}_d \left[ \frac{1}{N} \sum_{i=1}^{N} \left( V^\pi_{\theta}(s^{(i)}_t) - \hat{V}^{\text{the}}_{t,f} \right)^2 \right].
\] \hspace{1cm} (10)

Note, that both variants reduce to the original loss proposed in \([19]\) when assuming a single agent.

2.2 Policy Gradient

Given the definition of \( Q^{\text{ret}}_{t,f} \), we learn the weights \( \omega \) of the policy \( \pi_{\omega}(s|a) \) by maximizing the expected advantage

\[
\mathcal{L}(\omega) = \mathbb{E}_d \left[ \frac{1}{N} \sum_{i=1}^{N} \rho_t(\omega) \left( Q^{\text{ret}}_{t,f} - V^\pi(s^{(i)}_t) \right) \right].
\] \hspace{1cm} (11)

Taking the gradient with respect to the weights \( \omega \) yields the off-policy gradient \([4]\). The importance weights \( \rho_t(\omega) \) reflect the assumed dynamics of the environment.

Adopting the \textit{full dynamics} \( r^{t+1}, s^{t+1} = D(s^t, a^t) \) and the factorization of the policy from eq. \((1)\) yields the importance weight

\[
\rho_t^N(\omega) = \prod_{i=1}^{N} \frac{\pi_{\omega}(a^{(i)}_t|s^{(i)}_t)}{\pi(a^{(i)}_t|s^{(i)}_t)}.
\] \hspace{1cm} (12)

Here the denominator denotes the policy that was used when \( s^{(i)} \) was sampled. When using this importance weight the value estimate and the policy update depend on the probability of the action of all agents.

In cases where interactions between the agents are negligible, the value estimate and the policy update should not be influenced by the probabilities of the other agents. This is achieved by using the importance weight

\[
\rho_t(\omega) = \frac{\pi_{\omega}(a^{(i)}_t|s^{(i)}_t)}{\pi(a^{(i)}_t|s^{(i)}_t)}.
\] \hspace{1cm} (13)

We denote it as the \textit{local dynamics model}. 

3
where the experiences from all agents are used during training. This study was extended by Terry while considering that in the next update the agents will adjust their behaviour. For continuous

The factor \( \beta \) is adjusted according to the fraction of off-policy samples \( n_{\text{off-policy}} \) in the replay memory

\[
\beta \leftarrow \begin{cases} 
(1 - \eta)\beta & \text{if } n_{\text{off-policy}} > n^* \\
(1 - \eta)\beta + \eta, & \text{otherwise.}
\end{cases}
\]

Here \( \eta \) denotes the learning rate used by Adam. This allows maintaining a target fraction \( n^* \) of off-policy experiences in the replay memory. The regularizer is the gradient of the Kullback-Leibler divergence between the current and the past policy \( \{\pi_k\} \)

\[
\hat{g}^\text{K}_\lambda(\omega) = \nabla_{\omega} D_{\text{KL}} (\pi_k \parallel \pi_\lambda). 
\]

For details we refer to the publication of the ReF-ER algorithm [19].

3 Related Works

In the earliest MARL studies [31] the value-learning problem is solved via Q-learning with each agent treating its peers as part of the environment. Recent work in cooperative MARL, Sunehag et al. [29] proposed value-decomposition networks (VDN). The joint Q-function is modeled as the sum of individual Q-functions, which solely rely on the observation and action of a single agent. The individual Q-functions are trained by back-propagating the gradients from the Q-learning rule using the joint reward. Thereafter QMIX was developed [24] which uses deep recurrent Q-networks [10] to represent individual state-action-value functions. The output of the Q-networks is combined with a mixing network to estimate a joint state-action-value-function. The weights of the mixing network are chosen such that the maximisation with respect to the actions of the agents yields the same optimum, independent whether the operation is applied on the mixed network or on the individual networks. Son et al. [28] propose QTRAN, which removes some of the structural constraints imposed by VDN and QMIX. Along this, Wen et al. [37] proposed SMIX(\( \lambda \)), where they benefit of the QMIX architecture combined with the novel variant of the off-policy algorithm SARSA(\( \lambda \)) [30] in order to learn the state-action value function. Foerster et al. [5] proposed LOLA, which anticipates the policy change of the other agents by including a one-step look-ahead. This allows to maximize the returns, while considering that in the next update the agents will adjust their behaviour. For continuous problems, Lowe et al. [16] extended DDPG [14] for the MARL setting. They learn a centralized critic from which they train a deterministic policy for each agent. Gupta et al. [8] compare the performance of policy gradient, temporal-difference error, and actor-critic methods in cooperative multi-agent systems. They introduced the Stanford Intelligent Systems Laboratory (SISL) environments that are used as benchmarks in the present work. They evaluated their methods using different training schemes: centralized, concurrent and parameter sharing. In the first scheme they learn and execute in a centralized manner, which suffers from the curse of dimensionality. In the concurrent scheme they train independent policies for each agent and in parameter sharing they learn a single policy where the experiences from all agents are used during training. This study was extended by Terry et al. [33], where the impact of parameter sharing was evaluated on eleven modern single-agent RL algorithms. To the best of our knowledge, this study achieved the highest reported results on the SISL environments, and therefore we use it as the state of the art against which we compare our results. Foerster et al. [6] proposed counterfactual multi agent policy gradients (COMA), which learns a centralized critic that is conditioned on the joint action, while each agent’s policy conditions on its observations. They address the credit assignment problem via a counterfactual baseline that compares the global reward to the obtained reward when that agent’s action is replaced with a default action. This mechanism allows to deduce the contribution of the single agent to the joint reward. However, this typically requires additional simulations to compute the reward using the default action. In order to facilitate this, they propose to learn an additional critic representation that allows an
efficient computation of the baseline. Christianos et al. [3] proposes SEAC, which is an actor-critic algorithm that shares the experiences among all agents. The gradients for the actor and the critic have additional terms from the shared experiences, which are multiplied by an importance weight. By training multiple policies they can increase the exploration rate and by sharing experiences they facilitate learning in approximately equal rates. FACMAC [23] learns a centralized but factored critic and gradient estimator. They have individual critic networks for each agent and combine them into a centralized critic using a non-linear mixing network similar as in QMIX [24]. The difference is that they do not impose any constraints on the critic. They use a gradient similar to MADDPG [16], but instead of optimizing each agents actions separately, they resample actions for all agents with the current policy before calculating the gradient.

4 SISL Environments

We benchmark the performance of our algorithms in the three cooperative Stanford Intelligent Systems Laboratory Environments (SISL) implemented in PettingZoo [8, 32] (see fig. 2).

In Multiwalker a package is located on top of three bipedal robots. The reward for each walker depends on the distance the package has traveled plus 130 times the change in the walker’s position. All agents receive -100 reward if any walker or the package falls. The walker that falls is further penalized by -10 reward. Each walker exerts force on two joints in their two legs, giving a continuous action space represented as a four element vector $A_{\text{multiwalker}} \subseteq \mathbb{R}^4$. The state consists of 31 real values $S_{\text{multiwalker}} \subseteq \mathbb{R}^{31}$ consisting of simulated noisy linear data about the environment and information about neighboring walkers. The environment ends after 500 steps or if the package or a walker falls.

In Waterworld five agents attempt to consume food while avoiding poison. There are ten moving poison targets, which have a radius of 0.75 times the radius of the agent. Furthermore there are five moving food targets with radius two times the size of the agent radius. In the center of the domain is a solid object. An agent obtains a shaping reward of $0.01$ for touching food. The food can be consumed if two agents touch it simultaneously, in which case both participating agents obtain 10 reward. On the other hand, touching a poison target and consuming it gives -1 reward. After consumption both food and poison targets randomly reappear at another location in the environment. The agents have a continuous action space represented by two elements $A_{\text{waterworld}} \subseteq \mathbb{R}^2$, which corresponds to horizontal and vertical thrust. In order to penalize unnecessary movement the agents obtain a negative reward based on the absolute value of the applied thrust. Each agents state results from 30 range-limited sensors, depicted by the black lines, which detect neighboring entities and result in a 242 element vector $S_{\text{waterworld}} \subseteq \mathbb{R}^{242}$. The environment terminates after 500 steps.

In Pursuit there are 30 blue evaders and eight red pursuer agents in a $16 \times 16$ grid with an obstacle in the center, shown in white. Every time the pursuers fully surround an evader, each agent receives a reward of 5 and the evader is removed from the environment. In order to facilitate training, pursuers receive a shaping reward of 0.01 every time they touch an evader. The pursuers have a discrete action space, consisting of directions up, down, left, right or stay $A_{\text{pursuit}} = \{\uparrow, \downarrow, \leftarrow, \rightarrow, \circ\}$. Each pursuer observes a $7 \times 7$ grid centered around itself, depicted by the orange boxes surrounding the red pursuer agents. For each of the gridpoints it receives three signals, the first signal indicates a wall, the second signal indicates the number of allies and the third signal indicates the number of opponents. Thus the observation space can be represented by $S_{\text{pursuit}} \subseteq \mathbb{Z}^{147}$. The environment terminates after 500 steps or if all evaders are captured.
Table 1: Maximum mean cumulative training reward, averaged over 5 runs and 100 episodes and state of the art by [33]. The number in the parenthesis denotes the episode in which it was achieved.

|              | Multiwalker | Waterworld | Pursuit     |
|--------------|-------------|------------|-------------|
| V-RACER (LDI, single policy) | 19.80 (3.2k) | 194.57 (19.2k) | 124.22 (20.8k) |
| PPO, ApeX DDPG, ApeX DQN        | 13.67 (9.2k) | 14.8 (6.3k)  | 77.63 (24k)   |
| QMIX                        | -5.65 (17.8k) | 1.62 (45.3k)  | 45.41 (59.6k) |
| MADDPG                     | -33.95 (44.2k) | -1.81 (41k)   | 4.04 (28.8k)  |

4.1 Evaluation

We benchmark the algorithms using five runs with 20,000 episodes each. For each run we compute the moving median of the cumulative reward averaged over all agents. The window size of the moving median is 100 episodes. Throughout this study, we apply the default parameters suggested in Novati and Koumoutsakos [19]. We set the discount factor $\gamma = 0.995$, we use a replay memory of size $2^{18} = 262,144$ with initial random exploration with a randomly initialized neural network for $2^{17} = 131,072$ experiences. The learning rate of the Adam optimizer is initialized to $\eta = 0.0001$ and the batchsize $B = 256$. For the approximation of the value function and the policy we use a neural network with two hidden layers of width 128. We initialize the Ref-ER factor $\beta = 0.3$, the target threshold $c_{\text{max}} = 4$. In contrast to the original V-RACER, we use the clipped normal distribution for continuous action domains and introduce a discrete variant of V-RACER (see Appendix). The source code is available on https://github.com/will-be-provided and we provide a pseudo code in the appendix. The SISL experiments were executed on a single CPU with 12 cores. The required training-times were approximately 48 hours per run. In total we performed 2x3x4x5=120 runs.

We distinguish the algorithms eq. (12) that calculate the importance weights as V-RACER full dynamics individual-FDI and full dynamics cooperative-FDCo, depending on the value estimator (eq. (8) and eq. (9)). Accordingly, we name the algorithms V-RACER local dynamics individual-LDI and local dynamics cooperative-LDCo, if we apply eq. (13) to calculate the importance weight. Finally, we distinguish training a single shared policy for all agents and one policy per agent. We sample a mini-batch for all variants and use the observations from all agents to train the neural networks using eq. (10) and eq. (11).

In fig. 3 we show the results for the three SISL environments. We find that the LDI algorithm results are superior to those of the other algorithmic variants. More specifically, while in the
Table 2: Testing the optimal policy for LDI on the SISL environments. The mean, maximal, and minimal cumulative reward is computed over 50 episodes. We highlight the better result.

|                      | Multiwalker |                | Waterworld |                | Pursuit |
|----------------------|-------------|----------------|------------|----------------|---------|
|                      | Mean        | Max            | Min        | Mean           | Max     | Min     |
| Single Policy        | 1.0         | 1.9            | 0.4        | 202.3          | 271.8   | 140.6   |
|                      |             |                |            | 56.8           | 84.7    | 11.6    |
| Multiple Policies    | -0.1        | 1.5            | -2.2       | 236.4          | 303.4   | 191.0   |
|                      |             |                |            | 130.3          | 146.6   | 112.8   |

Multiwalker (a,d) environment the improvement is minimal, the advantage is clearly noticeable in the Waterworld (b,e) and Pursuit (c,f) environments. We argue that the FDI and LDI algorithms dominate their cooperative counterparts (FDCo and LDCo) because of the credit assignment problem: After averaging the rewards it is difficult to identify which actions contribute most to the success of the agents. Furthermore, sharing the experiences among all agents in a single policy is sufficient to resolve the non-stationary issue and using eq. (12) does not produce valuable information. In contrast, we find that taking the product of the individual importance weights harms the update. The relative order in terms of performance for the employed algorithms is similar when training multiple policies. Interestingly, the maximal performance of the LDI, reached during training, is worse for multiple policies. During the evaluation, however, the single policy variant is outperformed (see table 2). Another important difference arises for the LDCo. For multiple policies experience sharing does not remove the non-stationary issue. Together with the credit assignment problem, it hinders the convergence of the LDCo method. The FDCo alleviates this problem. By adding the information of the other agents in the importance weight (eq. (12)), the destructive effect of non-stationary and the credit assignment problem is resolved.

Additionally, we compare our results when training a single policy with the results in Terry et al. [33], which, to the best of our knowledge, represent the state of the art on the SISL environments. In Table 1 we show a comparison between the best results in Terry et al. [33] and our best performing alternative (LDI). We find that on all environments LDI outperforms the existing results by a margin.

In Multiwalker, we find that the LDI is the preferred algorithm. This can be especially seen during the first 8k episodes. After that, the returns for some runs drop significantly and yield a lower asymptotic median return. This behaviour is often observed in Walker-like environments. In the failing runs the agents start falling while trying to move faster and faster. In order to avoid this catastrophic failure the algorithm converges to a safe, but sub-optimal policy. Nonetheless, the models that consider the local dynamics are the better performing alternatives. In the Multiwalker the LDI obtains a final return of 4.09, which is comparable to the attained reward with the LDCo (3.73). The other variants do not achieve positive rewards (-31.68 for FDI and -38.11 for FDCo). We observe that in LDI certain runs identify a policy which consistently achieves a cumulative median return of around 20, however the algorithm does not learn the associated policy in every run. Here, the state of the art [33] achieves the highest cumulative reward when using a multi agent version of PPO [26]. From table 1 it can be seen, that the LDI outperforms the multi agent version of PPO.

In the second continuous action environment Waterworld, we see clear differences between the algorithms. The LDI outperforms the other variants significantly. We observe, that the FDI and the LDCo show similar performance. In Waterworld, it seems that correlating gradient updates (FDI) and rewards (LDCo) have similar negative effects. This effects seem to sum up when considering both correlations (FDCo). In Waterworld the state of the art is a MA version of ApeX DDPG [11]. The best variant of the proposed algorithm outperforms the existing results by 13X.

Finally, we discuss the results on the Pursuit benchmark. The negative effect arising from the credit assignment problem (LDCo) is larger than when correlating the update (FDI), which is contrary to our findings in Waterworld. Combining both assumptions (FDCo) is still the worst performing alternative. As can be seen in table 1 the present best performing alternative outperforms the state of the art (a MA version of ApeX DQN [11]). Comparing the maximal achieved mean returns shows that ReF-ER MARL provides 1.6X higher return.
5 ReF-ER and Scientific Multi-Agent Reinforcement Learning

We test LDI with multiple policies in the problem of fish schooling in the presence of strong hydrodynamic interactions, in high fidelity simulations of the Navier-Stokes equations \[34\]. We consider 20 self-propelled swimmers (figure fig. 4) interacting through their vortex wakes. The non-linear vortex interactions make this a challenging control problem \[38\]. Here, each swimmer observes a 16-dimensional state, encoding the ability to sense the environments via sight, flow sensing, and proprioception. Given the observation of the environment, the swimmers can control their motion via two actions that allow steering and changing the swimming velocity. Their reward is composed of the instantaneous swimming efficiency and a penalty term for collisions. In fig. 4 we show the trajectories relative to the center of mass of the school. Controlled schooling behaviour is observed for 100 tail beat periods whereas without control the swimmers collide at 1/5th of this time. Control for high fidelity simulations has a high computational cost, which requires asynchronous training and data collection. We use one compute node to train the neural networks and 64 independent simulations with two nodes for data collection. In total, the run took four days on 129 nodes of a Cray XC50 system equipped with a 12 core Intel® Xeon® CPU and one NVIDIA® Tesla® P100 GPU. This suggests the suitability of our method to expensive scientific computing applications.

6 Discussion

We present ReF-ER MARL, the multi-agent generalization of the V-RACER algorithm with ReF-ER \[19\]. The combination of V-RACER with ReF-ER has shown significant promise in several benchmark problems and challenging fluid dynamics applications.

In the proposed ReF-ER MARL the actions of the agents are independent while the probability distribution of an action is conditioned on the respective agent’s state. We examine the effects of different relations between the agents and vary the strength of their interactions. This is achieved by modifying the value estimator and the importance weight. For the value estimator we distinguish an individual and a cooperative setting by either using the individual reward or the average of the rewards from all agents. The strength of the agents’ interaction is controlled via the importance weight.

We benchmark ReF-ER MARL on the Stanford Intelligent Systems Laboratory (SISL) Environments. We compare the value-estimates, and importance weights for different assumptions. In these collaborative environments one may expect that assuming a strong interaction and cooperative estimates for
the state-value is beneficial. However, we find that the preferred approach is to estimate the value using individual rewards, and to consider a local dynamics model. We find that ReF-ER MARL using a single feed forward neural network outperforms the state of the art algorithms, that often relies on complex network architectures. Finally, we test the proposed algorithms when training multiple policies. The median returns are lower during training. During testing, one controller per agent outperforms a shared controller. Besides robustness, we show that introducing stricter dependencies between the agents via the full dynamics (FDCo) cures the deadly combination of non-stationarity and reward averaging. Furthermore, the LDI with multiple policies is applied to enforce coordinated swimming on 20 swimmers experiencing hydrodynamic interactions in high-fidelity simulations of the Navier-Stokes equation. It enables stable schooling formations for up to 100 tail-beat periods (the formation brakes up after 20 tail beats without control).

**Broader Impact**

Multiagent reinforcement learning has been recently extended to solve accurately underresolved partial differential equations by casting the closure problem as control with continuous action spaces [20, 1]. The success of this effort relied on using V-RACER with ReF-ER [19]. We extend this framework to a variety of multiagent models. We benchmark the proposed algorithms in the SISL environments as well as a computationally expensive application of fish schooling involving direct numerical simulations of the Navier-Stokes equation. The present work can serve as a reference for scientific reinforcement learning community, in applications that require continuous control.

This research doesn’t put anyone at disadvantage nor does a failure of the proposed method yield unfavorable consequences. This work does not leverage any biases in the data.

**Acknowledgments and Disclosure of Funding**

We acknowledge the infrastructure and support of CSCS, providing the necessary computational resources under project s929.

**A V-RACER for discrete action space**

The V-RACER paper [19] presents the continuous action version of V-RACER. In this work, we extend V-RACER for the discrete action domain. For discrete action environments we employ a neural network which takes as an input the state and outputs the state-value estimate, the energies of each action \( \epsilon_i \) for \( i = 1, \ldots, |A| \) and an inverse temperature \( \beta \) parameter. The probability \( p_i \) of sampling action \( a_i \) is calculated according to the Boltzmann distribution

\[
p_i = \frac{\exp (-\epsilon_i) \beta}{\sum_{k=1}^{|A|} \exp (-\epsilon_k) \beta}.
\]

(17)

Here we omit the derivation of the importance weight and the gradient thereof, as well as the gradient of the Kullback-Leibler divergence of the current policy from the past policy.

**B Clipped Normal Distribution**

We use the clipped normal distribution to enforce actions within an interval \([a, b] \subset \mathbb{R}\). The probability density function of the clipped normal distribution [7] is given by

\[
f(x; \mu, \sigma) = \delta(x - a) F_N(a; \mu, \sigma) + \mathbb{1}_{a \leq x \leq b} f_N(x; \mu, \sigma) + \delta(x - b) [1 - F_N(b; \mu, \sigma)],
\]

(18)

where \( \mathbb{1}_{a \leq x \leq b} \) is the indicator function that is 1 inside \([a, b]\) and 0 outside, \( \delta \) denotes the Dirac delta distribution, and \( f_N(x; \mu, \sigma) \) is the density function of the normal distribution with mean \( \mu \) and standard deviation \( \sigma \). In contrast to the squashed normal distribution [9], the clipped normal distribution retains higher probability mass towards the action bounds. We found that clipping performs superior than applying squashing. In the following we derive the gradient of the Kullback-Leibler divergence, and the importance weight and the gradient thereof.
B.1 Kullback-Leibler Divergence

The definition of the Kullback-Leibler divergence is given by

\[
D_{\text{KL}}(p \| q) = \int_{-\infty}^{\infty} \log \left[ \frac{p(x; \mu_p, \sigma_p)}{q(x; \mu_q, \sigma_q)} \right] p(x; \mu_p, \sigma_p) \, dx .
\] (19)

Plugging in the expression from eq. (18) we find

\[
D_{\text{KL}}(p \| q) = \log \left[ \frac{F_N(a; \mu_p, \sigma_p)}{F_N(a; \mu_q, \sigma_q)} \right] F_N(a; \mu_p, \sigma_p)
+ \int_{a}^{b} \log \left[ \frac{f_N(x; \mu_p, \sigma_p)}{f_N(x; \mu_q, \sigma_q)} \right] f_N(x; \mu_p, \sigma_p) \, dx
+ \log \left[ \frac{1 - F_N(b; \mu_p, \sigma_p)}{1 - F_N(b; \mu_q, \sigma_q)} \right] [1 - F_N(b; \mu_p, \sigma_p)] .
\] (20)

Plugging in the expression for the normal distribution we can write the integral expression for

\[
x \in (a, b)
\] as

\[
I = \frac{1}{\sqrt{2\pi} \sigma_p} \int_{a}^{b} \left\{ \log \left( \frac{\sigma_q}{\sigma_p} \right) - \frac{1}{2} \left[ \frac{(x - \mu_p)^2}{\sigma_p^2} - \frac{(x - \mu_q)^2}{\sigma_q^2} \right] \right\} \exp \left[ -\frac{(x - \mu_p)^2}{2\sigma_p^2} \right] \, dx .
\] (21)

Using a substitution of variables \( x' = \frac{x - \mu_p}{\sqrt{2} \sigma_p} \), the integral reads

\[
I = \frac{1}{\sqrt{\pi}} \int_{\frac{a - \mu_p}{\sqrt{2} \sigma_p}}^{\frac{b - \mu_p}{\sqrt{2} \sigma_p}} \left[ \log \left( \frac{\sigma_q}{\sigma_p} \right) - x'^2 + \frac{1}{2} \left( \sqrt{2} \sigma_p x' + \mu_p - \mu_q \right) \right] e^{-x'^2} \, dx' .
\] (22)

We expand \( Q \) giving

\[
Q = \log \left( \frac{\sigma_q}{\sigma_p} \right) - x'^2 - \frac{2\sigma_p^2 x'^2}{\sigma_p^2} + 2\sqrt{2} \sigma_p (\mu_p - \mu_q) x' + (\mu_p - \mu_q)^2
\]

\[= \log \left( \frac{\sigma_q}{\sigma_p} \right) + \frac{(\mu_p - \mu_q)^2}{2\sigma_p^2} + \sqrt{2} \sigma_p (\frac{\mu_p - \mu_q}{\sigma_p^2}) x' - \frac{1}{2} \left( \frac{\sigma_q^2}{\sigma_p^2} \right) x'^2 .
\] (23)

Using the identities

\[
\int_{a}^{b} e^{-x^2} \, dx = \frac{\sqrt{\pi}}{2} \left[ \text{erf}(b) - \text{erf}(a) \right] ,
\]

\[
\int_{a}^{b} x e^{-x^2} \, dx = -\frac{1}{2} \left[ \exp(b^2) - \exp(a^2) \right] ,
\]

\[
\int_{a}^{b} x^2 e^{-x^2} \, dx = \frac{\sqrt{\pi}}{4} \left[ \text{erf}(b) - \text{erf}(a) \right] - \frac{1}{2} \left[ b \exp(-b^2) - a \exp(-a^2) \right] ,
\] (24)
Using the derivatives of the cumulative distribution function with respect to the mean and standard deviation

\[
D_{KL}(p||q) = \log \left[ \frac{F_N'(a; \mu_p, \sigma_p)}{F_N'(a; \mu_q, \sigma_q)} \right] F_N(a; \mu_p, \sigma_p)
+ \frac{1}{2} \left[ \log \left( \frac{\sigma_q}{\sigma_p} \right) + \frac{(\mu_p - \mu_q)^2}{2\sigma_q^2} \right] \left[ \text{erf} \left( \frac{b - \mu_p}{\sigma_p \sqrt{2}} \right) - \text{erf} \left( \frac{a - \mu_p}{\sigma_p \sqrt{2}} \right) \right]
- \frac{1}{\sqrt{2\pi}} \sigma_p \left( \frac{\mu_p - \mu_q}{\sigma_q} \right) \exp \left[ - \frac{(b - \mu_p)^2}{2\sigma_q^2} \right] - \exp \left[ - \frac{(a - \mu_p)^2}{2\sigma_q^2} \right]
- \frac{1}{4} \left( 1 - \sigma_q^2 \right) \left[ \text{erf} \left( \frac{b - \mu_p}{\sqrt{2}\sigma_p} \right) - \text{erf} \left( \frac{a - \mu_p}{\sqrt{2}\sigma_p} \right) \right]
+ \frac{1}{2} \sqrt{2\pi} \left( 1 - \sigma_q^2 \right) \left[ \left( \frac{b - \mu_p}{\sqrt{2}\sigma_p} \right) \exp \left( - \frac{(b - \mu_p)^2}{2\sigma_q^2} \right) \right] - \left( \frac{a - \mu_p}{\sqrt{2}\sigma_p} \right) \exp \left( - \frac{(a - \mu_p)^2}{2\sigma_q^2} \right)
+ \log \left[ \frac{1 - F_N(b; \mu_p, \sigma_p)}{1 - F_N(b; \mu_q, \sigma_q)} \right] \left[ 1 - F_N(b; \mu_p, \sigma_p) \right]
= \log \left[ \frac{F_N'(a; \mu_p, \sigma_p)}{F_N'(a; \mu_q, \sigma_q)} \right] F_N(a; \mu_p, \sigma_p)
+ \frac{1}{2} \left[ \log \left( \frac{\sigma_q}{\sigma_p} \right) + \frac{(\mu_p - \mu_q)^2}{2\sigma_q^2} - \frac{1}{2} \left( 1 - \sigma_q^2 \right) \right] \left[ \text{erf} \left( \frac{b - \mu_p}{\sigma_p \sqrt{2}} \right) - \text{erf} \left( \frac{a - \mu_p}{\sigma_p \sqrt{2}} \right) \right]
+ \frac{1}{\sqrt{2\pi}} \sigma_p \left( \frac{\mu_p - \mu_q}{\sigma_q} \right) \exp \left( - \frac{(b - \mu_p)^2}{2\sigma_q^2} \right) - \frac{1}{2} \left( 1 - \sigma_q^2 \right) \left[ \text{erf} \left( \frac{b - \mu_p}{\sqrt{2}\sigma_p} \right) - \text{erf} \left( \frac{a - \mu_p}{\sqrt{2}\sigma_p} \right) \right]
- \frac{1}{\sqrt{2\pi}} \sigma_q \left( \frac{a - \mu_p}{\sqrt{2}\sigma_p} \right) \exp \left( - \frac{(a - \mu_p)^2}{2\sigma_q^2} \right)
+ \log \left[ \frac{1 - F_N(b; \mu_p, \sigma_p)}{1 - F_N(b; \mu_q, \sigma_q)} \right] \left[ 1 - F_N(b; \mu_p, \sigma_p) \right] .
\]

(25)

Using the derivatives of the cumulative distribution function with respect to the mean and standard deviation

\[
\frac{F_N(x; \mu, \sigma)}{\partial \mu_q} = -f_N(x; \mu, \sigma),
\]

\[
\frac{F_N(x; \mu, \sigma)}{\partial \sigma_q} = -\frac{x - \mu}{\sigma} f_N(x; \mu, \sigma),
\]

(26)

and the derivative of the error function

\[
\frac{\partial}{\partial z} \text{erf}(z) = \frac{2}{\sqrt{\pi}} e^{-z^2},
\]

(27)
the derivative of the Kullback-Leibler divergence with respect to \( \mu_q \) and \( \sigma_q \) is given by

\[
\frac{\partial D_{KL}(p \| q)}{\partial \mu_q} = \frac{F_N(\mu; \mu_q, \sigma_q)}{F_N(\mu; \mu_q, \sigma_q)} f_N(\mu; \mu_q, \sigma_q) + \frac{1}{\sigma_q^2} \left[ \text{erf} \left( \frac{b - \mu_p}{\sqrt{2} \sigma_p} \right) - \text{erf} \left( \frac{a - \mu_p}{\sqrt{2} \sigma_p} \right) \right] - \frac{1}{\sqrt{2\pi} \sigma_q^2} \left[ \exp \left( -\frac{(b - \mu_p)^2}{2 \sigma_p^2} \right) - \exp \left( -\frac{(a - \mu_p)^2}{2 \sigma_p^2} \right) \right]
\]

\[
\frac{\partial D_{KL}(p \| q)}{\partial \sigma_q} = \frac{a - \mu_q}{\sigma_q} \frac{a - \mu_q}{\sigma_q} \frac{\partial D_{KL}(p \| q)}{\partial \mu_q} + \frac{1}{\sqrt{2\pi} \sigma_q^3} \left[ \frac{\sigma_p^2}{\sigma_q^2} \left( \frac{b - \mu_p}{\sigma_p} + 2 \sigma_p (\mu_p - \mu_q) \right) \right] \exp \left( -\frac{(b - \mu_p)^2}{2 \sigma_p^2} \right) - \frac{1}{\sqrt{2\pi} \sigma_q^3} \left[ \frac{\sigma_p^2}{\sigma_q^2} \left( \frac{a - \mu_p}{\sigma_p} + 2 \sigma_p (\mu_p - \mu_q) \right) \right] \exp \left( -\frac{(a - \mu_p)^2}{2 \sigma_p^2} \right) - \frac{b - \mu_p}{\sigma_q} \left( 1 - \frac{1}{\sqrt{2\pi} \sigma_q^2} \right) \exp \left( -\frac{(a - \mu_p)^2}{2 \sigma_p^2} \right) - \frac{b - \mu_p}{\sigma_q} \left( 1 - \frac{1}{\sqrt{2\pi} \sigma_q^2} \right) \exp \left( -\frac{(a - \mu_p)^2}{2 \sigma_p^2} \right).
\]

B.2 Importance Weight

The gradient of the importance weight

\[
\text{IW}(x; \mu_q, \sigma_q, \mu_p, \sigma_p) = \begin{cases} 
F_N(x; \mu_q, \sigma_q) & \text{if } x = a, \\
F_N(x; \mu_q, \sigma_q) & \text{if } a < x < b, \\
1 - F_N(x; \mu_q, \sigma_q) & \text{if } x = b,
\end{cases}
\]

with respect to the parameters \( \mu_q \) and \( \sigma_q \) can be computed using

\[
\frac{\partial f_N(x; \mu, \sigma)}{\partial \mu_q} = \frac{x - \mu}{\sigma^2} f_N(x; \mu, \sigma),
\]

\[
\frac{\partial f_N(x; \mu, \sigma)}{\partial \sigma_q} = \left( -\frac{1}{\sigma} + \frac{(x - \mu)^2}{\sigma^3} \right) f_N(x; \mu, \sigma) = \frac{\mu^2 - \sigma^2 - 2 \mu x + x^2}{\sigma^3} f_N(x; \mu, \sigma),
\]

and eq. (26) as

\[
\frac{\partial \text{IW}}{\partial \mu_q} = \begin{cases} 
\frac{F_N(x; \mu_q, \sigma_q)}{f_N(x; \mu_q, \sigma_q)} & \text{if } x = a, \\
\frac{1 - F_N(x; \mu_q, \sigma_q)}{f_N(x; \mu_q, \sigma_q)} & \text{if } a < x < b, \\
\frac{f_N(x; \mu_q, \sigma_q)}{1 - F_N(x; \mu_q, \sigma_q)} & \text{if } x = b.
\end{cases}
\]

\[
\frac{\partial \text{IW}}{\partial \sigma_q} = \begin{cases} 
\frac{a - \mu_q}{\sigma_q} f_N(x; \mu_q, \sigma_q) & \text{if } x = a, \\
\frac{b - \mu_q}{\sigma_q} \left( 1 - F_N(x; \mu_q, \sigma_q) \right) & \text{if } a < x < b, \\
\frac{b - \mu_q}{\sigma_q} F_N(x; \mu_q, \sigma_q) & \text{if } x = b.
\end{cases}
\]
C Algorithms

The general MARL loop is presented in algorithm 1.

**Algorithm 1 Multi-Agent Reinforcement Learning (Synchronous Variant)**

**Input**: Environment function \( D(s, a) \), Replay Memory \( \mathcal{RM} \), Neural Network(s) \( \mathcal{NN} \) for agents \( i = 1, \ldots, N \), Termination Criteria \( T \)

```plaintext
while not T do
    // COLLECTING EXPERIENCES
    Sample Initial States \( s_0^{(i)} \) for \( i = 1, \ldots, N \)
    for \( t \in [1, T] \) do
        Forward Neural Network \( V^\pi(s_{t-1}^{(i)}), \pi_{t-1}(\cdot|s_{t-1}^{(i)}) = \mathcal{NN}(s_{t-1}^{(i)}; \vartheta, \omega) \) for \( i = 1, \ldots, N \)
        Sample Actions \( a_{t-1}^{(i)} \sim \pi_{t-1}(\cdot|s_{t-1}^{(i)}) \) for \( i = 1, \ldots, N \)
        Run Environment Function \( r_{t-1}, s_t = D(a_{t-1}, s_{t-1}) \)
    end
    // POSTPROCESS EPISODE
    Save Episode \( \mathcal{E}_k = \{(s_t, a_t, r_t, V_t, \pi_t)\}_{t=0}^{T_k} \) in Replay Memory \( \mathcal{RM} \)
    Set \( \hat{V}_t^{abc,(i)} = \hat{V}^\pi(s_t^{(i)}) \) for \( i = 1, \ldots, N \)
    for \( t \in [T-1, \ldots, 1] \) do
        Compute \( \hat{V}_t^{abc,(i)} = V^\pi(s_t^{(i)}) + \hat{p}_t \left[ f(r_t) + \gamma \hat{V}_{t+1}^{abc,(i)} - V^\pi(s_t^{(i)}) \right] \) for \( i = 1, \ldots, N \)
        Add \( \hat{V}_t^{abc,(i)} \) for \( i = 1, \ldots, N \) to Replay Memory \( \mathcal{RM} \)
    end
    // TRAINING NEURAL NETWORK
    for \( i \in [1, \ldots, T] \) do
        miniBatch = generateMinibatch()
        trainPolicy( miniBatch )
        updateReFERparams()
    end
end
```

Here \( f \) implements the chosen relation between the agents. The function \text{UPDATE}REFER\text{PARAMS} updates the ReF-ER parameters, and \text{GENERATE}MINIBATCH samples a minibatch of experiences from the replay memory. \text{TRAIN}POLICY implements the learning algorithm and is detailed in algorithm 2. Depending on the chosen variant of the algorithm the function and \text{COMPUTE}IMP\text{ORTANCE}WEIGHT and \text{COMPUTE}POLICY\text{GRADIENT} implement the respective variant as described in the main text. The function \text{IS}ON\text{POLICY} classifies the experience as on- or off-policy.
Algorithm 2 TRAIN POLICY

Input: miniBatch

\[ \hat{g}^V(\vartheta) = \hat{g}^\pi(\omega) = 0 \]

for \((s_b, a_b, r_b, \pi_b) \in \text{miniBatch}\) do

// FORWARD NEURAL NETWORK

\[ V^\pi(s_b^{(i)}) = NN(s_b^{(i)}; \vartheta, \omega) \text{ for } i = 1, \ldots, N \]

// BACKWARDS UPDATE OF VALUE ESTIMATOR

\[ b_{\text{first}} \leftarrow \text{find first experience for episode containing } b \]

for \(t \in b, \ldots, b_{\text{first}}\) do

\[ V^\pi_f(s_t^{(i)}) \leftarrow \text{scalarize value from replay memory} \]

\[ \hat{V}^\text{tbc}(i) = V^\pi_f(s_t^{(i)}) + \hat{\rho}t \left[ f(r_t) + \gamma \hat{V}^\text{tbc}(i) - V^\pi_f(s_t^{(i)}) \right] \text{ for } i = 1, \ldots, N \]

end

// COMPUTE IMPORTANCE WEIGHT

importanceWeight = computeImportanceWeight(a_b, \pi_b, \pi_\vartheta)

// COMPUTE VALUE GRADIENT

\[ V^\pi_f(s_t^{(i)}) \leftarrow \text{scalarize value } V^\pi_f(s_t^{(i)}) \]

\[ \hat{g}_b^{(i)}(\vartheta) = \frac{1}{N} \left[ V^\pi_f(s_t^{(i)}) - V^\text{tbc}(s_t^{(i)}) \right] \nabla_{\vartheta} V^\pi_f(s_t^{(i)}) \text{ for } i = 1, \ldots, N \]

// COMPUTE POLICY GRADIENT

\[ \hat{g}_b^{(i)}(\omega) = \nabla_{\vartheta} D_{KL}(\pi_b \parallel \pi_\omega)(s_b^{(i)}) \text{ for } i = 1, \ldots, N \]

if isOnPolicy(importanceWeight) then

\[ \hat{g}_b^{(i)}(\omega) = \text{computePolicyGradient(importanceWeight,} \ V^\pi_f(s_t^{(i)}), \pi_\omega(s_t^{(i)})) \]

else

\[ \hat{g}_b^{(i)}(\omega) = 0 \]

end

// ACCUMULATE GRADIENT FOR VALUE FUNCTION

\[ \hat{g}^V(\vartheta) = \hat{g}^V(\vartheta) + \frac{1}{N_{\text{miniBatch}}} \hat{g}_b^{(i)}(\vartheta) \text{ for } i = 1, \ldots, N \]

// ACCUMULATE GRADIENT FOR POLICY

\[ \hat{g}^\pi(\omega) = \hat{g}^\pi(\omega) + \frac{1}{N_{\text{miniBatch}}} \left[ (\beta \hat{g}_b^{(i)}(\omega) + (1 - \beta)\hat{g}_b^{KL}(\omega)) \right] \text{ for } i = 1, \ldots, N \]

end

// UPDATE HYPERPARAMETERS

\[ \vartheta \leftarrow \vartheta - \eta \hat{g}^V(\vartheta) \]

\[ \omega \leftarrow \omega - \eta \hat{g}^\pi(\omega) \]
References

[1] H Jane Bae and Petros Koumoutsakos. Scientific multi-agent reinforcement learning for
wall-models of turbulent flows. Nature Communications, 13(1):1–9, 2022. URL https://doi.org/10.1038/s41467-022-28957-7

[2] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. Science, 365(6456):885–890, 2019. ISSN 0036-8075. doi: 10.1126/science.aay2400. URL https://science.sciencemag.org/content/365/6456/885

[3] Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Shared experience actor-critic
for multi-agent reinforcement learning. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia
Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, Advances in Neural Information
Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020,
NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/7967cc8e3ab559e68cc944c44b1cf3e8-Abstract.html

[4] Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic. CoRR,
abs/1205.4839, 2012. URL http://arxiv.org/abs/1205.4839

[5] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and
Igor Mordatch. Learning with opponent-learning awareness. In Elisabeth André, Sven Koenig,
Mehdi Dastani, and Gita Sukthankar, editors, Proceedings of the 17th International Conference
on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15,
2018, pages 122–130. International Foundation for Autonomous Agents and Multiagent Systems
Richland, SC, USA / ACM, 2018. URL http://dl.acm.org/citation.cfm?id=3237408

[6] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon
Whiteson. Counterfactual multi-agent policy gradients. In Sheila A. McIlraith and Kilian Q.
Weinberger, editors, Proceedings of the Thirty-Second AAAI Conference on Artificial Intelli-
gence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and
the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New
Orleans, Louisiana, USA, February 2-7, 2018, pages 2974–2982. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17193

[7] Yasuhiro Fujita and Shin-ichi Maeda. Clipped action policy gradient. In Proceedings of the
35th International Conference on Machine Learning, volume 80, pages 1597–1606, 2018. URL
http://proceedings.mlr.press/v80/fujita18a.html

[8] Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. Cooperative multi-agent control
using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors,
Autonomous Agents and Multi Agent Systems - AAMAS 2017 Workshops, Best Papers, São
Paulo, Brazil, May 8-12, 2017, Revised Selected Papers, volume 10642 of Lecture Notes in
Computer Science, pages 66–83. Springer, 2017. doi: 10.1007/978-3-319-71682-4_5. URL https://doi.org/10.1007/978-3-319-71682-4_5

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-
policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy
and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine
Learning, volume 80 of Proceedings of Machine Learning Research, pages 1861–1870. PMLR,
10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html

[10] Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable
mdps. In 2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015, pages 29–
37. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673

[11] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van
Hasselt, and David Silver. Distributed prioritized experience replay. CoRR, abs/1803.00933,
2018. URL http://arxiv.org/abs/1803.00933
[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980

[13] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing, 190:82–94, 2016. doi: 10.1016/j.neucom.2016.01.031. URL https://doi.org/10.1016/j.neucom.2016.01.031

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016. URL http://arxiv.org/abs/1509.02971

[15] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In William W. Cohen and Haym Hirsh, editors, Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994, pages 157–163. Morgan Kaufmann, 1994. doi: 10.1016/b978-1-55860-335-6.50027-1. URL https://doi.org/10.1016/b978-1-55860-335-6.50027-1

[16] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 6379–6390, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c19911a1d64c-Abstract.html

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. Nat., 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236

[18] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 1046–1054, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c3992e9a6c5ae12bd18488bc579b30d-Abstract.html

[19] Guido Novati and Petros Koumoutsakos. Remember and forget for experience replay. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 4851–4860. PMLR, 2019. URL http://proceedings.mlr.press/v97/novati19a.html

[20] Guido Novati, Hugues Lascombes de Laroussilhe, and Petros Koumoutsakos. Automating turbulence modelling by multi-agent reinforcement learning. Nat. Mach. Intell., 3(1):87–96, 2021. doi: 10.1038/s42256-020-00272-0. URL https://doi.org/10.1038/s42256-020-00272-0

[21] Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. J. Artif. Intell. Res., 32:289–353, 2008. doi: 10.1613/jair.2447. URL https://doi.org/10.1613/jair.2447

[22] OpenAL, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Sharig Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning. arXiv e-prints, art. arXiv:1912.06680, December 2019.
[23] Bei Peng, Tabish Rashid, Christian A. Schroeder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. FACMAC: Factored Multi-Agent Centralised Policy Gradients. arXiv e-prints, art. arXiv:2003.06709, March 2020.

[24] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In Jennifer G. Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 4292–4301. PMLR, 2018. URL http://proceedings.mlr.press/v80/rashid18a.html

[25] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of JMLR Workshop and Conference Proceedings, pages 1889–1897. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/schulman15.html

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. CoRR, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. CoRR, abs/1712.01815, 2017. URL http://arxiv.org/abs/1712.01815

[28] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 5887–5896. PMLR, 2019. URL http://proceedings.mlr.press/v97/son19a.html

[29] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning. CoRR, abs/1706.05296, 2017. URL http://arxiv.org/abs/1706.05296

[30] Richard S. Sutton and A Barto. Reinforcement Learning - An Introduction. MIT Press, 2018. ISBN 9780262039246. doi: 10.1016/S1364-6613(99)01331-5. URL http://incompleteideas.net/book/RLbook2018.pdf

[31] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Machine Learning Proceedings 1993, pages 330–337. Morgan Kaufmann, San Francisco (CA), 1993. ISBN 978-1-55860-307-3. doi: https://doi.org/10.1016/B978-1-55860-307-3.50049-6. URL https://www.sciencedirect.com/science/article/pii/B9781558603073500496

[32] J. K Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sulivan, Luis Santos, Rodrigo Perez, Caroline Horsch, Clemens Dieffendahl, Niall I. Williams, Yashas Lokesh, Ryan Sullivan, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. arXiv preprint arXiv:2009.14471, 2020.

[33] Justin K Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black. Revisiting parameter sharing in multi-agent deep reinforcement learning. arXiv preprint arXiv:2005.13625, 2020.

[34] Siddhartha Verma, Guido Novati, and Petros Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. Proceedings of the National Academy of Sciences, 115(23):5849–5854, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1800923115. URL https://www.pnas.org/content/115/23/5849
[35] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 11 2019. ISSN 0028-0836. doi: 10.1038/s41586-019-1724-z. URL [https://doi.org/10.1038/s41586-019-1724-z](https://doi.org/10.1038/s41586-019-1724-z).

[36] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *CoRR*, abs/1611.01224, 2016. URL [http://arxiv.org/abs/1611.01224](http://arxiv.org/abs/1611.01224).

[37] Chao Wen, Xinghu Yao, Yuhui Wang, and Xiaoyang Tan. Smix(λ): Enhancing centralized value functions for cooperative multi-agent reinforcement learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7301–7308. AAAI Press, 2020. URL [https://ojs.aaai.org/index.php/AAAI/article/view/6223](https://ojs.aaai.org/index.php/AAAI/article/view/6223).

[38] Huiyang Yu, Bo Liu, Chengyun Wang, Xuechao Liu, Xi-Yun Lu, and Haibo Huang. Deep-reinforcement-learning-based self-organization of freely undulatory swimmers. *Phys. Rev. E*, 105:045105, Apr 2022. doi: 10.1103/PhysRevE.105.045105. URL [https://link.aps.org/doi/10.1103/PhysRevE.105.045105](https://link.aps.org/doi/10.1103/PhysRevE.105.045105).