Independent Generative Adversarial Self-Imitation Learning in Cooperative Multiagent Systems

Xiaotian Hao*, Weixun Wang*, Jianye Hao✉, Yaodong Yang
College of Intelligence and Computing, Tianjin University
Tianjin, China
{xiaotianhao,wxwang,jianye.hao}@tju.edu.cn,yydapple@gmail.com

ABSTRACT

Many tasks in practice require the collaboration of multiple agents through reinforcement learning. In general, cooperative multiagent reinforcement learning algorithms can be classified into two paradigms: Joint Action Learners (JALs) and Independent Learners (ILs). In many practical applications, agents are unable to observe other agents' actions and rewards, making JALs inapplicable. In this work, we focus on independent learning paradigm in which each agent makes decisions based on its local observations only. However, learning is challenging in independent settings due to the local viewpoints of all agents, which perceive the world as a non-stationary environment due to the concurrently exploring teammates. In this paper, we propose a novel framework called Independent Generative Adversarial Self-Imitation Learning (**IGASIL**) to address the coordination problems in fully cooperative multiagent environments. To the best of our knowledge, we are the first to combine self-imitation learning with generative adversarial imitation learning (GAIL) and apply it to cooperative multiagent systems. Besides, we put forward a Sub-Curriculum Experience Replay mechanism to pick out the past beneficial experiences as much as possible and accelerate the self-imitation learning process. Evaluations conducted in the testbed of StarCraft unit micromanagement and a commonly adopted benchmark show that our **IGASIL** produces state-of-the-art results and even outperforms JALs in terms of both convergence speed and final performance.

KEYWORDS

Multiagent learning; Learning agent-to-agent interactions (coordination); Adversarial machine learning

ACM Reference Format:
Xiaotian Hao*, Weixun Wang*, Jianye Hao✉, Yaodong Yang. 2019. Independent Generative Adversarial Self-Imitation Learning in Cooperative Multiagent Systems. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

With the advance of deep neural network [14, 22], Deep Reinforcement Learning (DRL) approaches have made significant progress for a number of applications including Atari games [30], Go [42], game theory [23, 46] and robot locomotion and manipulation [24, 41]. In practice, a large number of important applications can be naturally modeled as cooperative multiagent systems. Examples include coordination of robot swarms [18], coordination of autonomous vehicles [7], network packet delivery [48], managing air traffic flow [2] and energy distribution [47].

However, directly applying single-agent reinforcement learning approaches such as Q-learning to cooperative multiagent environments behaves poorly. Thus, effective coordination mechanism needs to be incorporated into agents' learning strategies to address the cooperative multiagent problems. Multiagent reinforcement learning (MARL) can be generally classified in to two paradigms [8]: Joint Action Learners (JALs) and Independent Learners (ILs). JALs observe the rewards and actions (policies) taken by all agents whose information is explicitly considered during policy update, whereas ILs make decisions based on their local observations, actions and rewards only. Under the JAL paradigm, MADDPG [27] is a recently proposed approach for multiagent games with large continuous state space and action space. By taking the other agents' observations and policies directly into consideration, MADDPG learns a centralized critic and uses the centralized critic to provide a better guidance for the policy update. However, MADDPG does not consider some specially designed mechanisms for handling the multiagent cooperative challenges when dealing with difficult cooperation environments (e.g., sparse rewards, high miss-coordination penalties, exploration-exploitation trade-off [29]). Besides, due to the inaccessibility of all other agents' states and actions in practice and the exponential growth of the state-action space in the number of agents, JALs are difficult to be applied to practical applications.

Avoiding the above two restrictions, ILs are more universally applicable and have been widely studied over the past years, e.g., Distributed Q-learning [21], Hysteretic Q-learning [28] and Lenient Learners [37]. However, for ILs, one typical issue is that each agent's policy is changing as training progresses, and the environment becomes non-stationary from the perspective of any individual agent since other agents' policies are changing concurrently. Hysteretic Q-learning [28] and Lenient Learners [37] are proposed to facilitate multiple reinforcement learning agents to overcome the independent learning problems (e.g., the non-stationary problem [29]). Very recently, the idea of hysteretic Q-learning and lenient learners has been successfully applied to deep multiagent reinforcement learning settings [35, 36]. However, all these approaches are Q-learning based methods and are naturally suitable for settings with discrete action space only. Therefore, it's difficult to apply these approaches to solve the cooperative multiagent continuous control tasks.

In this work, we propose a novel framework under the independent learning paradigm called independent generative adversarial self imitation learning (IGASIL), which conducts both learning

* Equal contribution. ✉Corresponding author.
Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

and execution in a fully decentralized manner. In the framework, there are n independent agents cooperatively solving a task without knowing other agents' policies and making decisions based on their own local observations. Initially, each agent maintains a positive buffer and a normal buffer. At run time, each agent interacts with the environment independently according to the current policy. The resulting trajectory is stored twice in the positive buffer and the normal buffer. The positive buffer is a specially designed sub-curriculum experience replay which continuously helps to pick out and reserve preferable experiences the agent has experienced. Combining self-imitation learning with generative adversarial imitation learning, each agent trains a discriminator using samples from these two buffers whose target is to capture the features of the past good experiences. Besides the environment rewards, each agent receives additional rewards from the discriminator, which would guide the agents to imitate from the past good experiences and do more exploration around these high-reward regions. Once the agents find better policies, they will produce higher quality trajectories. Thus, the learning will turn into a virtuous circle until a good cooperation is achieved.

The main contributions of this paper can be summarized as follows.

(1) To the best of our knowledge, we are the first to combine self imitation learning with generative adversarial imitation learning and propose a novel framework called Independent Generative Adversarial Self Imitation Learning (IGASIL) to address the multiagent coordination problems.
(2) We put forward a Sub-Curriculum Experience Replay mechanism to accelerate the self-imitation learning process.
(3) IGASIL is well applicable to both discrete and continuous action spaces and can be integrated with any Policy Gradient or Actor-Critic algorithm in fully cooperative multiagent environments.
(4) Besides, our proposed method follows the decentralized training pattern which does not require any communication among agents during learning.
(5) Experimental results show that our method outperforms state-of-the-art in cooperative multiagent continuous and discrete control tasks in terms of both convergence speed and final performance.

2 BACKGROUND

2.1 Markov Decision Process

We use the tuple $(S, A, P, r, \rho_0, \gamma)$ to define an infinite-horizon, discounted Markov decision process (MDP), where S represents the state space, A represents the action space, $P : S \times A \times S \to [0, 1]$ denotes the transition probability distribution, $r : S \times A \to \mathbb{R}$ denotes the reward function, $\rho_0 \to [0, 1]$ is the distribution of the initial state s_0, and $\gamma \in (0, 1)$ is the discount factor. Let π_θ denote a stochastic policy $\pi : S \times A \to [0, 1]$, where θ is the parameter of the policy. The performance of a stochastic policy π_θ is usually evaluated by its expected cumulative discounted reward J_{π_θ}:

$$J_{\pi_\theta} = \mathbb{E}_{\rho_0, P, \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)] \tag{1}$$

Reinforcement Learning (RL) [44] is a set of algorithms trying to infer a policy π_θ, which maximizes the expected cumulative discounted reward J_{π_θ} when given access to a reward signal $r(s, a)$.

2.2 Generative Adversarial Imitation Learning

Imitation learning is also known as learning from demonstrations or apprenticeship learning, whose goal is to learn how to perform a task directly from expert demonstrations, without any access to the reward signal $r(s, a)$. Recent main lines of researches within imitation learning are behavioural cloning (BC) [6, 39], which performs supervised learning from observations to actions when given a number of expert demonstrations; inverse reinforcement learning (IRL)[1], where a reward function is estimated that explains the demonstrations as (near) optimal behavior; and generative adversarial imitation learning (GAIL) [3, 4, 17, 43], which is inspired by the generative adversarial networks (GAN) [15]. Let T_E denote the trajectories generated by the behind expert policy π_E, each of which consists of a sequence of state-action pairs. In the GAIL framework, an agent mimics the behavior of the expert policy π_E by matching the generated state-action distribution $\rho_{\pi_\theta}(s, a)$ with the expert's distribution $\rho_{\pi_E}(s, a)$. The state-action visitation distribution (occupancy measure [17]) of a policy π_θ is defined as:

$$\rho_{\pi_\theta}(s, a) = \pi_\theta(a|s) \sum_{t=0}^{\infty} \gamma^t p(s_t = s|\pi_\theta) \tag{2}$$

where $p(s_t = s|\pi_\theta)$ is the probability of being in state s at time t when starting at state $s_0 \sim \rho_0$ and following policy π_θ. Thus, J_θ can be written as:

$$\begin{aligned} J_{\pi_\theta} &= \mathbb{E}_{\rho_0, P, \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)] \\ &= \sum_{t=0}^{\infty} \sum_{s} p(s_t = s|\pi_\theta) \sum_{a} \pi_\theta(a|s) \gamma^t r(s_t, a_t) \\ &= \sum_{s} \sum_{a} \pi_\theta(a|s) \sum_{t=0}^{\infty} p(s_t = s|\pi_\theta) \gamma^t r(s_t, a_t) \\ &= \sum_{s,a} \rho_{\pi_\theta}(s, a) r(s_t, a_t) \end{aligned} \tag{3}$$

which only depends on the discounted state-action visitation distribution $\rho_{\pi_\theta}(s, a)$. The optimum is achieved when the distance between these two distributions is minimized as measured by Jensen-Shannon divergence. The formal GAIL objective is denoted as:

$$\begin{aligned} &\min_\theta \max_w \mathbb{E}_{(s,a)\sim\rho_{\pi_E}(s,a)}[\log(D_w(s, a))] + \\ &\mathbb{E}_{(s,a)\sim\rho_{\pi_\theta}(s,a)}[\log(1 - D_w(s, a))] - \lambda_H H(\pi_\theta) \end{aligned} \tag{4}$$

where D_w is a discriminative binary classifier parameterized by w which tries to distinguish state-action pairs from the trajectories generated by π_θ and π_E, $H(\pi_\theta) \triangleq \mathbb{E}_{\rho_0, P, \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t(-\log \pi_\theta(a|s))]$ is the $\gamma-$discounted causal entropy of policy π_θ [5] and λ_H is the coefficient. Unlike GANs, the original GAIL requires interactions with the environment/simulator to generate state-action pairs, and thus the objective (4) is not differentiable end-to-end with respect to the policy parameter θ. Hence, optimization of the policy requires RL techniques based on Monte-Carlo estimation of policy

gradients. The optimization over the GAIL objective is performed by alternating between K gradient step to increase (4) with respect to the discriminator parameters w, and a Trust Region Policy Optimization (TRPO) step to decrease (4) with respect to the policy parameters θ (using $\log(D_w(s, a))$ as the reward function).

2.3 Sample-Efficient GAIL

One of the most important advantages of GAIL is that it can obtain a higher performance than behavioral cloning when given only a small number of expert demonstrations. However, a large number of policy interactions with the learning environment are required for policy convergence. As illustrated in [20], while GAIL requires as little as 200 expert frame transitions to learn a robust reward function on most MuJoCo [45] tasks, the number of policy frame transitions sampled from the environment can be as high as 25 million in order to reach convergence, which is intractable for real-world applications. To this end, [20] address the sample inefficiency issue via incorporating an off-policy RL algorithm and an off-policy discriminator to dramatically decrease the sample complexity by many orders of magnitude. Experimental results show that their off-policy approach works well even without using the importance sampling.

2.4 Self-Imitation Learning

In an environment with the very sparse reward, it's difficult to learn the whole task at once. It is natural to master some basic skills for solving easier tasks firstly. e.g., In Montezuma's Revenge (an Atari game), the agent needs to pick up the key and then open the door. Directly learning opening the door is hard due to the poor exploration, but it is easier to master picking up the key at first. Based on this idea, self-imitation learning (SIL) [34] is a very recent approach proposed to solve the sparse reward problem by learning to imitate the agent's own past good experiences. In brief, SIL stores previous experiences in a replay buffer and learns to imitate the experiences when the return is greater than the agent's expectation. Experimental results show that this bootstrapping approach (learn to imitate the agent's own past good decisions) is highly promising on hard exploration tasks A proper level of exploitation of past good experiences during learning can lead to a deeper exploration (moving to the deeper region) of the learning environment. Similar idea and results can also be found in [19].

3 PROBLEM DESCRIPTION

The setting we are considering is a fully cooperative partially observable Markov game [26], which is a multiagent extension of a Markov decision process (MDPs). A Markov game for N agents is defined by a set of states S describing the possible configurations of all agents and environment, a set of actions $A_1, ..., A_N$ and a set of observations $O_1, ..., O_N$ for each agent. Initial states are determined by a distribution $\rho_0 : S \to [0, 1]$. State transitions are determined by a function $P : S \times A_1 \times ... \times A_N \times S \to [0, 1]$. For each agent i, rewards are given by function $r_i : S \times A_1 \times ... \times A_N \to \mathbb{R}$, observations are given by function $o_i : S \to O_i$. To choose actions, each agent i uses a stochastic policy $\pi_i : O_i \times A_i \to [0, 1]$. The joint policy $\boldsymbol{\pi}$ of all agents is defined as $\boldsymbol{\pi} : \langle \pi_1, ..., \pi_N \rangle$. The joint action is represented as $\boldsymbol{a} = \langle a_1, ..., a_n \rangle$. We consider a finite horizon setting, with episode length T. If all agents receive the same rewards ($r_1 = r_2 = ... = r_N$), the Markov game is fully cooperative, which means a best-interest action of one agent is also a best-interest action of all agents. Besides, we only consider the environments with deterministic reward functions at present.

In the following of this paper, we are going to analyze and deal with the coordination problems under the following two difficult cooperative environments: (1) cooperative endangered wildlife rescue; (2) decentralised StarCraft micromanagement from an independent perspective. For example, in the cooperative endangered wildlife rescue task, there are N slower independent rescue agents which have to cooperatively chase and rescue one of the M faster wounded animals in a randomly generated environment with continuous state and action spaces. Each rescue agent makes decisions (go north, south, east, or west to chase one of the M animals) based on its local observation only and can't observe the others' policies **(local observation and continuous action space)**. Only when the N rescue agents chase and capture the same wounded animal simultaneously, will they get a reward based on the caught animal's value. So, the reward is very sparse and it's hard for the independent rescuers to explore **(sparse rewards)**. Besides, the changing of the other agents' policies (e.g. the other agents' move to different directions for exploration instead of cooperatively capturing the same animal with the current agent) will influence the reward of current agent's action Therefore, the environment becomes non-stationary from the perspective of each individual rescuer **(non-stationary)**. If a rescue agent changes its actions too quickly when perceiving the changed reward (due to the others' explorations), the others will change their policies in their turn **(exploration-exploitation)**. Moreover, different wounded animals has different rewards and different penalties (the animal with the higher reward also has the higher penalty for miss-coordination). So, to avoid punishment, the rescuers prefer to capture the animals with the lowest reward instead of the global optimal one **(high penalty and shadowed equilibrium)**. Detail settings of the game are shown in Section 5.1.1. Thus, it's hard to coordinate the independent learners to achieve successful cooperation and converge to a better equilibrium. Thus, additional cooperation mechanisms are needed.

4 INDEPENDENT SELF-IMITATION LEARNING FRAMEWORK

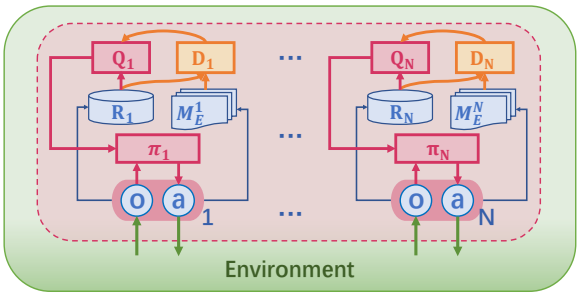


Figure 1: IGASIL framework for cooperative multiagent systems.

4.1 Independent Generative Adversarial Self-Imitation Learning

Combining the idea of self-imitation learning with GAIL, we propose a novel independent generative adversarial self-imitation learning (IGASIL) framework aiming at facilitating the coordination procedure of the interactive agents, reducing the learning variance and improving the sample efficiency. The learning procedure for each independent learner i is summarized in Algorithm 1.

Algorithm 1 Independent Generative Adversarial Self-Imitation Learning

1: **Input:** For each agent i, initial parameters of actor, critic, discriminator $\theta_i^0, \phi_i^0, w_i^0$, sub-curriculum experience replay M_E^i and a normal replay buffer R_i.
2: Initialize $M_E^i \leftarrow \varnothing$, $R_i \leftarrow \varnothing$.
3: **for** n=0,1,2,... **do**
4: Sample a trajectory $T_n^i \sim \pi_{\theta_i}$.
5: Store T_n^i in R_i.
6: Store T_n^i in M_E^i according to Algorithm 2.
7: Sample state-action pairs $X_n \sim R_i$ and $X_E \sim M_E$ with the same batch size.
8: Update w_i^n to w_i^{n+1} by ascending with gradients:

$$\begin{aligned}\Delta_{w_i^n} =& \hat{\mathbb{E}}_{X_E}[\log(D_{w_i^n}(s,a))]+ \\ & \hat{\mathbb{E}}_{X_n}[\log(1-D_{w_i^n}(s,a))]\end{aligned} \tag{5}$$

9: Sample (s,a,r,s',done) tuples $X_n' \sim R_i$.
10: Calculate the imitation reward for each $(s,a) \sim X_n'$ by:

$$r_{imit}(s,a) = \log(D_{w_i^n}(s,a)) - \log(1-D_{w_i^n}(s,a)) \tag{6}$$

11: Calculate the reshaped reward for each $(s,a) \sim X_n'$ by:

$$r'(s,a) = r + \lambda_{imit} * r_{imit}(s,a) \tag{7}$$

12: Replace r in X_n' with $r'(s,a)$.
13: Using samples X_n' to update the policy parameter θ_i^n to θ_i^{n+1} and the critic parameter ϕ_i^n to ϕ_i^{n+1} according to DDPG [25] (off-policy A2C [9])
14: **end for**

An illustration of our IGASIL is shown in Figure (1). Initially, each agent i maintains a sub-curriculum experience replay buffer M_E^i and a normal buffer R^i (Algorithm 1, Line 1-2). At run time, each agent interacts with the environment independently according to the current policy. The resulting trajectory is stored in M_E^i and R^i respectively (stored twice) (Algorithm 1, Line 4-6). The normal buffer R^i is used for off-policy training, which will be discussed in Section 4.1.1. The sub-curriculum experience replay buffer M_E^i of each agent preserves the past useful skills (demonstrations) for future use, which will be detailed in Section 4.2. We consider these useful demonstrations in M_E^i as self generated expert data and regard the policy behind these self-generated demonstrations as π_E^i for each agent i. At the same time, each agent trains a discriminator D_i using samples from these two buffers whose target is to capture the features of the past good experiences. (Algorithm 1, Line 7-8). Then, each agent begins to update its policy based on two types of rewards: (1) the imitation rewards given by the discriminator D_i (Algorithm 1, Line 9-10), which will be discussed in Section 4.1.2; (2) the original environment rewards, which are combined according to Equation (7), in which λ_{imit} control the weight of the imitation reward[1] (Algorithm 1, Line 11-12). The final reshaped reward $r'(s,a)$ will encourage each agent to explore more around the nearby region of the past good experience to check whether a better coordination can be achieved or have been achieved. After that, the policy of each agent is updated according to the corresponding update rules (Algorithm 1, Line 13). Thus, under the guidance of the discriminator, the past good experiences and skills are dynamically reused. After that, each agent's policy is more likely updated towards a better direction independently. Though we use the off-policy actor-critic approaches (DDPG and off-policy A2C) in our algorithm, our self-imitation framework can be integrated with any policy gradient or actor-critic methods.

Since all independent agents receive exactly the same reward and use the same learning approach (same parameters and settings), the positive trajectories stored in M_E^i would be stored in a synchronized way, which means the agents could cooperatively imitate the "same" past good experience in a distributed way. Thus, all independent agents would have the same behavioral intentions (e.g., jointly imitating the same past good experience and doing deeper exploration, which we call "the joint intention") during learning. As the "joint" imitation learning progresses, the policy of each agent would be induced to update towards the "same" direction. Consequently, the non-stationary and learning issues can be alleviated.

4.1.1 ***Sample-efficient GASIL.*** One limitation of GAIL is that it requires a significant number of interactions with the learning environment in order to imitate an expert policy [20], which is also the case of our settings. To address the sample inefficiency of GASIL, we use off-policy RL algorithms (Here, we use DDPG and off-policy A2C) and perform off-policy training of the GAIL discriminator performed in such way: for each agent i, instead of sampling trajectories from the current policy directly, we sample transitions from the replay buffer R_i collected while performing off-policy training:

$$\begin{aligned}&\min_{\theta}\max_{w}\hat{\mathbb{E}}_{(s,a)\sim\pi_E^i}[\log(D_{w_i}(s,a))]+ \\ &\hat{\mathbb{E}}_{(s,a)\sim R_i}[\log(1-D_{w_i}(s,a))] - \lambda_H H(\pi_\theta^i)\end{aligned} \tag{8}$$

Equation (8) tries to match the occupancy measures between the expert and the distribution induced by the replay buffer R_i instead of the latest policy π_i.It has been found that the off-policy GAIL works well in practice even without using importance sampling [20]. As will be shown in Section 5.2, we also observe similar phenomenons in our cooperative endangered wildlife rescue environment.

4.1.2 ***Unbiased imitation reward.*** Another problem of GAIL is that either $r_{imit}(s,a) = -log(1-D(s,a))$ or $r_{imit}(s,a) = log(D(s,a))$ (which is often used as the reward function in GAIL approaches) has reward biases that can either implicitly impose prior knowledge about the true reward, or alternatively, prevent the policy from imitating the optimal expert [20]. We summarize the reason

[1] In our settings, we grow the λ_{imit} exponentially as learning progresses (One intuition is that as the training progresses, the trajectories produced by the agent becomes better and better. Thus, the agent should pay more attention to these better ones).

of the two rewards' bias here: (1) $-log(1 - D(s, a))$ is always positive and potentially provides a survival bonus which drives the agent to survive longer in the environment to collect more rewards. (2) $log(D(s, a))$ is always negative and provides a per step penalty which drives the agent to exit from the environment earlier. Thus, to stabilize the training process of our IGASIL, we use a more stable reward function as shown in Equation (9). Similar analysis can be found in [13] and [20].

$$r_{imit}(s, a) = log(D(s, a)) - log(1 - D(s, a)) \tag{9}$$

4.2 Sub-Curriculum Experience Replay

In a complex cooperative game, a series of actions need to be taken simultaneously by all agents to achieve a successful cooperation. However, due to the independent learning agents (ILs) interacting with the environment according to their own observations and policies without any communication, each agent might randomly take different actions for exploration at the same state. But, to achieve a perfect cooperation, each agent must exactly select the "right" action at all states. This means the collected trajectories of successful cooperation are very few during learning. So, it's difficult for the independent agents to grasp all these series of actions simultaneously to achieve perfect cooperation at once. Therefore, additional mechanisms are needed to induce the individual agents to gradually pick the "right" actions at the same state.

Curriculum learning is an extension of transfer learning, where the goal is to automatically design and choose a sequence of tasks (i.e. a curriculum) $T_1, T_2, ...T_t$ for an agent to train on, such that the learning speed or performance on a target task T_t will be improved [33]. Inspired by this idea, we want our independent learning agents to follow the curriculum learning paradigm. For example, it's easier for the agents to firstly learn to cooperate at some easier states. And then, reusing the basic skills learned in the previous step, the agents would gradually achieve deeper cooperation and finally are able to solve the target task. The main idea is that the past useful skills can be reused to facilitate the coordination procedure. Similar ideas have been applied to a series of curriculum learning tasks [31], [32]. In our settings, we consider a whole trajectory as an instance of solving the target task. Our goal is to find and reuse the past useful skills/experiences for each agent to accelerate the cooperation process. An intuitive way is to pick out these useful experiences by rewards. One example is that given two trajectories with rewards $[0, +1, +3, +1, 0, 0, -20]$ and $[0, 0, 0, 0, 0, 0, -15]$, though the total rewards are both -15 (low), there is still some useful experience included in the first trajectory (e.g.: the sub-trajectory $[0, +1, +3, +1]$ with a total reward +5 still demonstrates some good behaviors). By imitating the behaviors from these good sub-trajectories, the agents can still grasp some useful cooperation skills. Another example is considering a trajectory with sparse rewards $[0, 0, 0, ..., 0, 0, +1]$ (only receiving +1 at the terminal state), imitating from the sub-trajectories near the terminal state (e.g. $[0, +1]$, $[0, 0, +1]$) would drive the agent to quickly master skills around the terminal state, reduce unnecessary explorations, and thus ease the reward back-propagation problem when rewards are sparse, which is similar to the idea of reverse curriculum generation for reinforcement learning [10].

Algorithm 2 Sub-Curriculum Experience Replay

1: **Given:** A learning policy π_{θ_i} for agent i.
2: Initialize the min-heap based positive trajectory buffer M_E^i.
3: **for** n=0,1,2,... **do**
4: Sample a trajectory $T_n^i \sim \pi_{\theta_i}$.
5: Calculate the discounted Return of T_n^i as $R_{T_n^i}$.
6: Store $(T_n^i, R_{T_n^i})$ in M_E^i. ▷ *resorting by priority*
7: **for** j=0,N **do**
8: Randomly sample a sub-trajectory $sub_j(T_n^i)$ from T_n^i without repetition.
9: Calculate the discounted Return of $sub_j(T_n^i)$ as $R_{sub_j(T_n^i)}$.
10: Store $(sub_j(T_n^i), R_{sub_j(T_n^i)})$ in M_E^i. ▷ *resorting by priority*
11: **end for**
12: Update the policy π_{θ_i} according to Algorithm 1.
13: **end for**

Given the above analysis, we build a min-heap based sub-curriculum experience replay (SCER) M_E^i with a small buffer size k for each agent i to continuously maintain the past beneficial experiences. The formal description of our SCER is summarized in Algorithm 2. The priority of the min-heap is based on the return of each trajectory/sub-trajectory. To pick out the beneficial (useful) experiences as much as possible, we randomly sample some sub-trajectories from a given trajectory without repetition (Algorithm 2, Line 8), calculate their discounted returns and feed them into M_E^i (Algorithm 2, Line 9-10). The ranking and filtering of the trajectories are processed within M_E^i. Since M_E^i is built based on a min-heap, it can be viewed as a filter which keeps the latest top-k-return trajectories/sub-trajectories (Algorithm 2, Line 10). The positive buffer M_E^i usually starts by storing suboptimal trajectories (e.g., killing only few enmies in StarCraft micromanagement games), and our sub-curriculum ER with IGASIL allows each agent to learn better sub-policies in the subspaces. Based on the pre-learned skills, the agents are easier to explore to the deeper regions and the positive buffer will receive trajectories with higher quality. This leads to agents learning better coordinated policies in return.

5 EXPERIMENTS

In the following experiments, we evaluate the effectiveness of our IGASIL framework in two cooperative multiagent benchmarks: (1) cooperative endangered wildlife rescue, which has the very sparse reward and high miss-coordination penalty [27]; (2) decentralised StarCraft micromanagement, which has multi types of units, strong stochasticity and uncertainty [11, 38, 40].

5.1 Effectiveness of our Approach

Architecture & Training. In this paper, all of our policies, critics and discriminators are parameterized by a two-layer ReLU MLP (Multilayer Perceptron) followed by a fully connected layer activated by tanh functions for DDPG's policy nets(DDPG is used only for the animal rescue game), softmax functions for AC's policy nets and sigmoid functions for all discriminators. Only in decentralised StarCraft micromanagement task , we share the parameters among the homogeneous agents (units with the same type) to accelerate

the training process.Due to the space limit, we will put the details of the network structures and all hyperparameter settings in an online appendix (e.g., Arxiv).

5.1.1 Cooperative Endangered Wildlife Rescue.

Game Settings. Cooperative endangered wildlife rescue is a more tough version (sparse rewards and high penalty of miss-coordination) of the "predator-prey" task illustrated in MADDPG [27], which requires more accurate cooperation. There are N slower cooperating rescue agents which cooperatively chase and rescue one of the M faster wounded animals in a randomly generated environment. Each time if all the cooperative rescuers capture a wounded animal **simultaneously**, the agents will be rewarded by some rewards based on the wounded animal they saved. Different wounded animals (e.g., Lion, wildebeest and Deer) correspond to different rewards and different risks. Different risks means that the penalties for miss-coordination on different animals are different (e.g., hurt by the lion). The target for each rescue agent is learning to rescue the same wounded animal independently without knowing each other's policy. Besides, in our settings, we stipulate a rescue agent can hold a wounded animal without suffering any penalty for some game steps T_{hold} before the other partners' arrival. So, the difficulty level of the task can be modulated by the value of T_{hold}. The larger, the easier. In the following experiments, we set N to 2, M to 3 and T_{hold} to 8. An typical illustration of the cooperative endangered wildlife rescue task is shown in Figure (2).

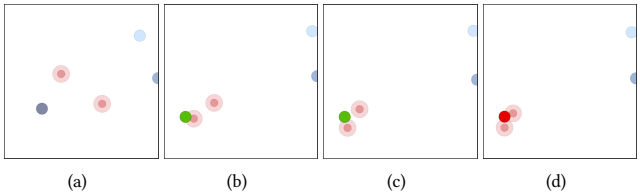


Figure 2: (a) Two rescue agents are in red (translucent red represents the grasper of each agent) and three wounded animals are in blue (deeper blue represents higher value and higher risk); (b), (c) The animal in dark blue turns to green, which means it has been hold by a rescue agent (can't move anymore); (d) The animal in dark blue turns to red, which means it has been captured and saved by the two rescue agents and the episode finished.

States and Actions. All rescue agents and wounded animals can observe the relative positions of others. Besides, each rescue agent can observe the relative velocities of wounded animals (can't observe the other rescuers' velocities). Actions are accelerations in 4 directions (controlled by 2 actions actually: north or south, east or west). The acceleration of a direction is controlled by the force applied. To sum up, the action space is continuous with a valid range of [-1, 1].

Reward Function. The environmental rewards are sparse and only depend on the terminal state of each episode. The payoff matrix of the terminal state is defined in Table (1). At each state s_T, if both agents capture and save the same target a, b or c simultaneously, they will both receive 11, 7 or 5. Else, if one agent captures i while the other captures j, they will both receive $r(i, j)$. Finally, if one agent holds i and the other doesn't come over in T_{hold} steps, they will both be punished by $r(i, 3)$. According to Table (1), the theoretical-optimal action is "*catch a*", but the action "*catch c*" can be easily mistaken for having the highest reward due to the lowest penalty for exploration. The game can be seen as a Markov extension of the climbing game [8] with continuous state and action spaces.

Table 1: The payoff matrix of the rescue agents at the terminal state of each episode. Both agents receive the same payoff written in the corresponding cell.

		Agent 2			
		catch a	catch b	catch c	on the road
Agent 1	catch a	11	-30	0	-30
	catch b	-30	7	6	-10
	catch c	0	6	5	0
	on the road	-30	-10	0	0

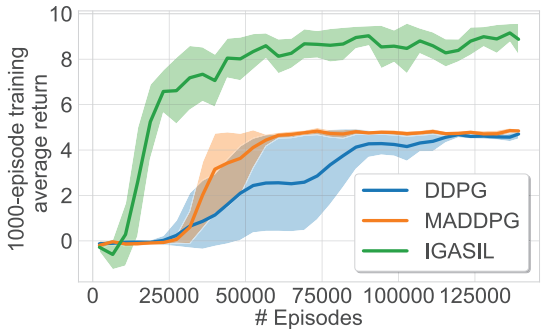
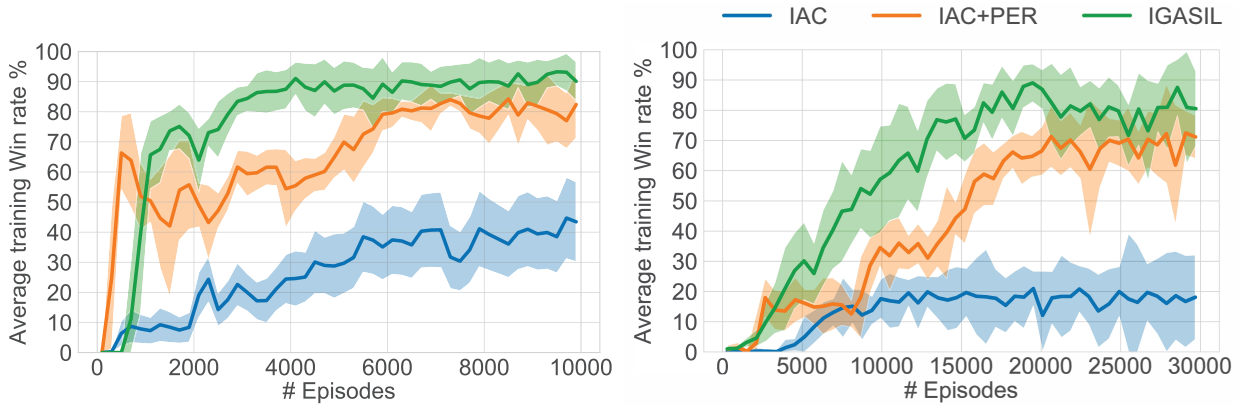


Figure 3: The 1000-episode averaged return of IGASIL versus maddpg and ddpg during training in cooperative endangered wildlife rescue task from the viewpoint of the two rescue agents.

Experimental Results To make the different algorithms comparable, we pre-train both the rescue agents and wounded animal agents with DDPG and save the animal models during training. Then, we reuse the same pre-trained animal models as the default policies for the wounded animals in all experiments. The learning curves of IGASIL versus MADDPG and DDPG are plotted in Figure (3) under five random seeds (1,2,3,4,5). To show a smoother learning procedure, the reward value is averaged every 1000 episodes. Apparently, our IGASIL outperforms MADDPG and DDPG by a significant margin in terms of both convergence rate and final performance. To obviously express the different equilibrium the three algorithms converged to, we show the average number of touches of the three wounded animals by the two rescue agents during training with different algorithms in Figure (5). As illustrated in Table (1), "animal a" has the highest reward +11 and the highest miss-coordination penalty -30. In Figure (5), we can easily observe that only our IGASIL succeeds in converging to the optimal Nash equilibrium (a, a) (learned to rescue "animal a") while MADDPG and DDPG converge to the worst equilibrium (c, c) (learned to rescue "animal c"). This result shows that only our IGASIL overcame the risk of being punished by miss-coordination and achieved a better cooperation result (which need more accurate collaborations).

5.1.2 Decentralised StarCraft Micromanagement.



(a) 5m (b) 2d3z

Figure 4: Win rates for IGASIL, IAC and IAC+PER on two different scenarios.

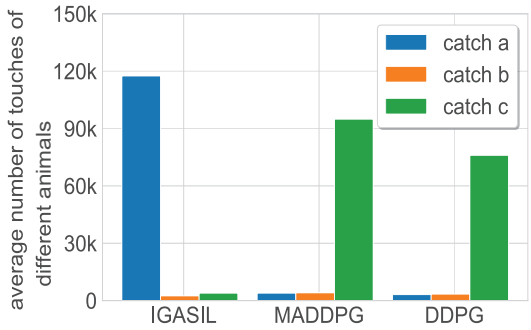


Figure 5: The average number of touches of different wounded animals by the two rescues during training with different algorithms.

Game Settings. In this section, we focus on the problem of micromanagement in StarCraft, which refers to the low-level control of individual units' positioning and attack commands as they fight with enemies. This task is naturally represented as a multiagent system, where each StarCraft unit is controlled by a decentralized independent controller (agent). We consider two scenarios with symmetric teams formed of: 5 marines (5m) and 2 dragoons with 3 zealots (2d_3z). The enemy team is controlled by the built-in StarCraft AI, which uses reasonable but suboptimal hand-crafted heuristics. Since the game is easily obtained and is fair for comparison, micromanagement of StarCraft has become a standard testbed for multagent learning algorithms (for both independent learners and joint learners), which has been widely studied in recent years such as COMA [11], BiCNet [38], QMIX [40]. Different from their approaches which are all joint learners, we focus on fully independent learning paradigm (independent learning and independent execution). Similar settings can be found in [12]. The settings of action space, state features and reward function are similar to that in COMA [11]. Due to the space limit, we move these details to the online appendix (e.g., Arxiv).

Experimental Results. Figure (4) shows the average training win rates as a function of episode for each method and for each StarCraft scenario. For each method, we conduct 5 independent trials and calculate the win rate every 200 training episodes and average them across all trials. In Figure (4), IAC represents the independent Actor-Critic. The actor and critic parameters are also shared among the homogeneous agents. IAC+PER means we add a positive replay buffer M_E^i to each IAC but each agent only stores the original entire trajectory into M_E^i instead of additionally sampling and storing some sub-trajectories (sub-skills). IGASIL is our approach, which equals to IAC+SCER.

Table 2: Mean win percentage across final 1000 evaluation episodes for the different scenarios. The highest mean performances are in bold. The results of COMA are extracted from the published paper [11].

Map	Heur.	IAC	IAC+PER	COMA	IGASIL
5 M	66	45	85	81	**96**
2d3z	63	23	76	47	**87**

The results show that our IGASIL is superior to the IAC baselines in all scenarios. For the parameters sharing among the homogeneous agents (which has been shown to be useful in facilitate training [16]), IAC also learned some coordination on the simpler m5v5 scenario. However, it's hard for IAC to achieve cooperation on the more complicated 2d3z scenario, due to the different types of units, local observations and the resulting dynamics. On 2d3z, our IGASIL still achieves a 82% win rate during training and achieves a 87% win rate in evaluation. In Table (2), we summarize the averaged evaluation win rates of different approaches under multiple combat scenarios (The results of COMA are extracted from the published paper [11].). The winning rate of the built-in AI (Heur.) is also provided as an indicator of the level of difficulty of the combats. The best result for a given map is in bold. The results show that our independent IGASIL outperforms all approaches in the performance of evaluation win rate and even outperforms the centralized trained COMA. Besides, our IGASIL converges faster than COMA according to Figure (4) and Figure (3) of [11].

5.2 Sample Efficiency of IGASIL

To show the sample efficiency of our off-policy IGASIL in cooperative multiagent systems, we compare the performance of the on-policy (on-policy AC+SCER) and off-policy versions (based on off-policy AC+SCER) of IGASIL in the animal rescue task. To clearly

see the influence of the on-policy and off-policy only, we initialize M_E^i for each agent i with the same 32 demonstrations (demonstrated by the pre-trained DDPG agent in Section 5.1.1, which will cooperatively catch "animal c", resulting an average return +5). We perform learning under 5 random seeds (1,2,3,4,5). The imitation learning results are shown in Figure (6). The x-axis represents the number of episodes interacting with the environment during training. From the figure, we see both the on-policy and off-policy IGASILs finally achieved cooperation (learned to "catch c" simultaneously). However, the off-policy IGASIL is able to recover the expert policy given a significantly smaller number of samples than the on-policy version (about 10 times less). Thus, the sample-efficiency was significantly improved.

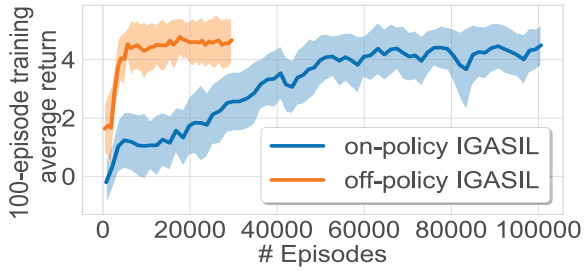


Figure 6: The 100-episode training average returns of on-policy and off-policy IGASIL respectively.

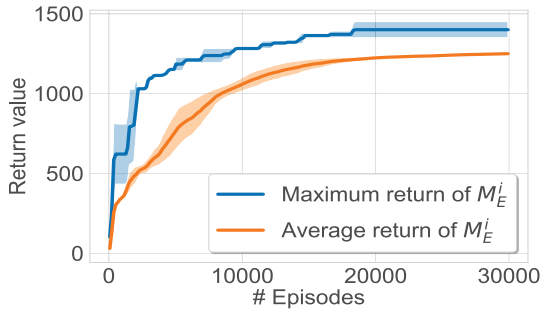


Figure 7: The curves of the maximum and average return values of the trajectories/sub-trajectories stored in M_E^i during training on the 2d3z scenario.

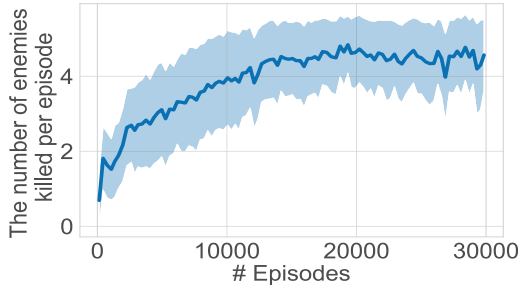


Figure 8: The curve of the number of enemies killed per episode during training on the 2d3z scenario (5 enemies in total).

5.3 Contributions of Sub-Curriculum Experience Replay

We analyze the roles and contributions of the sub-curriculum experience replay here. In Figure (4), we see IAC+PER outperforms the IAC baseline in both scenarios, which means adding a positive buffer M_E^i to store the good trajectories and doing self-imitation learning could help the independent agents reach a cooperation. Besides, IGASIL (IAC+SCER) outperforms IAC+PER in both scenarios, which indicates that adding additional sub-trajectories sampled from the original one (useful sub-skills) to M_E^i could further accelerate the self-imitation procedure and achieve better cooperation. Besides, from Figure (7), we see that as the self-imitation learning progresses, better experiences are stored in M_E^i. Then, the agents would grasp these beneficial skills stored in M_E^i via self-imitation learning. Using the learned new skills, the agents are more likely to reach a better cooperation and learn better policies. As shown in Figure (8), as the stored experiences in M_E^i gets better and better, the number of enemies killed per episode (which indicates the performance of the current policies) grows. All these analysis illustrates that our sub-curriculum experience replay does help the independent agents to achieve better cooperation. However, in our settings, we limit the size of the normal buffers and positive buffers. The techniques proposed in [12] might be incorporated into our framework to allow us to use larger buffer size and further improve the sample efficiency. We will consider it as future work.

6 CONCLUSION & FUTURE WORK

In this paper, we presented a novel framework called independent generative adversarial self-imitation learning (IGASIL) to address the coordination problems in some difficult fully cooperative Markov Games. Combining self-imitation learning with generative adversarial imitation learning, IGASIL address the challenges (e.g., non-stationary and exploration-exploitation) by guiding all agents to frequently explore more around the nearby regions of the past good experiences and learn better policy. Besides, we put forward a sub-curriculum experience replay mechanism to accelerate the self-imitation learning process. Evaluations conducted in the testbed of StarCraft unit micromanagement and cooperative endangered wildlife rescue show that our IGASIL produces state-of-the-art results in terms of both convergence speed and final performance. In order to obviously see whether our IGASIL can significantly facilitate the coordination procedure of the interactive agents, we only consider the environments with deterministic reward functions at present. In our future work, we are going to deal with more challenging cooperative tasks (e.g., environments with stochastic rewards).

ACKNOWLEDGMENTS

The work is supported by the National Natural Science Foundation of China (Grant Nos.: 61702362, U1836214), Special Program of Artificial Intelligence, Tianjin Research Program of Application Foundation and Advanced Technology (No.: 16JCQNJC00100), and Special Program of Artificial Intelligence of Tianjin Municipal Science and Technology Commission (No.: 569 17ZXRGGX00150).

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 1.

[2] Adrian K Agogino and Kagan Tumer. 2012. A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems* 24, 1 (2012), 1–25.

[3] Feryal Behbahani, Kyriacos Shiarlis, Xi Chen, Vitaly Kurin, Sudhanshu Kasewa, Ciprian Stirbu, João Gomes, Supratik Paul, Frans A Oliehoek, João Messias, et al. 2018. Learning from Demonstration in the Wild. *arXiv preprint arXiv:1811.03516* (2018).

[4] Raunak P Bhattacharyya, Derek J Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J Kochenderfer. 2018. Multi-agent imitation learning for driving simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1534–1539.

[5] Michael Bloem and Nicholas Bambos. 2014. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 4911–4916.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[7] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2013. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9, 1 (2013), 427–438.

[8] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998 (1998), 746–752.

[9] Thomas Degris, Martha White, and Richard S Sutton. 2012. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839* (2012).

[10] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300* (2017).

[11] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926* (2017).

[12] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1146–1155.

[13] Justin Fu, Katie Luo, and Sergey Levine. 2017. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. *arXiv preprint arXiv:1710.11248* (2017).

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[16] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 66–83.

[17] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 4565–4573.

[18] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. 2017. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011* (2017).

[19] Bingyi Kang, Zequn Jie, and Jiashi Feng. 2018. Policy Optimization with Demonstrations. In *International Conference on Machine Learning*. 2474–2483.

[20] I. Kostrikov, K. Krishna Agrawal, D. Dwibedi, S. Levine, and J. Tompson. 2018. Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning. *ArXiv e-prints* (Sept. 2018). arXiv:1809.02925

[21] Martin Lauer and Martin Riedmiller. 2000. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[23] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 464–473.

[24] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[26] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*. Elsevier, 157–163.

[27] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[28] Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat. 2007. Hysteretic Q-Learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams.. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'07*. 64–69.

[29] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. 2012. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review* 27, 1 (2012), 1–31.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[31] Sanmit Narvekar. 2016. Curriculum Learning in Reinforcement Learning:(Doctoral Consortium). In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1528–1529.

[32] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. 2016. Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 566–574.

[33] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. 2017. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 147. 149.

[34] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-Imitation Learning. *arXiv preprint arXiv:1806.05635* (2018).

[35] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *arXiv preprint arXiv:1703.06182* (2017).

[36] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. 2018. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 443–451.

[37] Liviu Panait, Keith Sullivan, and Sean Luke. 2006. Lenient learners in cooperative multiagent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 801–803.

[38] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069* (2017).

[39] Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation* 3, 1 (1991), 88–97.

[40] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1803.11485* (2018).

[41] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[42] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[43] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. 2018. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 7472–7483.

[44] Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*. MIT press.

[45] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 5026–5033.

[46] Weixun Wang, Jianye Hao, Yixi Wang, and Matthew Taylor. 2018. Towards Cooperation in Sequential Prisoner's Dilemmas: a Deep Multiagent Reinforcement Learning Approach. *arXiv preprint arXiv:1803.00162* (2018).

[47] Yaodong Yang, Jianye Hao, Mingyang Sun, Zan Wang, Changjie Fan, and Goran Strbac. 2018. Recurrent Deep Multiagent Q-Learning for Autonomous Brokers in Smart Grid.. In *IJCAI*. 569–575.

[48] Dayong Ye, Minjie Zhang, and Yun Yang. 2015. A multi-agent framework for packet routing in wireless sensor networks. *sensors* 15, 5 (2015), 10026–10047.