Learning to Communicate Using Counterfactual Reasoning

Simon Vanneste
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium
simon.vanneste@uantwerpen.be

Astrid Vanneste
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium
astrid.vanneste@uantwerpen.be

Kevin Mets
University of Antwerp - imec
IDLab - Department of Computer Science
Antwerp, Belgium
kevin.mets@uantwerpen.be

Tom De Schepper
University of Antwerp - imec
IDLab - Department of Computer Science
Antwerp, Belgium
tom.deschepper@uantwerpen.be

Ali Anwar
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium
ali.anwar@uantwerpen.be

Siegfried Mercelis
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium
siegfried.mercelis@uantwerpen.be

Steven Latré
University of Antwerp - imec
IDLab - Department of Computer Science
Antwerp, Belgium
steven.latre@uantwerpen.be

Peter Hellinckx
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium
peter.hellinckx@uantwerpen.be

ABSTRACT

Learning to communicate in order to share state information is an active problem in the area of multi-agent reinforcement learning (MARL). The credit assignment problem, the non-stationarity of the communication environment and the problem of encouraging the agents to be influenced by incoming messages are major challenges within this research field which need to be overcome in order to learn a valid communication protocol. This paper introduces the novel multi-agent counterfactual communication learning (MACC) method which adapts counterfactual reasoning in order to overcome the credit assignment problem for communicating agents. Next, the non-stationarity of the communication environment, while learning the communication Q-function, is overcome by creating the communication Q-function using the action policy of the other agents and the Q-function of the action environment. As the exact method to create the communication Q-function can be computationally intensive for a large number of agents, two approximation methods are proposed. Additionally, a social loss function is introduced in order to create influenceable agents, which is required to learn a valid communication protocol. Our experiments show that MACC is able to outperform the state-of-the-art baselines in four different scenarios in the Particle environment. Finally, we demonstrate the scalability of MACC in a matrix environment.

KEYWORDS

Communication Learning, Multi-Agent, Reinforcement Learning

Proc. of the Adaptive and Learning Agents Workshop (ALA 2022), Cruz, Hayes, da Silva, Santos (eds.), May 9-10, 2022, Online, https://ala2022.github.io/. 2022.

1 INTRODUCTION

A lot of research has been done towards single-agent reinforcement learning (RL) [15, 16, 27, 28, 30]. However, many of the practical applications can naturally be described as cooperative multi-agent systems such as industrial robotics and network packet routing. Single agent RL approaches suffer in this kind of applications as the observation space increases and the joint action space will increase exponentially with the number of agents. Cooperative Multi-agent reinforcement learning (MARL) [1] models these applications as a number of decentralized policies with their individual observation, actions and policy. These decentralized policies are trained using a shared team reward in order to promote cooperation between the agents. While this separation into different agents will help with the scalability issue, the agent can encounter partial observability due to the individual agents only having access to their individual observations. This can make it hard or even impossible to find an effective action policy. Cooperative multi-agent systems can be extended with inter-agent communication to overcome this issue. The communication allows the agent to share information about their individual observation with other agents. As it is unknown what information is needed by the other agents, the communication between the agents needs to be learned. The learned communication protocol gives the agent access to the parts of the global state which it requires to learn an effective action policy.

In this work, we use the centralized training and decentralized execution (CTDE) paradigm [5, 7, 10]. This allows us to train the policies in a centralized manner and use additional information (e.g. other policies, observations). After the training of the policies, these policies can be executed decentralized without the need for additional information that is used during training. We focus on a non-differentiable binary communication setting (e.g. {[0, 0], [0, 1], [1, 0], [1, 1]}) in which it takes one timestep to receive a message

which was sent by another agent. This one timestep delayed binary communication is very close to the communication that is used within most real world applications (e.g. digital communication). However, several problems arise within this setting. First, the agents within a cooperative multi-agent reinforcement learning setting that use a shared team reward will encounter the credit assignment problem [2, 7]. The credit assignment problem occurs when agents are having difficulty deducing what impact an action had on the shared team reward. This problem becomes even larger when learning inter-agent communication as the shared team reward is not only used to learn an action policy, but also to learn a communication protocol. Next, the non-stationarity problem [6] will occur when multiple agent simultaneously learn in a multi-agent setting. This makes the environment appear non-stationary as actions that were previously rewarded at a given state may not be rewarded anymore as the behaviour of the other agents has changed. This problem also increases when learning inter-agent communication because the change in the action policy of the agents changes the utility of a message.

In this paper, we present multi-agent counterfactual communication (MACC) learning, a RL method to simultaneously learn to act and communicate in a multi-agent environment. Multi-agent counterfactual reasoning for the action policy, in order to overcome the credit assignment problem, has already been described and used by Foerster et al. [7] in their COMA method. However, MACC extends COMA by using counterfactual reasoning to learn a communication protocol with a centralized critic. This critic requires an additional novel communication Q-function which is created from the action Q-function and the policies of the other agents using a novel decomposition into two separate communication Q-functions. This minimizes the non-stationarity of the communication environment for the communication Q-function. As the computational cost of the exact method to calculate the communication Q-function rises exponentially with the number of agents, we investigate two approximation methods. Next, we investigate the impact of communication between two communication policies on the communication Q-function and propose an algorithm to include this. Additionally, a novel social loss function is introduced for the agent policies in order to promote social behaviour and thereby improve the learning stability. MACC is evaluated and compared against MADDPG [13] and COMA [7] in four different scenarios within the Particle environment from OpenAI [13, 18]. We chose these methods because they most closely match our approach and the problem setting that we target. Finally, we investigate the scalability of the different MACC variations on a Matrix environment.

This paper is structured as follows. In Section 2 of this paper, we discuss relevant literature to our work. Section 3 provides background in Markov Decision Processes and counterfactual reasoning [7]. In Section 4, we explain the different components of MACC in detail. Section 5 shows the different experiments we performed to demonstrate our method. Finally, our conclusions and future work are described in Section 6.

2 RELATED WORK

Recently, several different models for multi-agent communication learning have been presented. The foundations in this research field were laid by Foerster et al. [5] and Sukhbaatar et al. [25]. Foerster et al. [5] proposed two different methods. In Reinforced Inter Agent Learning (RIAL), the communication policy is learned by applying the reward in the same way as we do for the action policy. However, the best results are achieved with Differentiable Inter Agent Learning (DIAL) which uses gradient feedback from the receiving agent to learn the communication policy. Sukhbaatar et al. [25] use a similar technique in CommNet. However, CommNet uses continuous communication instead of discrete communication and assumes that multiple steps of communication can occur before the agents take an action. These fundamental works were followed by more research in the field of multi-agent communication learning. Peng et al. [21] presented BiCNet, an actor-critic model that is able to play real-time strategy games such as StarCraft. Mao et al. [14] proposed two methods, one where communication between actors is learned and one where communication between critics is learned. In contrast to our method, both Peng et al. [21] and Mao et al. [14] use a separate local critic for each of the agents.

A lot of recent work investigates non-broadcast communication. Jiang and Lu [9] propose ATOC which allows agents to choose whether communication is necessary for cooperation and which agents to communicate with. TarMAC [3] allows both the sender and the receiver to determine the importance of a certain message. I2C [4] is a method which realizes non-broadcast communication by learning a prior network in order to create a belief of the other agents which is used to decide whether to communicate with the other agent or not. Other recent advances include the work of Ossenkopf et al. [20]. They use deep MARL to improve the long-term coordination of agents. Vanneste et al. [29] target the lazy agent problem in communication learning by applying value decomposition [26] on DIAL [5], resulting in improved learning speed and performance. In the multi-agent deep deterministic policy gradient (MADDPG) method, Lowe et al. [13] introduced the idea of using a centralized critic for each agent in MARL . They adapted deep deterministic policy gradient [23] by using this centralized critic. Simões et al. [24] introduce Asynchronous Advantage Actor Centralized-Critic with Communication (A3C3), a method based on the single agent Asynchronous Advantage Actor-Critic (A3C) method [15]. A3C3 uses a centralized critic to make a value estimation of a centralized observation. The communication network is optimized by propagating the gradients of the receiving actors through the communication network.

Jaques et al. [8] propose a method that uses social influence as an extra component of the communication reward. The communication policy is trained by a reward composed of the sum of the team reward and a social influence reward. This reward is calculated by determining how much the message influenced the action choice of the other agent. The ideas behind the work of Jaques et al. [8] and our work seem similar. However, there are major differences. Jaques et al. [8] still use the team reward and purely looks at the change in action distribution. This change in the action distribution does not necessarily result in an improved communication protocol. Instead of using the change in action distribution, we use the action distribution in combination with the action Q-function to determine if a message will result in improved performance.

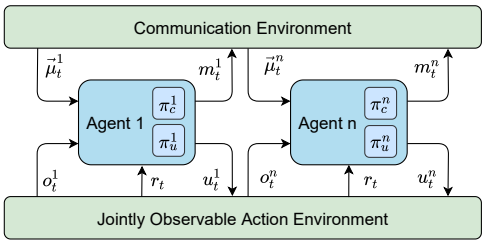


Figure 1: Dec-MDP with a separate action environment and communication environment.

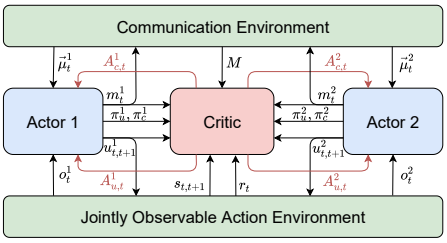


Figure 2: The MACC architecture using a centralised critic.

3 BACKGROUND

In this section, the Markov Decision Process for multi-agent communication and counterfactual reasoning are discussed. In this work, we use the notation where the superscript indicates the index of a certain agent. The index uses the notation a for the current agent and $-a$ for every agent except the current agent. The subscript is used to indicate if a symbol is used for the action (u) or communication (c) environment or to indicate the timestep t. The notation $s_{t,t+1}$ is used as an abbreviated notation for s_t, s_{t+1}. Next, $\mathbb{E}[X; P]$ will represent the expected value of X under the distribution P.

3.1 Markov Decision Processes

In this paper, we use the decentralized Markov Decision Process framework (Dec-MDP) [19] extended with communication as shown in Figure 1. In Dec-MDPs, n agents learn their policy based on the global team reward r_t. Every agent only receives a partial observation o_t of the full state s_t. In addition to the jointly observable action environment, a communication environment is added. Each time-step t, agent a receives, alongside the observation o_t^a from the action environment, a number of incoming messages $\vec{\mu}_t^a$ from the communication environment. The action policy of the agent $\pi_u^a(u_t^a|o_t^a, \vec{\mu}_t^a)$ takes these inputs and samples an action u_t^a. These actions are then processed by the environment and a team reward r_t is given to the agents. The communication policy of the agent $\pi_c^a(m_t^a|o_t^a, \vec{\mu}_t^a)$ uses the observation and received messages to generate an outgoing message m_t^a. The messages are processed by the communication environment to get the input messages for the next time-step $\vec{\mu}_{t+1} = M(m_t)$. The communication function M determines who receives which messages at the next time-step and $\vec{\mu}_{t+1}^a = M^a(m_t)$ is the part of the communication environment which determines what messages are received by agent a. When combining the policies of all agents we can describe the joint action policy $\pi_u(u_t|o_t, \vec{\mu}_t)$ and the joint communication policy $\pi_c(m_t|o_t, \vec{\mu}_t)$. In our work, we allow for agents that can either send messages, perform actions in the environment or both depending on the requirements of the application.

3.2 Counterfactual Multi-Agent Policy Gradient

Foerster et al. [7] showed that policy gradient agents in a multi-agent system can be trained using a centralised critic. This centralised critic is able to predict the joint state-action utility $Q_u(s_t, u_t)$ which can be used to calculate the action advantage function A_u^a. This calculation uses counterfactual reasoning by subtracting the expected utility of the action policy $V_u^a(s_t, u_t^{-a})$ from the state-action utility. The expected utility can be expanded into the marginalization over the counterfactual actions $u_t'^a$ of the action policy π_u^a which is multiplied with the joint action Q-value of that action permutation. Equation 1 and 2 show the advantage calculation for the action policy of agent a as described by Foerster et al. [7].

$$A_u^a(s_t, u_t) = Q_u(s_t, u_t) - V_u^a(s_t, u_t^{-a}) \tag{1}$$

$$V_u^a(s_t, u_t^{-a}) = \sum_{u_t'^a} \left(Q_u(s_t, (u_t'^a, u_t^{-a})) \pi_u^a(u_t'^a|o_t^a) \right) \tag{2}$$

The actions of the other agents u_t^{-a} are constant during the marginalization so that the agent only reasons about its own actions. The action advantage function A_u^a calculates the advantage for the action u_t^a using the joint action Q-function. The joint action Q-function is learned by the centralized critic during training.

4 METHODS

In this section, we discuss the multi-agent counterfactual communication (MACC) learning method. Figure 2 shows the high level architecture of MACC. In MACC, multiple actors act in the environment and communicate with each other. The centralized critic learns a joint action Q-function based on the current state and the received rewards. The critic uses the action Q-function in order to calculate the action and communication advantage for both policies using the action and communication policies of the different agents.

First, in Section 4.1 we discuss the modifications we made to perform counterfactual reasoning for the action policy to include received messages. Next, the training of the joint action Q-function is described in Section 4.2. In Section 4.3, we discuss how counterfactual reasoning can be used to calculate an advantage for the communication policies. This advantage calculations requires a communication Q-function which is discussed in Section 4.4. Section 4.5 describes approximation methods to calculate the communication Q-function. In Section 4.6, we discuss the final component of the communication Q-function when including counterfactual reasoning with communication policies that can receive messages. Finally, the novel social loss function is discussed in Section 4.7.

4.1 Counterfactual Reasoning in the Action Environment

The action policy advantage and value calculations are adapted to include received messages $\vec{\mu}_t^a$ as show in Equation 3 and Equation 4.

$$A_u^a(s_t, \vec{\mu}_t^a, u_t) = Q_u(s_t, u_t) - V_u^a(s_t, \vec{\mu}_t^a, u_t^{-a}) \quad (3)$$

$$V_u^a(s_t, \vec{\mu}_t^a, u_t^{-a}) = \sum_{u_t'^a} \left(Q_u(s_t, (u_t'^a, u_t^{-a})) \pi_u^a(u_t'^a | o_t^a, \vec{\mu}_t^a) \right) \quad (4)$$

4.2 Action Q-function

In order to calculate the action policy advantage, the critic needs to learn the joint action Q-function $Q_u(s, u, \theta_u)$ which is represented by the neural network parameters θ_u. This neural network is trained by minimising the loss function from Equation 5.

$$\mathcal{L}(\theta_u^i) = \mathbb{E}_{s_t, u_t, r_t, s_{t+1}}[(r_t + \gamma Q_u(s_{t+1}, u_{t+1}, \theta_u^{i-}) - Q_u(s_t, u_t, \theta_u^i))^2] \quad (5)$$

The loss function uses a target network θ_u^{i-} [17], which is a delayed version of the θ_u^i parameters, in order to stabilize the training of the action Q-function. The loss function uses the observed future actions u_{t+1} (the SARSA update rule as described by Rummery and Niranjan [22], Sutton and Barto [27]) instead of using the actions that maximize the Q-function ($\max_{u'_{t+1}} Q_u(s_{t+1}, u'_{t+1}, \theta_u)$) in order to minimize the number of inference calls to the action Q-network when the joint action space is large (due to the large number of action permutation over the different agents). In this work, we use a replay buffer to reuse past experiences to train the joint action Q-function. However, the joint action loss function is on-policy which cannot be used in combination with a replay buffer unless the actions $\widetilde{u}_{t+1}$ are resampled based on the current policy $\widetilde{u}_{t+1} \sim \pi_u(o_{t+1}, M(\widetilde{m}_t))$. Additionally, the MACC action policies can depend on the received messages which depend on the communication policy. This means that the message $\widetilde{m}_t$ also needs to be resampled from the communication policies $\widetilde{m}_t \sim \pi_c(o_t, \mu_t)$.

4.3 Counterfactual Reasoning in the Communication Environment

We adapted the counterfactual reasoning process to work within the communication environment. This is shown in Equation 6 and 7. First, the state s_t is expanded to $s_{t,t+1}$ to include $t+1$ which is required when messages take one timestep to arrive. The $t+1$ timestep can be replaced with t for instant communication and with $t + t_d$ for communication delayed by t_d timesteps. Next, the action Q-function $Q_u(s_t, u_t)$ is replaced by a communication Q-function $Q_c(s_{t,t+1}, m_t)$.

$$A_c^a(s_{t,t+1}, \vec{\mu}_t^a, m_t) = Q_c(s_{t,t+1}, m_t) - V_c^a(s_{t,t+1}, \vec{\mu}_t^a, m_t^{-a}) \quad (6)$$

$$V_c^a(s_{t,t+1}, \vec{\mu}_t^a, m_t^{-a}) = \sum_{m_t'^a} \left(Q_c(s_{t,t+1}, (m_t'^a, m_t^{-a})) \pi_c^a(m_t'^a | o_t^a, \vec{\mu}_t^a) \right) \quad (7)$$

However, the communication Q-function Q_c cannot be learned as the communication environment appears non-stationary because the utility of a message changes when the action policy of the other agent changes during the training process. In this work, we created the communication Q-function Q_c analytically using the action policies of the other agents and the centralized action Q-function. The communication Q-function Q_c is composed out of the communication policy to the action policy Q-function Q_{cu} (the impact of a message on other action policies) and the discounted communication policy to the communication policy Q-function Q_{cc} (the impact of a message on other communication policies). This is shown in Equation 8. The individual calculations for Q_{cu} are discussed in Section 4.4 and the calculations for Q_{cc} are discussed in Section 4.6.

$$Q_c(s_{t,t+1}, m_t) = Q_{cu}(s_{t,t+1}, m_t) + \gamma_c Q_{cc}(s_{t,t+1}, m_t) \quad (8)$$

The communication discount factor γ_c is used to limit the impact of a message on the communication Q-function. The discount factor γ_c will be equal to the action discount factor γ when using a communication delay of a single timestep. In the general setting the discount factor can be defined using the number of timesteps the communication is delayed t_d and the action discount factor γ as $\gamma_c = \gamma^{t_d}$.

4.4 Communication Policy to Action Policy Q-function

The communication policy to action policy Q-function Q_{cu} represents the expected return when a message is sent to the action policy of the different agents. This expectation is shown in Equation 9.

$$Q_{cu}(s_{t,t+1}, m_t) = \mathbb{E}[Q_u(s_{t,t+1}, u'_{t+1}); \pi_u(u'_{t+1} | o_{t+1}, M(m_t))] \quad (9)$$

We can uses this expectation to define an exact function for the communication policy to action policy Q-function Q_{cu} as shown in Equation 10 which is used in MACC Exact. Intuitively, we will iterate over all possible action permutations u'_{t+1} and multiply the utility for a joint state action pair $Q_u(s_{t,t+1}, u'_{t+1})$ with the probability of the joint action being selected $\pi_u(u'_{t+1} | o_{t+1}, M(m_t))$ given the input messages m_t. The state and actions used in this communication are from $t+1$ as we used one step delayed communication.

$$Q_{cu}(s_{t,t+1}, m_t) = \sum_{u'_{t+1}} \left(Q_u(s_{t+1}, u'_{t+1}) \pi_u(u'_{t+1} | o_{t+1}, M(m_t)) \right) \quad (10)$$

However, the computation cost of the exact MACC method rises exponentially with the number of agents as the exact method needs to iterate over all possible action permutations of the other agents. In the next section, we discuss two methods to calculate the communication policy to action policy Q-function Q_{cu} using approximations to reduce this computational cost.

4.5 Approximation Methods

In this section, the two approximation methods for the communication policy to action policy Q-function Q_{cu} are discussed. These methods aim to reduce the computational cost of counterfactual reasoning for the communication policy for a higher number of agents.

4.5.1 Sample Mean method. The Sample Mean method allows us to sample the communication to action policy Q-function under the distribution π_u using n samples (see Equation 11). When using a sufficiently high number of samples, the sample mean method will approximate the communication policy to action policy Q-function $Q_{cu} \approx \hat{Q}_{cu}$.

$$\hat{Q}_{cu}(s_{t,t+1}, m_t) = \frac{1}{n} \sum_{i=0}^{n} Q_u(s_{t+1}, \widetilde{u}_{t+1}) \quad \text{with: } \widetilde{u}_{t+1} \sim \pi_u(o_{t+1}, M(m_t)) \quad (11)$$

The variance of this estimation will depend on the variance of Q_u under π_u and the number of samples n as shown in Equation 12. The exact number of samples needs to be determined empirically.

$$VAR[\hat{Q}_{cu}] = \frac{VAR[Q_u; \pi_u]}{n} \tag{12}$$

4.5.2 Agent Based Sampling Method. The Agent Based Sampling method (ABS) uses the decomposition of the Q-function expectation with respect to the joint action policy π_u. This decomposition will allow us to compute the two expectations using different methods to find a balance between the exact method and the sampling mean method.

Lemma 4.1. *The communication policy to action policy Q-function Q_{cu} expectation under π_u can be decomposed into a Q-function expectation with respect for $\pi_u^{a'}$ and $\pi_u^{-a'}$ for any agent a'.*

$$\begin{aligned} &Q_{cu}(s_{t,t+1}, m_t) = \\ &\mathbb{E}[\mathbb{E}[Q_u(s_{t,t+1}, (u'^{a'}_{t+1}, u'^{-a'}_{t+1})); \pi_u^{a'}(u'^{a'}_{t+1}|o^{a'}_{t+1}, M^{a'}(m_t))]; \\ &\quad \pi_u^{-a'}(u'^{-a'}_{t+1}|o^{-a'}_{t+1}, M^{-a'}(m_t))] \end{aligned} \tag{13}$$

Proof.

$$\begin{aligned} &Q_{cu}(s_{t,t+1}, m_t) \\ &\quad = \mathbb{E}[Q_u(s_{t,t+1}, u'_{t+1}); \pi_u(u'_{t+1}|o_{t+1}, M(m_t)] \\ &\quad = \sum_{u'_{t+1}} \left(Q_u(s_{t+1}, u'_{t+1})\pi_u(u'_{t+1}|o_{t+1}, M(m_t))\right) \\ &\quad = \sum_{u'^{-a'}_{t+1}} \sum_{u'^{a'}_{t+1}} \left(Q_u(s_{t+1}, (u'^{a'}_{t+1}, u'^{-a'}_{t+1}))\cdot\right. \\ &\quad \left. \pi_u^{a'}(u'^{a'}_{t+1}|o^{a'}_{t+1}, M^{a'}(m_t))\pi_u^{-a'}(u'^{-a'}_{t+1}|o^{-a'}_{t+1}, M^{-a'}(m_t))\right) \\ &\quad = \sum_{u'^{-a'}_{t+1}} \left(\mathbb{E}[Q_u(s_{t,t+1}, (u'^{a'}_{t+1}, u'^{-a'}_{t+1})); \pi_u^{a'}(u'^{a'}_{t+1}|o^{a'}_{t+1}, M^{a'}(m_t))]\right. \\ &\quad \left. \pi_u^{-a'}(u'^{-a'}_{t+1}|o^{-a'}_{t+1}, M^{-a'}(m_t))\right) \\ &\quad = \mathbb{E}[\mathbb{E}[Q_u(s_{t,t+1}, (u'^{a'}_{t+1}, u'^{-a'}_{t+1})); \pi_u^{a'}(u'^{a'}_{t+1}|o^{a'}_{t+1}, M^{a'}(m_t))]; \\ &\quad \pi_u^{-a'}(u'^{-a'}_{t+1}|o^{-a'}_{t+1}, M^{-a'}(m_t))] \end{aligned} \tag{14}$$

□

The sampling mean will be used for the expectation over the other agents $-a'$ and the expectation over agent a' will be calculated using the exact method as show in Equation 15. When using a sufficiently large number of samples n the agent based sampling estimation will approximate communication policy to action policy Q-function $Q_{cu} \approx \hat{Q}_{cu}$. Note that we select a different agent for every sample in order to prevent a certain bias towards any agent. Intuitively, this is similar to how humans will reason about the reaction of other humans about a certain event. They will reason about how every individual will react instead of going over all the possible permutations. Note that this method will also be used to efficiently calculate the communication policy to communication policy Q-function Q_{cc} as discussed in the next section.

$$\begin{aligned} \hat{Q}_{cu}(s_{t,t+1}, m_t) &= \frac{1}{n}\sum_{a'=0}^{n} \mathbb{E}[Q_u(s_{t,t+1}, (u'^{a'}_{t+1}, u'^{-a'}_{t+1})); \\ &\qquad \pi_u^{a'}(u'^{a'}_{t+1}|o^{a'}_{t+1}, M^{a'}(m_t))] \\ &= \frac{1}{n}\sum_{a'=0}^{n} \sum_{u'^{a'}_{t+1}} Q_u(s_{t+1}, (u'^{a'}_{t+1}, \tilde{u}^{-a'}_{t+1}))\cdot \\ &\qquad \pi_u^{a'}(u^{a'}_{t+1}|o^{a'}_{t+1}, M^{a'}(m_t)) \\ &\qquad \text{with: } \tilde{u}^{-a'}_{t+1} \sim \pi_u^{-a'}(o^{-a'}_{t+1}, M^{-a'}(m_t)) \end{aligned} \tag{15}$$

The variance of the ABS estimation is shown in Equation 16. Note that this variance will be lower as the variance will be under $\pi_u^{-a'}$ instead of under π_u. However, a sample from the ABS method is more computationally expensive. Nevertheless, we hypothesise that the variance will be reduced because we can directly use the distribution $\pi_u^{a'}$ in our estimation instead of sampling this distribution.

$$VAR[\hat{Q}_{cu}] = \frac{VAR[Q_{cu}; \pi_u^{-a'}]}{n} \tag{16}$$

In our experiments the number of samples n will be equal to the amount of agents. Hence, the computational cost grows linearly instead of exponentially when using the exact method.

4.6 Communication Policy to Communication Policy Q-function

The communication policy to communication policy Q-function Q_{cc} is defined as shown in Equation 17.

$$Q_{cc}(s_{t,t+1}, m_t) = \mathbb{E}[Q_c(s_{t+1,t+2}, m'_{t+1}); \pi_c(m'_{t+1}|o_{t+1}, M(m_t))] \tag{17}$$

However this poses a computational challenge as in order to compute Q_{cc} at timestep t, we need to calculate Q_c at timestep $t+1$ for which we again need to calculate Q_{cc} at timestep $t+1$. This can be overcome by starting the calculations for a certain episode (with an episode length of T) at timestep $t = T-2$ (as $Q_{cc} = 0$ at timestep $t = T-1$ because of the one step communication delay) and calculate Q_{cc} backwards towards $t = 0$. Algorithm 1 shows the full MACC algorithm including the Q_{cc} calculations. Note that we cannot use the exact method or the sampling mean method because we do not calculate the Q_c value for every message combination m'_{t+1} but only for the alternative message $m'^{a'}_{t+1}$ from the different agents a'. This means that we need to use the agent based sampling method as defined in section 4.5.2 which leads to the Equation 18.

$$\begin{aligned} Q_{cc}(s_{t,t+1}, m_t) \approx \frac{1}{n}\sum_{a'=0}^{n} \sum_{m'^{a'}_{t+1}} &\left(Q_c(s_{t+1,t+2}, (m'^{a'}_{t+1}, m^{-a'}_{t+1}))\cdot\right. \\ &\left. \pi_c^{a'}(m'^{a'}_{t+1}|o^{a'}_{t+1}, M^{a'}(m_t))\right) \end{aligned} \tag{18}$$

4.7 Social Loss

The MACC methods use the change in the future expected reward through the choice of alternative actions in function of an input message to learn a communication policy. However, when the action policy ignores the received messages, it becomes impossible to learn a valid communication protocol which in turn does not give the action policy any incentive to use the received messages. In order

Algorithm 1 MACC

Initialise $\theta_u, \theta_u^-, \theta_{\pi_u}, \theta_{\pi_c}$
for each episode e **do**
  s_0 = initial state, t = 1
  $\mu_0^a = 0$ for each agent a
  while $s_t \neq$ terminal **and** $t < T$ **do**
    for each agent a **do**
      $u_t^a \sim \pi_u^a(o_t^a, \mu_t^a, \theta_{\pi_u^a}), m_t^a \sim \pi_c^a(o_t^a, \mu_t^a, \theta_{\pi_c^a})$
    end for
    $\vec{\mu}_{t+1} = M(m_t)$
    Get s_{t+1}, r_t from the environment
    $t = t + 1$
  end while
  Calculate Q_u^a, Q_{cu}^a for each agent a
  $Q_{c,t-1}^a = 0$ for each agent a
  for j = t - 2 **to** 0 **do**
    for each agent a **do**
      Calculate $Q_{cc,j}^a$ using $Q_{cu,j+1}^a, Q_{c,j+1}^a$
      $Q_{c,j} = Q_{cu,j} + \gamma_c Q_{cc,j}$
    end for
  end for
  Calculate A_u, A_c, V_u, V_c using Q_u, Q_c
  Update θ_u, θ_u^- using r and $\theta_{\pi_u}, \theta_{\pi_c}$ using A_u, A_c
end for

to promote social behaviour from the action policy, which in this context means taking different actions based on the messages of other agents, an optional social loss function $\mathcal{L}^s$ is added to the action policy loss function. This loss function encourages the action policy to have a different policy distribution for different received messages $\vec{\mu}'$. The social loss function is shown in Equation 19. This loss function iterates over the different input bits and compares the output distribution, given the original message with the output distribution, with the given input message with a bit flip $\neg\vec{\mu}_x$ for the selected bit x. The social loss function can be adapted to use different loss functions like the Kullback-Liebler loss function. However, we found that the absolute difference performed better due to the linear nature of the loss function.

$$\mathcal{L}^s(\theta^i_{\pi_u^a}) = -\frac{\lambda}{k}\sum_{x=0}^{k} |\pi_u^a(o^a, \vec{\mu}, \theta^i_{\pi_u^a}) - \pi_u^a(o^a, (\neg\vec{\mu}_x, \vec{\mu}_{-x}), \theta^i_{\pi_u^a})| \tag{19}$$

Note that this social loss is an optional loss function which can be controlled by the hyperparameter λ which allows us to tune the social loss loss in order to encourage social behaviour without removing the possibility for the agent to ignore the messages when this is required in a certain environment. Additionally, if the sending agent cannot improve the team reward by sending different messages, the communication policy learns to always send the same message in order to promote the best action distribution.

5 EXPERIMENTS

In these experiments, MACC is evaluated with and without the social loss ($\lambda = 0$), with different approximation methods and compared with COMA [7] and the RLlib MADDPG [13] implementation because these methods target the credit assignment problem, use a centralized critic and can be used to train discrete non-differentiable inter-agent communication. The RLlib framework [11] is used to train the different methods efficiently. We used the same hyperparameters for MADDPG as in the work of Lowe et al. [13] and determined the hyperparameters for the other methods empirically and by using a grid search. The results for these methods are the average of five training runs for every method with the best performing run and the least performing run removed. The methods are evaluated on multiple scenarios for both the particle environment [13] and on the custom matrix environment. These scenarios are selected to validate different multi-agent communication configurations.

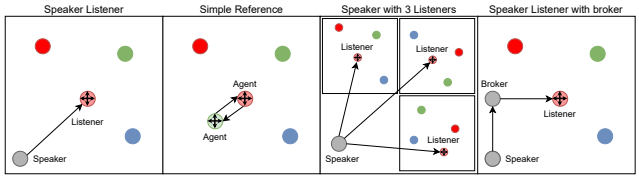


Figure 3: The four different Particle environment scenarios in which the arrows represent the communication topology.

5.1 Particle Environment

Our experiments use different scenarios from the particle environment [13] (see Figure 3) where multiple agents are simulated in a 2D environment. We use the Speaker Listener scenario and the Simple Reference scenario from this environment. The goal in these scenarios is that the agent or both agents, for the Simple Reference environment, move towards the goal landmark. But the agents do not know which of the landmarks is the target landmark as this information is only available to the other agent. The agents need to learn to communicate this information and to use this information to move to the target landmark. Additionally, we propose two more complex novel scenarios, the Speaker with 3 Listeners scenario and the Speaker Listener with Communication Broker scenario. The Speaker with 3 Listeners scenario extends the Speaker Listener scenario with additional listeners that have to navigate to their own landmark. These agents will have the same target landmark color but the landmarks have a different location. The Speaker Listener with Communication Broker scenario is also an extension of the Speaker Listener scenario in which an additional agent will learn to pass messages from the speaker to the listener as the speaker and listener are not allowed to communicate directly increasing the communication complexity. The agents in these different scenarios use a shared team reward which represents the average distance between the agents and their the target landmark.

5.1.1 Speaker Listener. The training results of the different methods for the Speaker Listener scenario are shown in Figure 4 and Table 1. These results show that every MACC configuration is able to outperform both COMA and MADDPG. They are able to learn a basic communication protocol (a higher reward than -35) but cannot achieve a reward higher then -25. Note that COMA and MADPPG learn this basic communication faster than MACC is able to learn a communication protocol. This is due to the way these

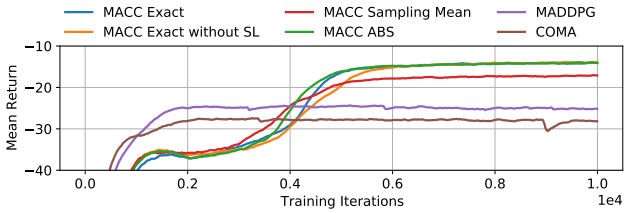


Figure 4: The Speaker Listener training results.

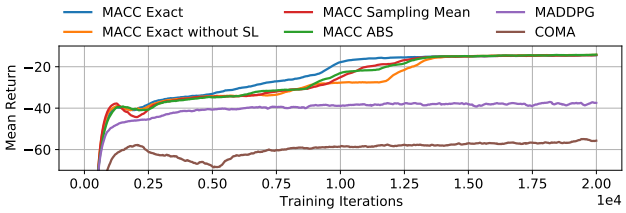


Figure 5: The Simple Reference training results.

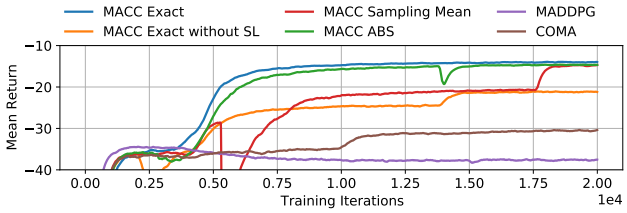


Figure 6: The Speaker with 3 Listeners training results.

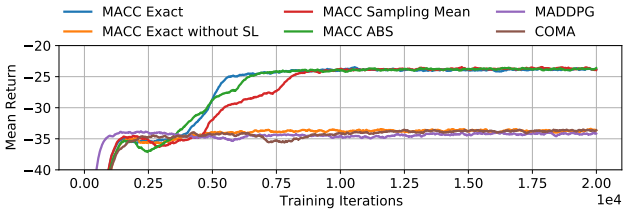


Figure 7: Speaker listener with broker training results.

Table 1: The reward statistics for the final 10% of the training run within the Particle environment.

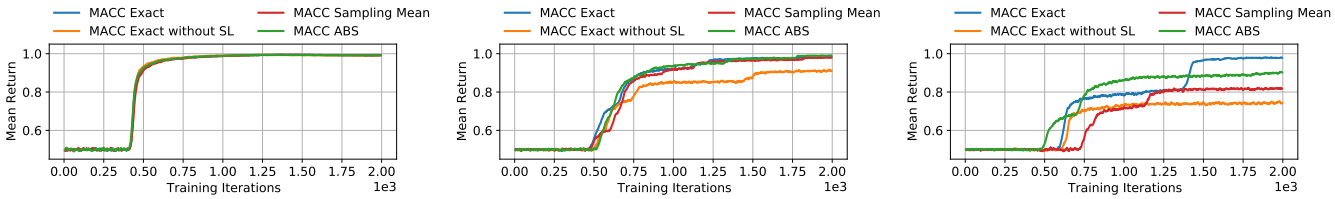
Experiment name	MACC Exact	MACC Exact without SL	MACC Sampling Mean	MACC ABS	COMA	MADDPG
Speaker Listener	-14.10 ± 1.04	-13.95 ± 1.09	-17.15 ± 4.55	-14.11 ± 1.12	-28.17 ± 8.58	-25.19 ± 7.20
Simple Reference	-14.56 ± 0.81	-14.39 ± 0.76	-14.51 ± 0.81	-14.32 ± 0.79	-56.03 ± 2.85	-37.80 ± 2.47
Speaker with 3 Listeners	-14.01 ± 0.72	-21.18 ± 3.62	-14.82 ± 0.72	-14.66 ± 0.81	-30.51 ± 4.85	-37.62 ± 1.25
Speaker Listener with broker	-23.94 ± 1.50	-33.74 ± 1.76	-23.71 ± 1.42	-23.74 ± 1.48	-33.79 ± 1.71	-34.21 ± 1.85

methods learn their communication protocol. COMA and MADDPG learn their communication protocol directly from the critic while MACC will use the critic in combination with the policies of the other agents to learn a communication protocol. The policies of the other agents need a certain amount of time to learn to change their behaviour based on the input message. However MACC is able to learn a valid communication protocol and achieve a reward of -15. MACC using the exact method and MACC ABS achieve a very similar learning behaviour while MACC Exact without the social loss function takes longer to learn the communication protocol and MACC Sampling Mean achieves a lower overal reward

5.1.2 Simple Reference. The results for the simple reference scenario are presented in Figure 5 and Table 1. MADDPG is able to successfully learn a valid action policy but is not able to learn a communication protocol since the average reward is too low. COMA is not able to learn a valid action or communication policy because of learning instability caused by the non-stationary communication environment, the credit assignment problem and the large action space of the agents. The large actions space is the number of actions times the number of possible output messages ($5 * 2^2 = 20$) as COMA does not have a separate communication policy. However, since MACC splits the action and communication policy, the different versions of MACC are able to learn a valid communication protocol. These results show that MACC Exact is able to achieve the maximum reward the fastest. The Sampling Mean and ABS approximations are performing similar but require more training iterations to achieve the maximum reward. Finally, MACC Exact without social loss takes the most training iterations to achieve the maximum reward due to the policies taking longer to change their action or message distribution based on the received message.

5.1.3 Speaker with 3 Listeners. The results for the Speaker with 3 Listeners scenario are shown in Figure 6 and Table 1. The COMA and MADDPG methods are not able to achieve the same results as in the Speaker Listener scenario. As both the action and communication protocol are learned based on the centralized critic, we see that COMA and MADDPG are not able to scale to learning a protocol with multiple receiving agents. MACC Exact without social loss is not able to achieve the maximum reward as the method reaches a local optimum because the action policies are not encouraged to learn social behaviour. The other MACC variants are able to learn the maximum reward. Note that the MACC Sampling Mean method shows a more unstable learning process compared to MACC Exact and MACC ABS. MACC Sampling Mean will sample actions from all the other agents which can lead to a high variance when the amount of agents increases.

5.1.4 Speaker Listener with Communication Broker. The results for the speaker listener with communication broker are shown in Figure 7 and Table 1. In this experiment, COMA, MADDPG and



(a) 2 agents (b) 4 agents (c) 6 agents

Figure 8: The matrix environment for different number of agents.

Table 2: The reward statistics for the final 10% of the training run within the Matrix environment.

Experiment name	MACC Exact	MACC Exact without SL	MACC Sampling Mean	MACC ABS
Matrix with 2 agents	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
Matrix with 4 agents	0.98 ± 0.02	0.91 ± 0.11	0.98 ± 0.02	0.99 ± 0.00
Matrix with 6 agents	0.98 ± 0.02	0.74 ± 0.20	0.82 ± 0.13	0.90 ± 0.11

MACC Exact without SL are not able to learn a valid communication protocol. The other MACC variants are able to learn a basic communication protocol but can only achieve a maximum reward of -25. The MACC Exact and MACC ABS are performing very similary while MACC Sample Mean takes a lot more training iterations to achieve the same reward.

5.2 Matrix Environment

The Matrix environment is based on the work of Lowe et al. [12]. Here, every individual agent will receive a distinct natural number within a certain range (0 or 1 in this experiment) with a 50% chance that all the agents received the same number. The goal of the agent is to determine if they received the same number or not. For every agent that correctly determines if the agents have the same number, the agents receives a team reward of $1/n$ for a configuration with n agents. So if all the agents are correct, a team reward of 1 is received. This scenario is a many to many communication topology which increases the communication complexity.

The goal of this experiment is to increase the number of agents in order to study the scalability of the different MACC versions. In this experiment, we validate three different configurations for the Matrix environment (2, 4 and 6 agents). As the policies of the different agents are identical to each other, we share the parameters between the different agents. The results for the different configurations in the Matrix environment are shown in Figure 8 and Table 2. These results show the scaling of the different MACC versions with an increasing number of agents. The Exact MACC method achieves the highest reward over the different configurations. The more computationally efficient MACC ABS method achieves similar results as the exact version. The Sampling Mean MACC achieves slightly lower performance although it has the same computational cost as the ABS version. Finally, the MACC Exact without social loss learns a basic communication protocol (the reward is higher than 0.5) but achieves the lowest average reward. This shows the importance of the social loss in a setting with a higher number of agents in order to promote social behaviour.

6 CONCLUSION

In this paper, we present a novel method for multi-agent communication learning called MACC which is able to reason about the impact of a certain action or message using counterfactual reasoning by using a centralised critic. The critic is able to reason about the impact of a certain message on both the action and communication policy. Additionally, we also introduce two approximation methods for MACC and a social loss function in order to improve the social behaviour. In the experiments, the different variants of MACC are compared to MADDPG and COMA, which also use a centralized critic, in four different scenarios in the Particle environment. These experiments show that the MACC variants are able to learn a valid communication protocol in a range of different communication scenarios and can outperform both MADDPG and COMA. The social loss can be used by MACC to promote social behaviour in the receiving agent. Additionally, we show that the approximation methods are able to learn a valid communication protocol. We observed that MACC Agent Based Sampling is able to slightly outperform MACC Sampling Mean as the number of agents is increased. Next, the scaling performance of the MACC variants are validated on the Matrix environment. These results show that MACC Exact and MACC Agent Based Sampling outperform the other MACC variants when the number of agents is increased. In this work, we also assumed to have full access to the policy of every agent. This requirement could be relaxed by learning a model of the policy of the other agents. Additionally, we also believe that the social loss function could be improved in order to further limit the chance of getting stuck in a local optimum, which is still possible in certain environments as demonstrated in the Speaker Listener with broker agent environment.

ACKNOWLEDGMENTS

This work was supported by the Research Foundation Flanders (FWO) under Grant Number 1S94120N and Grant Number 1S12121N. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] L. Busoniu, R. Babuska, and B. De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[2] Yu-Han Chang, Tracey Ho, and Leslie P Kaelbling. 2004. All learning is local: Multi-agent learning in global reward games. (2004).

[3] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. 2018. TarMAC: Targeted Multi-Agent Communication. arXiv:1810.11187 [cs.LG]

[4] Ziluo Ding, Tiejun Huang, and Zongqing Lu. 2020. Learning individually inferred communication for multi-agent cooperation. *arXiv preprint arXiv:2006.06455* (2020).

[5] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*. 2137–2145.

[6] 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 *International conference on machine learning*. PMLR, 1146–1155.

[7] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*.

[8] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. 2019. Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *International Conference on Machine Learning*. 3040–3049.

[9] Jiechuan Jiang and Zongqing Lu. 2018. Learning Attentional Communication for Multi-Agent Cooperation. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7254–7264. http://papers.nips.cc/paper/7956-learning-attentional-communication-for-multi-agent-cooperation.pdf

[10] Emilio Jorge, Mikael Kågebäck, Fredrik D Johansson, and Emil Gustavsson. 2016. Learning to play guess who? and inventing a grounded language as a consequence. *arXiv preprint arXiv:1611.03218* (2016).

[11] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[12] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. 2019. On the pitfalls of measuring emergent communication. *arXiv preprint arXiv:1903.05168* (2019).

[13] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)* (2017).

[14] Hangyu Mao, Zhibo Gong, Yan Ni, and Zhen Xiao. 2017. ACCNet: Actor-Coordinator-Critic Net for "Learning-to-Communicate" with Deep Multi-agent Reinforcement Learning. arXiv:1706.03235 [cs.AI]

[15] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs.LG]

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[17] 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), 529–533.

[18] Igor Mordatch and Pieter Abbeel. 2017. Emergence of Grounded Compositional Language in Multi-Agent Populations. *arXiv preprint arXiv:1703.04908* (2017).

[19] Frans A Oliehoek, Christopher Amato, et al. 2016. *A concise introduction to decentralized POMDPs*. Vol. 1. Springer.

[20] Marie Ossenkopf, Mackenzie Jorgensen, and Kurt Geihs. 2019. Hierarchical Multi-Agent Deep Reinforcement Learning to Develop Long-Term Coordination. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (Limassol, Cyprus) *(SAC '19)*. Association for Computing Machinery, New York, NY, USA, 922–929. https://doi.org/10.1145/3297280.3297371

[21] 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. *CoRR* abs/1703.10069 (2017). arXiv:1703.10069 http://arxiv.org/abs/1703.10069

[22] Gavin A Rummery and Mahesan Niranjan. 1994. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK.

[23] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*. PMLR, 387–395.

[24] David Simões, Nuno Lau, and Luís Paulo Reis. 2020. Multi-agent actor centralized-critic with communication. *Neurocomputing* 390 (2020), 40–56. https://doi.org/10.1016/j.neucom.2020.01.079

[25] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf

[26] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2085–2087.

[27] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press, Cambridge, MA, USA.

[28] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. arXiv:1509.06461 [cs.LG]

[29] Simon Vanneste, Astrid Vanneste, Stig Bosmans, Siegfried Mercelis, and Peter Hellinckx. 2020. Learning to Communicate with Multi-agent Reinforcement Learning Using Value-Decomposition Networks. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, Leonard Barolli, Peter Hellinckx, and Juggapong Natwichai (Eds.). Springer International Publishing, Cham, 736–745.

[30] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2015. Dueling Network Architectures for Deep Reinforcement Learning. arXiv:1511.06581 [cs.LG]