# IMPROVING COORDINATION IN SMALL-SCALE MULTI-AGENT DEEP REINFORCEMENT LEARNING THROUGH MEMORY-DRIVEN COMMUNICATION

**Emanuele Pesce**
WMG,University of Warwick
Coventry, CV4 7AL
e.pesce@warwick.ac.uk

**Giovanni Montana**
WMG,University of Warwick
Coventry, CV4 7AL
g.montana@warwick.ac.uk

**ABSTRACT**

Deep reinforcement learning algorithms have recently been used to train multiple interacting agents in a centralised manner whilst keeping their execution decentralised. When the agents can only acquire partial observations and are faced with tasks requiring coordination and synchronisation skills, inter-agent communication plays an essential role. In this work, we propose a framework for multi-agent training using deep deterministic policy gradients that enables concurrent, end-to-end learning of an explicit communication protocol through a memory device. During training, the agents learn to perform read and write operations enabling them to infer a shared representation of the world. We empirically demonstrate that concurrent learning of the communication device and individual policies can improve inter-agent coordination and performance in small-scale systems. Our experimental results show that the proposed method achieves superior performance in scenarios with up to six agents. We illustrate how different communication patterns can emerge on six different tasks of increasing complexity. Furthermore, we study the effects of corrupting the communication channel, provide a visualisation of the time-varying memory content as the underlying task is being solved and validate the building blocks of the proposed memory device through ablation studies.

## 1 Introduction

Reinforcement Learning (RL) allows agents to learn how to map observations to actions through feedback reward signals (Sutton and Barto, 1998). Recently, deep neural networks (LeCun et al., 2015; Schmidhuber, 2015) have had a noticeable impact on RL (Li, 2017). They provide flexible models for learning value functions and policies, allow to overcome difficulties related to large state spaces, and eliminate the need for hand-crafted features and ad-hoc heuristics (Cortes et al., 2002; Parker et al., 2003; Olfati-Saber et al., 2007). Deep reinforcement learning (DRL) algorithms, which usually rely on deep neural networks to approximate functions, have been successfully employed in single-agent systems, including video game playing (Mnih et al., 2015), robot locomotion (Lillicrap et al., 2015), object localisation (Caicedo and Lazebnik, 2015) and data-center cooling (Evans and Gao, 2016).

Following the uptake of DRL in single-agent domains, there is now a need to develop improved learning algorithms for multi-agent (MA) systems where additional challenges arise. Markov Decision Processes, upon which DRL methods rely, assume that the reward distribution and dynamics are stationary (Hernandez-Leal et al., 2017). When multiple learners interact with each other, this property is violated because the reward that an agent receives also depends on other agents' actions (Laurent et al., 2011). This issue, known as the *moving-target* problem (Tuyls and Weiss, 2012), removes convergence guarantees and introduces additional learning instabilities. Further difficulties arise from environments characterized by partial observability (Singh et al., 1994; Chu and Ye, 2017; Peshkin et al., 2000) whereby the agents do not have full access to the world state, and where coordination skills are essential.

An important challenge in multi-agent DRL is how to facilitate communication amongst interacting agents. Communication is widely known to play a critical role in promoting coordination between humans (Számadó, 2010). Humans have been proven to excel at communicating even in absence of a conventional code (De Ruiter et al., 2010).

When coordination is required and no common languages exist, simple communication protocols are likely to emerge (Selten and Warglien, 2007). Human communication involves more than sending and receiving messages, it requires specialized interactive intelligence where receivers have the ability to recognize intentions and senders can properly design messages (Wharton, 2003). The emergence of communication has been widely investigated (Garrod et al., 2010; Theisen et al., 2010), for example new signs and symbols can emerge when it comes to represent real concepts. Fusaroli et al. (2012) demonstrated that language can be seen as a social coordination device learnt through reciprocal interaction with the environment for optimizing coordinative dynamics. The relation between communication and coordination has been widely discussed (Vorobeychik et al., 2017; Demichelis and Weibull, 2008; Miller and Moser, 2004; Kearns, 2012). Communication is an essential skill in many tasks: for instance, in critical situations, where is of fundamental importance to properly manage critical and urgent situations, like emergency response organizations (Comfort, 2007). In multiplayer videogames, it is often essential to reach a sufficiently high level of coordination required to succeed (Chen, 2009).

Two-agents systems have often been studied when looking at the effects of communication on coordination. Galantucci (2005) showed that humans can easily produce new protocols to overcame the lack of a common language, through experiments in which pairs of participants playing video games were allowed to send messages through a medium that prevented the use of standard symbols. In two-players games, like the Battle of the Sexes, improved coordination resulted when allowing gamers to exchange messages (Cooper et al., 1989). Human conversations can be interpreted as a bi-directional communication form, where the same entity can both send and receive messages (Lasswell, 1948). This kind of communication can be efficiently explored in small-scale systems through coordination games (Cooper et al., 1992) and often it is the key to achieve success in real world scenarios such as bargaining with incomplete information (Brosig et al., 2003).

Analogously, the importance of communication has been recognised when designing artificial MA learning systems, especially in tasks requiring synchronization (Scardovi and Sepulchre, 2008; Wen et al., 2012). For example, in navigation tasks, agents can localise each other more easily through shared information (Fox et al., 2000). In group strategy coordination, such as automating negotiations, communication is fundamental to improve the final outcome (Wunder et al., 2009; Itō et al., 2011). A wide range of MA applications have benefitted from inter-agent message passing including distributed smart grid control (Pipattanasomporn et al., 2009), consensus in networks (You and Xie, 2011), multi-robot control (Ren and Sorensen, 2008), autonomous vehicle driving (Petrillo et al., 2018), elevators control (Crites and Barto, 1998), soccer-playing robots (Stone and Veloso, 1998) and for language learning in two-agent systems (Lazaridou et al., 2016).

Recently, Lowe et al. (2017) have proposed MADDPG (Multi-Agent Deep Deterministic Policy Gradient). Their approach extends the actor-critic algorithm (Degris et al., 2012) in which each agent has an actor to select actions and a critic to evaluate them. MADDPG embraces the centralised learning and decentralised execution paradigm (CLDE) (Foerster et al., 2016; Kraemer and Banerjee, 2016; Oliehoek and Vlassis, 2007). During centralised training, the critics receive observations and actions from all the agents whilst the actors only see their local observations. On the other hand, the execution only relies on actors. This approach has been designed to address the emergence of environment non-stationarity (Tuyls and Weiss, 2012; Laurent et al., 2011) and has been shown to perform well in a number of mixed competitive and cooperative environments. In MADDPG, the agents can only share each other's actions and observations during training through their critics, but do not have the means to develop an explicit form of communication through their experiences. The input size of each critic increases linearly with the number of agents (Lowe et al., 2017), which hinders its scalability (Jiang and Lu, 2018).

In this article, we consider tasks requiring strong coordination and synchronization skills. In order to thoroughly study the effects of communication on these scenarios, we focus on small-scale systems. This allows us to design tasks with an increasing level of complexity, and simplifies the investigation of possible correlations between the level of messages being exchanged and any environmental changes. We provide empirical evidences that the proposed method reaches very good performance on a range of two-agent scenarios when a high level of cooperation is required. In addition, we present experimental results for systems with up to six agents in the Supplementary Material (Section B.2 and B.3). In the real world, there is range of applications involving the coordination of only a few actors, for example motor interactions like sawing or cooperative lifting of heavy loads (Jarrassé et al., 2012).

In such cases, being able to communicate information beyond the private observations, and infer a shared representation of the world through interactions, becomes essential. Ideally, an agent should be able to remember its current and past experience generated when interacting with the environment, learn how to compactly represent these experiences in an appropriate manner, and share this information for others to benefit from. Analogously, an agent should be able to learn how to decode the information generated by other agents and leverage it under every environmental state. The above requirements are captured here by introducing a communication mechanism facilitating information sharing within the CLDE paradigm. Specifically, we provide the agents with a shared communication device that can be used to

learn from their collective private observations and share relevant messages with others. Each agent also learns how to decode the memory content in order to improve its own policy. Both the read and write operations are implemented as parametrised, non-linear gating mechanisms that are learned concurrently with the individual policies. When the underlying task to be solved demands for complex coordination skills, we demonstrate that our approach can achieve higher performance compared to the MADDPG baseline in small-scale systems. Furthermore, we demonstrate that being able to learn end-to-end a communication protocol jointly with the policies can also improve upon a *meta-agent* approach whereby all the agents perfectly share all their observations and actions in both training and execution. We investigate a potential interpretation of the communication patterns that have emerged when training two-agent systems through time-varying low-dimensional projections and their visual assessment, and demonstrate how these patterns correlate with the underlying tasks being learned.

This article is organised as follow. In Section 2 a general overview of related work is offered to characterize state-of-the-art approaches for MARL with special focus on communication systems. Section 3 contains the formalization of the problem setup, the details of the proposed method and the description of the learning process; all the experiments are reported in Section 4 where results are presented in terms of numerical metrics to evaluate the performance achieved on six different scenarios; an analysis of the communication channel is provided to support qualitative insights of the exchanged messages. Conclusive comments are given in Section 5. In the Supplementary Material, Section A describes details of MA-MADDPG, a comparative method, and Section B presents a range of additional experiments to further investigate the effects of memory corruption; changes in performance when increasing the number of agents; an ablation study to validate the components used in the proposed method; box plots with the main results, an assessment of the robustness of the method when changing the random seeds; additional analyses of the communication channel.

## 2 Related Work

The problem of reinforcement learning in cooperative environments has been studied extensively (Littman, 1994; Schmidhuber, 1996; Panait and Luke, 2005; Matignon et al., 2007). Early attempts exploited single-agent techniques like Q-learning to train all agents independently (Tan, 1993), but suffered from the excessive size of the state space resulting from having multiple agents. Subsequent improvements were obtained using variations of Q-learning (Ono and Fukumoto, 1996; Guestrin et al., 2002) and distributed approaches (Lauer and Riedmiller, 2000). More recently, DRL techniques like DQN (Mnih et al., 2013) have led to superior performance in single-agents settings by approximating policies through deep neural networks. Tampuu et al. (2017) have demonstrated that an extension of the DQN is able to train multiple agents independently to solve a popular two-agent system, the Pong game. Gupta et al. (2017) have analyzed the performance of popular DRL algorithms, including DQN, DDPG (Lillicrap et al., 2015), TRPO (Schulman et al., 2015) and actor-critic on different MA environments, and have introduced a curriculum learning approach to increase scalability. Foerster et al. (2017) have suggested using a centralized critic for all agents that marginalises out a single's agent action while other agents' actions are kept fixed. Iqbal and Sha (2019) proposed MAAC (Multi-Actor-Attention-Critic), a framework for learning decentralised policies with centralised critics, which selects relevant information for each agent at every time-step through an attention mechanism. In more recent work, a probabilistic recursive reasoning framework has been proposed to model behaviours in a two-agents context; each agent, through variational Bayes methods, approximates the other agent policy to predict its strategy and then to improve its own policy (Wen et al., 2019).

The role of communication in cooperative settings has also been explored, and different methods have been proposed differing on how the communication channels have been formulated using DRL techniques. Many approaches rely on *implicit* communication mechanisms whereby the weights of the neural networks used to implement policies or action-value functions are shared across agents or modelled to allow inter-agent information flow. For instance, in CommNet (Sukhbaatar et al., 2016), the policies are implemented through subsets of units of a large feed-forward neural network mapping the inputs of all agents to actions. At any given time step, the hidden states of each agent are used as messages, averaged and sent as input for the next layer. Singh et al. (2019) proposed IC3NEt, a model designed to improve CommNet, where the hidden states of the agents are also used as messages, but this time they are averaged only after being weighted by a gating mechanism. In addition, in IC3Net, each agent is implemented through an Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) in order to consider the history of the seen observations. In BiCNet (Peng et al., 2017), the agents' policies and value networks are connected through bidirectional neural networks, and trained using an actor-critic approach. Jiang and Lu (2018) proposed an attention mechanism that, when a need for communication emerges, selects which subsets of agents should communicate; the hidden states of their policy networks are integrated through an LSTM to generate a message that is used as input for the next layer of the policy network. Das et al. (2018) utilised a soft attention mechanism to allow the agents to select the recipients of their messages. Each agent, along with the message, broadcasts a signature which can be used to encode agent-specific information. Kong et al. (2017) introduce a master-slave architecture whereby a master agent provides

high-level instructions to organize the slave agents in an attempt to achieve fine-grained optimality. Similarly, in Feudal Multiagent Hierarchies (Ahilan and Dayan, 2019), an agent acts as "manager"and learns to communicate sub-goals to multiple workers operating simultaneously. A different approach is instead provided by the Bayesian Action Decoder (BAD) (Foerster et al., 2018), a technique for two-agent settings where an approximate Bayesian update is used to produce public belief that directly conditions the actions of all agents.

In our work, we introduce a mechanism to generate *explicit* messages capturing relevant aspects of the world, which the agents are able to collectively learn using their observations and interactions. The messages are then sent and received to complement their private observations when making decisions. Some aspects of our approach are somewhat related to DIAL (Differentiable Inter-Agent Learning) (Foerster et al., 2016) in that the communication is enabled by differentiable channels allowing the gradient of the Q-function to bring the proper feedback in small-scale scenarios. Like DIAL, we would like the agents to share explicit messages. However, whereas DIAL uses simple and pre-determined protocols, our agents are given the ability to infer complex protocols from experience, without necessarily relying on pre-defined ones, and utilise those to learn better policies. Explicit messages are also used in SchedNet (Kim et al., 2019) to investigate situations where the bandwidth is limited and only some of the agents are allowed to communicate. In their approach, also focusing on small-case scenarios to better capture the scheduling constraints, the agents produce messages by encoding their observations and a scheduler decides whether an agent is allowed to use a communication channel. A limited bandwidth channel is also used in our work, but all the agents have full access to the channel.

# 3 Memory-driven MADDPG

## 3.1 Problem setup

We consider a system with $N$ interacting agents, where $N$ is typically small, and adopt a multi-agent extension of partially observable Markov decision processes (Littman, 1994). This formulation assumes a set, $\mathcal{S}$, containing all the states characterising the environment; a sequence $\{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_N\}$ where each $\mathcal{A}_i$ is a set of possible actions for the $i^{th}$ agent; a sequence $\{\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_N\}$ where each $\mathcal{O}_i$ contains the observations available to the $i^{th}$ agent. Each $\boldsymbol{o}_i \in \mathcal{O}_i$ provides a partial characterisation of the current state and is private for that agent. Every action $a_i \in \mathcal{A}_i$ is deterministically chosen accordingly to a policy function, $\boldsymbol{\mu}_{\theta_i} : \mathcal{O}_i \mapsto A_i$, parametrised by $\theta_i$. The environment generates a next state according to a transition function, $\mathcal{T} : S \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$, that considers the current state and the $N$ actions taken by the agents. The reward received by an agent, $r_i : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N \mapsto \mathbb{R}$ is a function of states and actions. Each agent learns a policy that maximises the expected discounted future rewards over a period of $T$ time steps, $J(\theta_i) = \mathbb{E}[R_i]$, where $R_i = \sum_{t=0}^{T} \gamma^t r_i(s_i^t, a_i^t)$ is the $\gamma$-discounted sum of future rewards. During training, we would like an agent to learn by using not only its own observations, but through a collectively learned representation of the world that accumulates through experiences coming from all the agents. At the same time, each agent should develop the ability to interpret this shared knowledge in its own unique way as needed to optimise its policy. Finally, the information sharing mechanism would need to be designed in such a way to be used in both training and execution.

## 3.2 Memory-driven communication

We introduce a shared communication mechanism enabling agents to establish a communication protocol through a memory device $\mathcal{M}$ of pre-determined capacity $M$ (Figure 1). The device is designed to store a message $\mathbf{m} \in \mathbb{R}^M$ which progressively captures the collective knowledge of the agents as they interact. An agent's policy becomes $\boldsymbol{\mu}_{\theta_i} : \mathcal{O}_i \times \mathcal{M} \mapsto A_i$, i.e. it is dependent on the agent's private observation as well as the collective memory. Before taking an action, each agent accesses the memory device to initially retrieve and interpret the message left by others. After reading the message, the agent performs a writing operation that updates the memory content. During training, these operations are learned without any *a priori* constraint on the nature of the messages other than the device's size, $M$. During execution, the agents use the communication protocol that they have learned to read and write the memory over an entire episode. We aim to build a model trainable *end-to-end* only through reward signals, and use neural networks as function approximators for policies, and learnable gated functions as mechanisms to facilitate an agent's interactions with the memory. The chosen parametrisations of these operations are presented and discussed below.

**Encoding operation.** Upon receiving its private observations, each agent maps them on to an embedding representing the agent's current vision of the state:

$$\mathbf{e}_i = \varphi^{enc}_{\theta^e_i}(\boldsymbol{o}_i), \qquad \mathbf{e}_i \in \mathbb{R}^E \tag{1}$$

where $\varphi^{enc}_{\theta^e_i}$ is a neural network parametrised by $\theta^e_i$. The embedding $\mathbf{e}_i$ plays a fundamental role in selecting a new action and in the reading and writing phases.

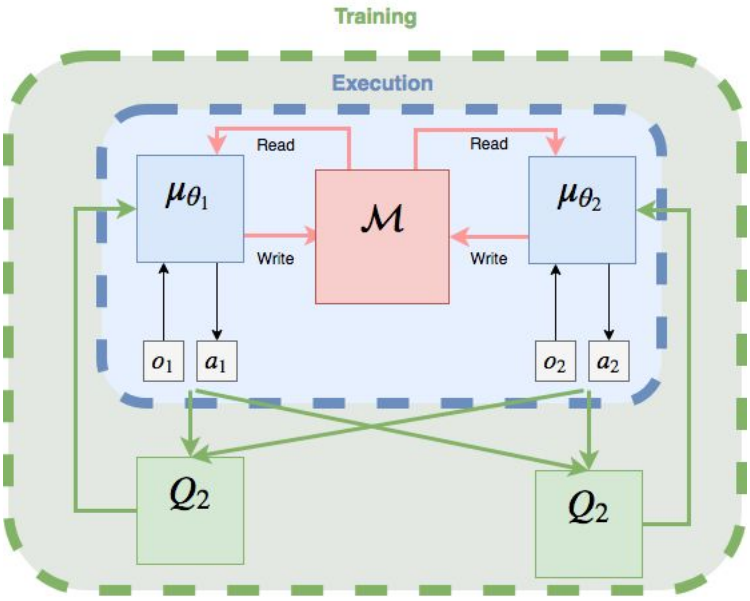



Figure 1: The MD-MADDPG framework. During training and testing, each policy uses its observation and the content of the shared memory to produce a new action and then update the shared channel. Critics are used during training only and each one of them takes as input all the observations and actions.

**Read operation.** After encoding the current information, the agent performs a read operation allowing to extract and interpret relevant knowledge that has been previously captured through $\mathcal{M}$. By interpreting this information content, the agent has access to what other agents have learned. A context vector $\mathbf{h}_i$ is generated to capture spatio-temporal information previously encoded in $\mathbf{e}_i$ through a linear mapping,

$$\mathbf{h}_i = \mathbf{W}_i^h \mathbf{e}_i, \qquad \mathbf{h}_i \in \mathbb{R}^H, \mathbf{W}_i^h \in \mathbb{R}^{H \times E}$$

where $\mathbf{W}_i^h$ represent the learnable weights of the linear projection. While $\mathbf{e}_i$ is defined as general observation encoder, $\mathbf{h}_i$ is specifically designed to extract features for the reading operation. The context vector $\mathbf{h}_i$ can be interpreted as an agent's internal representation that uses the observation embedding $\mathbf{e}_i$ to extract information to be utilized by the gating mechanism only (Eq. 2); its output is then used to extract information from the memory. The main function of the context vector is to facilitate the emergence of an internal representation specifically designed for interpreting the memory content during the read phase. An ablation study aimed at investigating the added benefits introduced by $\mathbf{h}_i$ is provided in the Supplementary Material (B.4). This study supports our intuition that the context vector is crucial for the proper functioning of the entire framework on more complex environments. The agent observation embedding $\mathbf{e}_i$, the reading context vector $\mathbf{h}_i$ and the current memory $\mathbf{m}$ contain different types of information that are used jointly as inputs to learn a gating mechanism,

$$\mathbf{k}_i = \sigma(\mathbf{W}_i^k[\mathbf{e}_i, \mathbf{h}_i, \mathbf{m}]), \qquad \mathbf{k}_i \in [0,1]^M, \mathbf{W}_i^k \in \mathbb{R}^{M \times (E+H+M)} \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function and $[\mathbf{e}_i, \mathbf{h}_i, \mathbf{m}]$ means that the three vectors are concatenated. The values of $\mathbf{k}_i$ are used as weights to modulate the memory content and extract the information from it, i.e.

$$\mathbf{r}_i = \mathbf{m} \odot \mathbf{k}_i \tag{3}$$

where $\odot$ represents the Hadamard product. $\mathbf{k}_i$ takes values in $[0,1]$ and its role is to potentially downgrade the information stored in memory or even completely discard the current content. Learning agent-specific weights $\mathbf{W}_i^h$ and $\mathbf{W}_i^k$ means that each agent is able to interpret $\mathbf{m}$ in its own unique way. As the reading operation strongly depends on the current observation, the interpretation of $\mathbf{m}$ can change from time to time depending on what an agent sees during an episode. Given that $\mathbf{r}_i$ depends on $\mathbf{m}$ and $\mathbf{e}_i$ (from $\boldsymbol{o}_i$ in Eq. 1), we lump all the adjustable parameters into $\theta_i^\zeta = \{\mathbf{W}_i^h, \mathbf{W}_i^k\}$ and write

$$\mathbf{r}_i = \zeta_{\theta_i^\zeta}(\boldsymbol{o}_i, \mathbf{m}). \tag{4}$$

**Write operation.** In the writing phase, an agent decides what information to share and how to properly update the content of the memory whilst taking into account the other agents. The write operation is loosely inspired by the LSTM

(Hochreiter and Schmidhuber, 1997) where the content of the memory is updated through gated functions regulating what information is kept and what is discarded. Initially, the agent generates a candidate memory content, $\mathbf{c}_i$, which depends on its own encoded observations and current shared memory through a non-linear mapping,

$$\mathbf{c}_i = tanh(\mathbf{W}_i^c[\mathbf{e}_i, \mathbf{m}]) \qquad \mathbf{c}_i \in [-1,1]^M, \mathbf{W}_i^c \in \mathbb{R}^{M\times(E+M)}$$

where $\mathbf{W}_i^c$ are weights to learn. An input gate, $\mathbf{g}_i$, contains the values used to regulate the content of this candidate while a forget gate, $\mathbf{f}_i$, is used to decide what to keep and what to discard from the current $\mathbf{m}$. These operations are described as follows:

$$\begin{aligned}\mathbf{g}_i &= \sigma(\mathbf{W}_i^g[\mathbf{e}_i, \mathbf{m}]), \qquad \mathbf{g}_i \in [0,1]^M, \mathbf{W}_i^g \in \mathbb{R}^{M\times(E+M)}\\ \mathbf{f}_i &= \sigma(\mathbf{W}_i^f[\mathbf{e}_i, \mathbf{m}]), \qquad \mathbf{f}_i \in [0,1]^M, \mathbf{W}_i^f \in \mathbb{R}^{M\times(E+M)}.\end{aligned}$$

The $i^{th}$ agent then finally generates an updated message as a weighted linear combination of old and new messages, as follows:

$$\mathbf{m}' = \mathbf{g}_i \odot \mathbf{c}_i + \mathbf{f}_i \odot \mathbf{m}. \tag{5}$$

The update $\mathbf{m}'$ is stored in memory $\mathcal{M}$ and made accessible to other agents. At each time step, agents sequentially read and write the content of the memory using the above procedure. Since $\mathbf{m}'$ depends on $\mathbf{m}$ and $\boldsymbol{e}_i$ (derived from $\boldsymbol{o}_i$ in Eq. 1) we collect all the parameters into $\theta_i^\xi = \{\mathbf{W}_i^c, \mathbf{W}_i^g, \mathbf{W}_i^f\}$ and write the writing operation as:

$$\mathbf{m}' = \xi_{\theta_i^\xi}(\boldsymbol{o}_i, \mathbf{m}). \tag{6}$$

**Action selector.** Upon completing both read and write operations, the agent is able to take an action, $a_i$, which depends on the current encoding of its observations, its own interpretation of the current memory content and its updated version, that is

$$a_i = \varphi_{\theta_i^a}^{act}(\mathbf{e}_i, \mathbf{r}_i, \mathbf{m}') \tag{7}$$

where $\varphi_{\theta_i^a}^{act}$ is a neural network parametrised by $\theta_i^a$. The resulting policy function can be written as a composition of functions:

$$\boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i, \mathbf{m}) = \varphi_{\theta_i^a}^{act}(\varphi_{\theta_i^e}^{enc}(\boldsymbol{o}_i), \zeta_{\theta_i^\zeta}(\boldsymbol{o}_i, \mathbf{m}), \xi_{\theta_i^\xi}(\boldsymbol{o}_i, \mathbf{m})) \tag{8}$$

in which $\theta_i = \{\theta_i^a, \theta_i^e, \theta_i^\zeta, \theta_i^\xi\}$ contains all the relevant parameters.

**Learning algorithm.** All the agent-specific policy parameters, i.e. $\theta_i$, are learned *end-to-end*. We adopt an actor-critic model within a CLDE framework (Foerster et al., 2016; Lowe et al., 2017). In the standard actor-critic model (Degris et al., 2012), we have an actor to select the actions, and a critic, to evaluate the actor moves and provide feedback. In DDPG (Silver et al., 2014; Lillicrap et al., 2015), neural networks are used to approximate both the actor, represented by the policy function $\boldsymbol{\mu}_{\omega_i}$, and its corresponding critic, represented by an action-value function $Q^{\boldsymbol{\mu}_{\omega_i}} : \mathcal{O}_i \times \mathcal{A}_i \mapsto \mathbb{R}$, in order to maximize the objective function $J(\omega_i) = \mathbb{E}[R_i]$. This is done by adjusting the parameters $\omega_i$ in the direction of the gradient of $J(\omega_i)$ which can be written as:

$$\nabla_{\omega_i} J(\omega_i) = \mathbb{E}_{s\sim\mathcal{D}}\big[\nabla_{\omega_i}\boldsymbol{\mu}_{\omega_i}(\boldsymbol{o}_i)\nabla_{a_i} Q^{\boldsymbol{\mu}_{\omega_i}}(\boldsymbol{o}_i, a_i)|_{a_i=\boldsymbol{\mu}_\omega(\boldsymbol{o}_i)}\big]$$

The actions $a$ are produced by the actor $\boldsymbol{\mu}_{\omega_i}$, are evaluated by the critic $Q^{\boldsymbol{\mu}_i}$ which minimises the following loss:

$$\mathcal{L}(\omega_i) = \mathbb{E}_{\boldsymbol{o}_i, a_i, r, \boldsymbol{o}_i'\sim\mathcal{D}}\Big[(Q^{\boldsymbol{\mu}_{\omega_i}}(\boldsymbol{o}_i, a_i) - y)^2\Big]$$

where $\boldsymbol{o}_i'$ is the next observation, $\mathcal{D}$ is an experience replay buffer which contains tuples $(\boldsymbol{o}_i, \boldsymbol{o}_i', a, r)$, $y = r + \gamma Q^{\boldsymbol{\mu}_\omega'}(\boldsymbol{o}_i', a_i')$ represent the target Q-value. $Q^{\boldsymbol{\mu}_{\omega_i}'}$ is a target network whose parameters are periodically updated with the current parameters of $Q^{\boldsymbol{\mu}_{\omega_i}}$ to make training more stable. $\mathcal{L}(\omega_i)$ minimises the expectation of the difference between the current and the target action-state function.

In this formulation, as there is no interaction between agents, the policies are learned independently. We adopt the CLDE paradigm by letting the critics $Q^{\boldsymbol{\mu}_{\omega_i}}$ use the observations $\mathbf{x} = (\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_N)$ and the actions of all agents, hence:

$$\nabla_{\omega_i} J(\boldsymbol{\mu}_{\omega_i}) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}\Big[\nabla_{\omega_i}\boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i)\nabla_{a_i} Q^{\boldsymbol{\mu}_{\omega_i}}(\mathbf{x}, a_1, a_2, \ldots, a_N)|_{a_i=\boldsymbol{\mu}_{\omega_i}(\boldsymbol{o}_i)}\Big] \tag{9}$$

where $\mathcal{D}$ contains transitions in the form of $(\mathbf{x}, \mathbf{x}', a_1, a_2, \ldots, a_N, r_1, \ldots, r_n)$ and $\mathbf{x}' = (\boldsymbol{o}_1', \boldsymbol{o}_2', \ldots, \boldsymbol{o}_N')$ are the next observations of all agents. Accordingly, $Q^{\boldsymbol{\mu}_{\omega_i}}$ is updated as

$$\begin{aligned}\mathcal{L}(\omega_i) &= \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'\sim\mathcal{D}}\Big[(Q^{\boldsymbol{\mu}_{\omega_i}}(\mathbf{x}, a_1, a_2, \ldots, a_N) - y)^2\Big],\\ y &= r_i + \gamma Q^{\boldsymbol{\mu}_{\omega_i}'}(\mathbf{x}', a_1', a_2', \ldots, a_N')\}\end{aligned} \tag{10}$$

in which $a'_1, a'_2, \ldots, a'_N$ are the next actions of all agents. By minimising Eq. 10 the model attempts to improve the estimate of the critic $Q^{\boldsymbol{\mu}_{\omega_i}}$ which is used to improve the policy itselfs through Eq. 9. Since the input of the policy described in Eq. 8 is $(\boldsymbol{o}_i, \mathbf{m})$ the gradient of the resulting algorithm to maximize $J(\theta_i) = \mathbb{E}[R_i]$ can be written as:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_{\theta_i}) = \mathbb{E}_{\mathbf{x},a,\mathbf{m}\sim\mathcal{D}}\Big[\nabla_{\theta_i}\boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i,\mathbf{m})\nabla_{a_i}Q^{\boldsymbol{\mu}_{\theta_i}}(\mathbf{x},a_1,\ldots,a_N)|_{a_i=\boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i,\mathbf{m})}\Big]$$

where $\mathcal{D}$ is a replay buffer which contains transitions in the form of $(\mathbf{x}, \mathbf{x}', a_1, \ldots, a_N, \mathbf{m}, r_1, \ldots, r_n)$. The $Q^{\boldsymbol{\mu}_{\theta_i}}$ function is updated according to Eq. 10. Algorithm 1 provides the pseudo-code of the resulting algorithm, that we call MD-MADDPG (Memory-driven MADDPG).

### 3.3 MD-MADDPG decentralised execution

During execution, only the learned actors $\boldsymbol{\mu}_{\theta_1}, \boldsymbol{\mu}_{\theta_2}, \ldots, \boldsymbol{\mu}_{\theta_N}$ are used to make decisions and select actions. An action is taken in turn by a single agent. The current agent receives its private observations, $\boldsymbol{o}_i$, reads $\mathcal{M}$ to extract $\mathbf{r}_i$ (Eq. 3), generates the new version of $\mathbf{m}$ (Eq. 5), stores it into $\mathcal{M}$ and selects its action $a_i$ using $\boldsymbol{\mu}_i$. The policy of the next agent is then driven by the updated memory.

**Algorithm 1** MD-MADDPG algorithm

Inizialise actors $(\boldsymbol{\mu}_{\theta_1}, \ldots, \boldsymbol{\mu}_{\theta_N})$ and critics networks $(Q_{\theta_1}, \ldots, Q_{\theta_N})$
Inizialise actor target networks $(\boldsymbol{\mu}'_{\theta_1}, \ldots, \boldsymbol{\mu}'_{\theta_N})$ and critic target networks $(Q'_{\theta_1}, \ldots, Q'_{\theta_N})$
Inizialise replay buffer $\mathcal{D}$
**for** episode = 1 to E **do**
    Inizialise a random process $\mathcal{N}$ for exploration
    Inizialise memory device $\mathcal{M}$
    **for** t = 1 to max episode length **do**
        **for** agent $i$ = 1 to $N$ **do**
            Receive observation $\boldsymbol{o}_i$ and the message $\mathbf{m} \leftarrow \mathcal{M}$
            Set $\mathbf{m}_i = \mathbf{m}$
            Generate observation encoding $\mathbf{e}_i$ (Eq. 1)
            Generate read vector $\mathbf{r}_i$ (Eq. 3)
            Generate new message $\mathbf{m}'$ (Eq. 5)
            Generate new time dependant noise istance $\mathcal{N}_t$
            Select action $a_i = \varphi^{act}_{\theta_i}([\mathbf{e}_i, \mathbf{r}_i, \mathbf{m}']) + \mathcal{N}_t$
            Store the new message in the memory device $\mathcal{M} \leftarrow \mathbf{m}'$
        **end for**
        Set $\mathbf{x} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_N)$ and $\boldsymbol{\Phi} = (\mathbf{m}_1, \ldots, \mathbf{m}_N)$
        Execute actions $\mathbf{a} = (a_1, \ldots, a_N)$, observe rewards $r$ and next observations $\mathbf{x}'$
        Store $(\mathbf{x}, \mathbf{x}', \mathbf{a}, \boldsymbol{\Phi}, r)$ in replay buffer $\mathcal{D}$
    **end for**
    **for** agent i = 1 to $N$ **do**
        Sample a random minibatch $\Theta$ of $B$ samples $(\mathbf{x}, \mathbf{x}', \mathbf{a}, \boldsymbol{\Phi}, r)$ from $\mathcal{D}$
        Set $y = r_i + \gamma Q^{\boldsymbol{\mu}'_{\theta_i}}(\mathbf{x}', a'_1, \ldots, a'_N)|_{a'_k=\boldsymbol{\mu}'_{\theta_k}(o_k,\mathbf{m_k})}$
        Update critic by minimizing:

$$\mathcal{L}(\theta_i) = \frac{1}{B}\sum_{(\mathbf{x},\mathbf{x}',\mathbf{a},\boldsymbol{\Phi},r)\in\Theta}(y - Q^{\boldsymbol{\mu}_{\theta_i}}(\mathbf{x}, a_1, \ldots, a_N))^2$$

        Update actor according to the policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{B}\sum_{(\mathbf{x},\mathbf{x}',\mathbf{a},\boldsymbol{\Phi},r)}\Big(\nabla_{\theta_i}\boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i,\mathbf{m}_i)\nabla_{a_i}Q^{\boldsymbol{\mu}_{\theta_i}}(\mathbf{x},a_1,\ldots,a_i,\ldots a_N)|_{a_i=\boldsymbol{\mu}_{\theta_i}(o_i,\mathbf{m}_i)}\Big)$$

    **end for**
    Update target networks:

$$\theta'_i = \tau\theta_i + (1-\tau)\theta'_i$$

**end for**

# 4 Experimental Settings and Results

## 4.1 Environments

In this section, we present a battery of six two-dimensional navigation environments (Figure 2), with continuos space and discrete time. We introduce tasks of increasing complexity, requiring progressively more elaborated coordination skills: five environments are inspired by the Cooperative Navigation problem from the multi-agent particle environment (Lowe et al., 2017; Mordatch and Abbeel, 2017) in addition to Waterworld from the SISL suite (Gupta et al., 2017). We focus on two-agents systems to keep the settings sufficiently simple and attempt an initial analysis and interpretation of emerging communication behaviours. A short description of the six environments is in order.

**Cooperative Navigation (CN).** This environment consists of $N$ agents and $N$ corresponding landmarks. An agent's task is to occupy one of the landmarks whilst avoiding collisions with other agents. Every agent observes the distance to all others agents and landmark positions.

**Partial Observable Cooperative Navigation (PO CN).** This is based on Cooperative Navigation, i.e. the task and action space are the same, but the agents now have a limited vision range and can only observe a portion of the environment around them within a pre-defined radius.

**Synchronous Cooperative Navigation (Sync CN).** The agents need to occupy the landmarks exactly at the same time in order to be positively rewarded. A landmark is declared as occupied when an agent is arbitrarily close to it. Agents are penalised when the landmarks are not occupied at the same time.

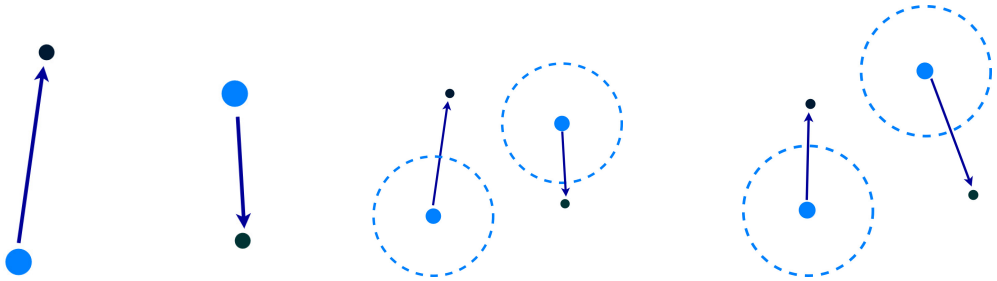
**Sequential Cooperative Navigation (Sequential CN).** This environments is similar to the previous one, but the agents here need to occupy landmarks sequentially and avoid to reach them simultaneuosly in order to be positively rewarded. Occupying the landmarks at the same time is penalised.

**Swapping Cooperative Navigation (Swapping CN).** In this case the task is more complex as it consists of two sub-tasks. Initially, the agents need to reach the landmarks and occupy them at same time. Then, they need to swap their landmarks and repeat the same process.

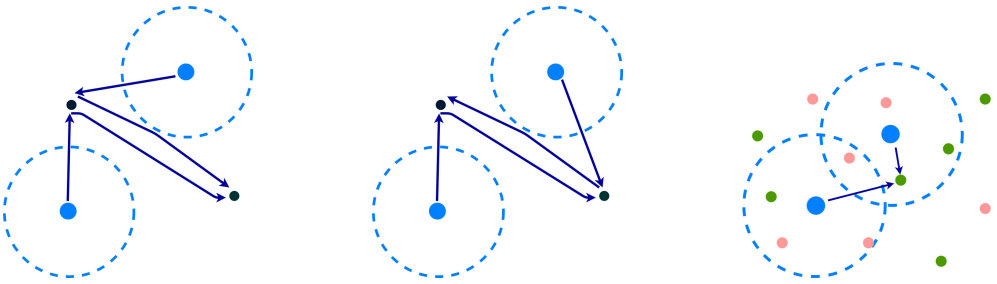
**Waterworld.** In this environment, two agents with limited range vision have to collaboratively capture food targets whilst avoiding poison targets. A food target can be captured only if both agents reach it at the same time. Additional details are reported in (Gupta et al., 2017).

## 4.2 Implementation Details

In all our experiments, we use a neural network with one layer (512 unites) for the encoding (Eq. 1), a neural network with one layer (256 units) for the action selector (Eq. 7) and neural networks with three hidden layers ($1024, 512, 256$ units, respectively) for the critics. For MADDPG and MA-MADDPG the actors are implemented with neural networks with 2 hidden layers ($512, 256$ units). The size of the $\mathbf{m}$ is fixed to 200; this value that has been empirically found to be optimal given the network architectures (Section B.7 provides a validation study on the choice of memory size). Consequently, the size of $\mathbf{h_i}$ and $\mathbf{e}_i$ is set to 200. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $10^{-3}$ for critic and $10^{-4}$ for policies. The reward discount factor is set to $0.95$, the size of the replay buffer to $10^6$ and the batch size to $1,024$. The number of time steps for episode is set to $1,000$ for Waterworld and 100 for the other environments. We update network parameters after every 100 samples added to the replay buffer using soft updates with $\tau = 0.01$. We train all the models over $60,000$ episodes of 100 time-steps each on all the environments, except for Waterworld for which we use $20,000$ episodes of $1,000$ time-steps each for training. The Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein, 1930) with $\theta = 0.15$ and $\sigma = 0.3$ is a stochastic process which, over time, tends to drift towards its mean. This is commonly employed within DDPG (Lillicrap et al., 2015) and in order to introduce temporally correlated noise. Doing so it is possible to avoid the effects of averaging random decorrelated signals which would lead a less effective exploration. Discrete actions are supported by the Gumbel-Softmax, a biased, low-variance gradient estimator (Jang et al., 2016). This estimator is typically used within the back-propagation algorithm in the presence of categorical variables. We use Python 3.5.4 (Van Rossum and Drake Jr, 1995) with PyTorch v0.3.0 (Paszke et al., 2017) as software for automatic differentiable computing and machine learning framework. All the computations were performed using Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz as CPU and GeForce GTX TITAN X as GPU.



(a) Cooperative Navigation (b) Partial Observable CN (c) Synchronous CN



(d) Sequential CN (e) Swapping CN (f) Waterworld

Figure 2: An illustration of our environments. Blue circles represent the agents; dashed lines indicate the range of vision; green and red circles represent the food and poison targets, respectively, while black dots represent landmarks to be reached.

### 4.3 Experimental results

In our experiments, we compared the proposed MD-MADDPG against four algorithms: MADDPG (Lowe et al., 2017), Meta-agent MADDPG (MA-MADDPG), CommNet (Sukhbaatar et al., 2016) and MAAC (Iqbal and Sha, 2019). MA-MADDPG is a variation of MADDPG in which the policy of an agent during both training and execution is conditioned upon the observations of all the other agents in order to overcome difficulties due to partial observability. These methods have been selected to provide fair comparisons since they offer different learning approaches in multi-agent problems. MADDPG is what our method builds on so this comparison can quantify the improvements brought by the proposed communication mechanism; MA-MADDPG offers an alternative information sharing mechanism; CommNet implements an explicit form of communication; MAAC is a recent is a state-of-the-art method in which critics select information to share through an attention mechanism. We analyse the performance of these competing learning algorithms on all the six environments described in Section 4.1. In each case, after training, we evaluate an algorithm's performance by collecting samples from an additional $1,000$ episodes, which are then used to extract different performance metrics: the *reward* quantifies how well a task has been solved; the *distance from landmarks* captures how closely an agent has reached the landmarks; the *number of collisions* counts how many times an agent has failed to avoid collisions with others; *sync occupations* counts how many times the landmarks have been occupied simultaneously and, analogously, *not sync occupations* counts how many times only one of the two landmarks has been occupied. For Waterworld, we also count the *number of food targets* and *number of poison targets*. Since this environment requires continuous actions, we cannot use MAAC as this method only operates on discrete action spaces. In Table 1, for each metric, we report the sample average and standard deviation obtained by each algorithm on each environment. A visualization of all results through boxplot can be found in Section B.5 in Supplementary Material. Illustrative videos to show the performance of MD-MADDPG on the environments are publicly available [1].

All algorithms perform very similarly in the Cooperative Navigation and Partial Observable Navigation cases. This result is expected because these environments involve relatively simple tasks that can be completed even without explicit message-passing and information sharing functionalities. Despite communication not being essential, MD-MADDPG reaches comparable performance to MADDPG and MA-MADDPG. In the Synchronous Cooperative Navigation case, the ability of MA-MADDPG to overcome partial observability issues by sharing the observations across agents seem to be crucial as the total rewards achieved by this algorithm are substantially higher than those obtained by both

[1] Supplementary illustrative videos: `https://www.youtube.com/watch?v=P9XWdpmsEy8`

| Environment | Metric | MADDPG | MA-MADDPG | CommNet | MAAC | MD-MADDPG |
|---|---|---|---|---|---|---|
| CN | Reward | $-2.30 \pm 0.11$ | $-2.29 \pm 0.10$ | $-2.7 \pm 0.26$ | $-4.72 \pm 1.35$ | $\mathbf{-2.27 \pm 0.10}$ |
| | Average distance | $0.15 \pm 0.051$ | $0.14 \pm 0.05$ | $-0.35 \pm 0.13$ | $1.36 \pm 0.67$ | $\mathbf{0.13 \pm 0.05}$ |
| | # collisions | $\mathbf{0.11 \pm 0.76}$ | $0.17 \pm 0.90$ | $0.19 \pm 1.06$ | $0.58 \pm 1.42$ | $0.12 \pm 0.82$ |
| PO CN | Reward | $\mathbf{-2.62 \pm 0.34}$ | $-2.67 \pm 0.38$ | $-2.78 \pm 0.43$ | $-3.17 \pm 0.62$ | $-2.68 \pm 0.46$ |
| | Average distance | $\mathbf{0.30 \pm 0.17}$ | $0.33 \pm 0.19$ | $0.39 \pm 0.21$ | $1.26 \pm 2.53$ | $0.34 \pm 0.22$ |
| | # collisions | $0.55 \pm 1.64$ | $\mathbf{0.14 \pm 0.69}$ | $0.37 \pm 1.48$ | $0.58 \pm 0.31$ | $0.26 \pm 1.06$ |
| Sync CN | Reward | $75.83 \pm 72.23$ | $\mathbf{192.92 \pm 29.78}$ | $188.02 \pm 35.41$ | $161.35 \pm 80.03$ | $92.90 \pm 69.78$ |
| | # sync occup. | $26.71 \pm 19.86$ | $\mathbf{53.96 \pm 20.16}$ | $3.62 \pm 12.14$ | $139.56 \pm 63.55$ | $31.6 \pm 19.34$ |
| | # not sync occup. | $21.36 \pm 16.60$ | $41.75 \pm 56.25$ | $46.35 \pm 29.47$ | $44.84 \pm 58.71$ | $\mathbf{17.58 \pm 12.00}$ |
| Sequential CN | Reward | $125.98 \pm 33.4$ | $117.52 \pm 35.62$ | $131.67 \pm 19.48$ | $90.11 \pm 21.33$ | $\mathbf{130.15 \pm 35.19}$ |
| | Average distance | $260.16 \pm 14.41$ | $114.7 \pm 45.71$ | $102.63 \pm 34.96$ | $101.11 \pm 40.71$ | $\mathbf{99.15 \pm 50.59}$ |
| Swapping CN | Reward | $125.60 \pm 50.13$ | $86.99 \pm 68.52$ | $109.55 \pm 56.64$ | $75.71 \pm 69.80$ | $\mathbf{129.63 \pm 47.26}$ |
| | Average distance | $76.70 \pm 30.24$ | $132.77 \pm 89.98$ | $123.54 \pm 84.9$ | $152.7 \pm 43.72$ | $\mathbf{53.21 \pm 40.80}$ |
| Waterworld | Reward | $262.29 \pm 141.07$ | $99.31 \pm 118.31$ | $139.29 \pm 121.42$ | $NA$ | $\mathbf{503.96} \pm 103.91$ |
| | # food targets | $13.91 \pm 7.30$ | $5.25 \pm 6.07$ | $10.2 \pm 7.1$ | $NA$ | $\mathbf{26.25 \pm 5.41}$ |
| | # poison targets | $8.61 \pm 3.32$ | $\mathbf{5.34 \pm 2.45}$ | $8.01 \pm 5.22$ | $NA$ | $7.77 \pm 2.95$ |

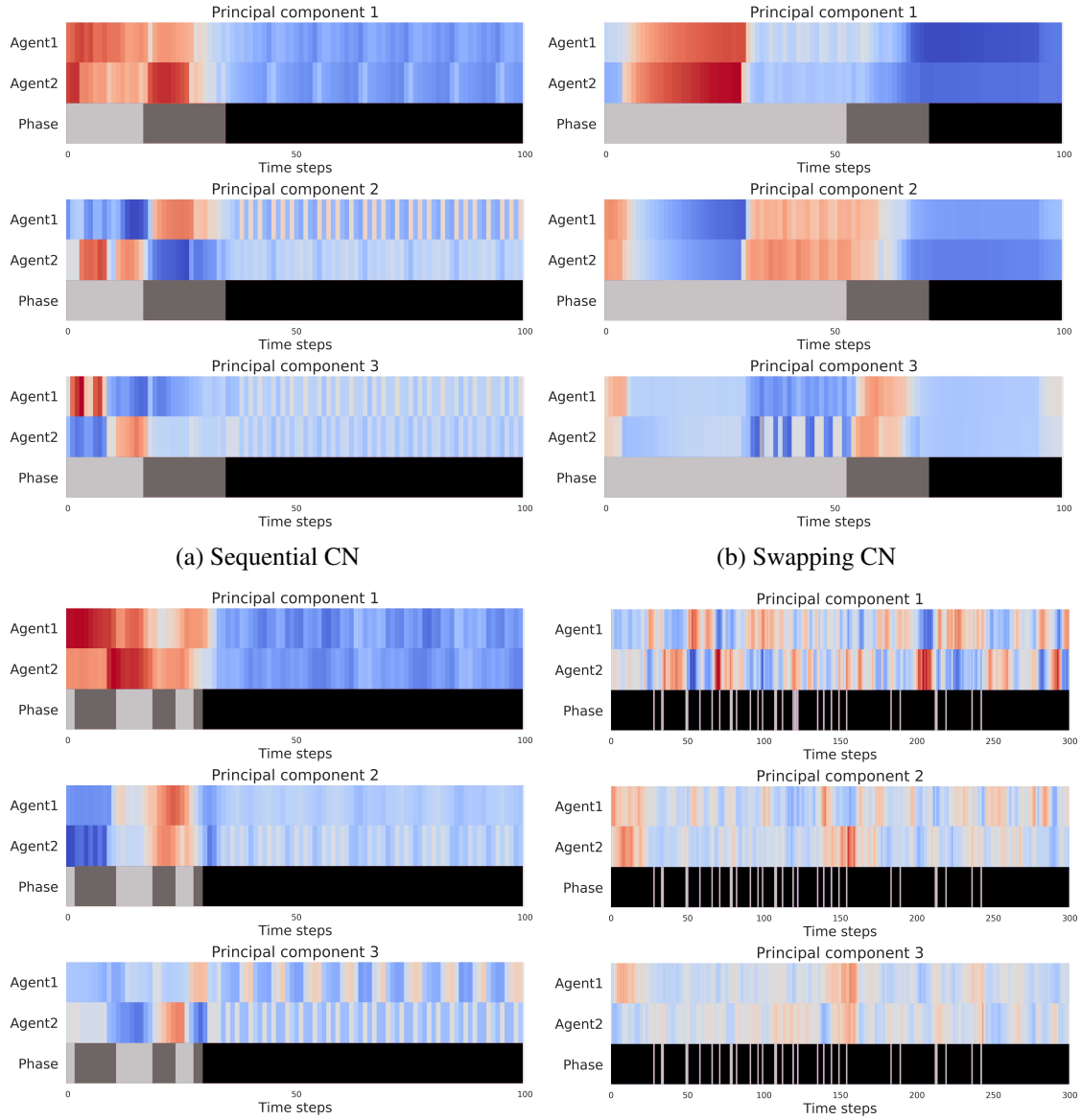
Table 1: Comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on six environments ordered by increasing level of difficulty, from CN to Waterword. The sample mean and standard deviation for $1,000$ episodes are reported for each metric.

MADDPG and MD-MADDPG. In this case, whilst not achieving the highest reward, MD-MADDPG keeps the number of unsynchronised occupations at the lowest level, and also performs better than MADDPG on all three metrics. It would appear that in this case pulling all the private observations together is sufficient for the agents to synchronize their paths leading to the landmarks.

When moving on to more complex tasks requiring further coordination, the performances of the three algorithms diverge further in favour of MD-MADDPG. The requirement for strong collaborative behaviour is more evident in the Sequential Cooperative Navigation problem as the agents need to explicitly learn to take either shorter or longer paths from their initial positions to the landmarks in order to occupy them in sequential order. Furthermore, according to the results in Table 1, the average distance travelled by the agents trained with MD-MADDPG is less then the half the distance travelled by agents trained with MADDPG, indicating that these agents were able to find a better strategy by developing an appropriate communication protocol. Similarly, in the Swapping Cooperative Navigation scenario, MD-MADDPG achieves superior performance, and is again able to discover solutions involving the shortest f. Waterworld is significantly more challenging as it requires a sustained level of synchronization throughout the entire episode and can be seen as a sequence of sub-tasks whereby each time the agents must reach a new food target whilst avoiding poison targets. In Table 1, it can be noticed that MD-MADDPG significantly outperforms both competitors in this case. The importance of sharing observations with other agents can also be seen here as MA-MADDPG generates good policies that avoid poison targets, yet in this case, the average reward is substantially lower than the one scored by MD-MADDPG. Experimental settings so far have involved two agents. In addition, we have also investigated settings with a higher number of agents, see Supplementary Material (Section B.2 for Cooperative Navigation and Section B.3 for Partial Observable Cooperative Navigation). These results show, that the prososed method can be successfully used on larger systems without incurring any numerical complications or convergence difficulties. When comparing to other algorithms, MD-MADDPG has resulted in superior performance performance are indeed achieved on Cooperative Navigation with respect to the reward metric. On Partially Observable Cooperative Navigation, there is no definite winner, nevertheless MD-MADDPG shows competitive performance, for example it outperforms all the baselines on the 5 agents scenario.

In Supplementary Material in Section B.4, we provide an ablation study showing that the main components of MD-MADDPG are needed for its correct behaviour. We investigate the effects of removing either one of the key components, i.e. context vector, read and write modules. Removing the context vector reduces the quality of the performance obtained on CN and on environments which require greater coordination efforts, like Sequential CN, Swapping CN and Waterworld. On PO-CN no significant differences in performance are reported, while on Synchronous CN sync occupations worsen (of approximatively five times the amount) and sync occupations improve (of approximatively twice the amount). This result is explained by the fact that in Sync CN, good strategies that do not involve explicit communication can be learnt to achieve good performance on sync occupations. The best overall performance method on this scenario is MA-MADDPG (see Table 1). This comparative method implements an implicit form of communication that is equivalent to a simple information sharing which can be very effective to overcome the partial observability

issue which is the main challenge in Sync CN. We have observed that without the writing or reading components the performance worsened on all the run experiments.




(a) Sequential CN

(b) Swapping CN

(c) Synchronous CN

(d) Waterworld

Figure 3: Visualisation of communications strategies learned by the agents in four different environments: the three principal components provide orthogonal descriptors of the memory content written by the agents and are being plotted as a function of time. Within each component, the highest values are in red, and the lowest values are in blue. The bar at the bottom of each figure indicates which phase (or sub-task) was being executed within an episode; see Section 4.4 for further details. The memory usage patterns learned by the agents are correlated with the underlying phases and the memory is no longer utilised once a task is about to be completed.

### 4.4 Communication analysis

In this section, we explore the dynamic patterns of communication activity that emerged in the environments presented in the previous section, and look at how the agents use the shared memory throughout an episode while solving the required task. For each environment, after training, we executed episodes with time horizon $T$ and stored the write vector $\mathbf{m}'$ of each agent at every time step $t$. Exploring how $\mathbf{m}'$ evolves within an episode can shed some light onto

the role of the memory device at each phase of the task. The analysis presented in this section focuses on the write vector as we expect it to be stronger correlated with the environment dynamics than the other components. The content of the writing vector corresponds to the content of the communication channel itself, and is expected to contain information related to the task (e.g. changes in current environment, agent's strategy or observed point of interests). A communication analysis with respect to the read vector $\mathbf{r}_i$ is presented in Supplementary Material (Section B.8). The content of the reading vector is an implicit representation internal to the agent itself that serves to interpret the content of the channel and at the same time to be utilised in the generation of $\mathbf{m}'$. In order to produce meaningful visualisations, we first projected the dimensions of $\mathbf{m}'$ onto the directions maximising the sample variance (i.e. the variance of the observed $\mathbf{m}'$ across simulated episodes) using a linear PCA.

Figure 3 shows the principal components (PCs) associated with the two agents over time for four of our six simulation environments. Only the first three PCs were retained as those were found to cumulatively explain over $80\%$ of variance in all cases. The values of each PC were standardised to lie in $[0, 1]$ in order have them in the same in same range for fair comparisons and are plotted on a color map: one is in red and zero in blue. The timeline at the bottom of each figure indicates which specific phase of an episode is being executed at any given time point, and each consecutive phase is coloured using a different shade of grey. For instance, in Sequential Cooperative Navigation, a single landmark is reached and occupied in each phase. In Swapping Cooperative Navigation, during the first phase the agents search and find the landmarks; in the second phase they swap targets, and in the third phase they complete the task by reaching the landmarks again. In the Synchronous Cooperative Navigation the phase indicates if none of the landmarks is occupied (light-grey), if just one is occupied (dark-grey) and if both are occupied (black). Usually, in the last phase, the agents learn to stay close to their targets. This analysis pointed out that in the final phases, when tasks are already completed and there is no need of coordination, the PCs representing the communication activities assume lower (blue values), while during previous phases, when tasks are still to be solved and cooperation is stronger required, they assume higher values (red). This led us to interpret the higher values as being indicative of high memory usage, and lower values as being associated to low activity. In most cases, high communication activity is maintained when the agents are actively working and completing a task, while during the final phases (where typically there is no exploration because the task is considered completed) low activity levels are more predominant.

This analysis also highlights the fact that the communication channel is used differently in each environment. In some cases, the levels of activity alternate between agents. For instance, in Sequential Cooperative Navigation (Figure 3a), high levels of memory usage by one agent are associated with low ones by the other. A different behaviour is observed for the other environments, indeed in Swapping Cooperative Navigation task where both agents produce either high or low activation value, whereas in Synchronous Cooperative Navigation the memory activity is very intense before the phase three, while agents are collaborating to complete the task. The dynamics characterizing the memory usage also change based on the particular phase reached within an episode. For example, in Figure 3a, during the first two phases the agents typically show alternating activity levels whilst in the third phase both agents significantly decrease their memory activity as the task has already been solved and there are no more changes in the environment. Figure 3 provides some evidence that, in some cases, a peer-to-peer communication strategy is likely to emerge instead of a master-slave one where one agent takes complete control of the shared channel. The scenario is significantly more complex in Waterworld where the changes in memory usage appear at a much higher frequency due to the presence of very many sequential sub-tasks. Here, each light-grey phase indicates that a food target has been captured. Peaks of memory activity seem to follow those events as the agents reassess their situation and require higher coordination to jointly decide what the next target is going to be. In Supplementary Material (B.1) we provide further experimental results showing the importance of the communication by corrupting the memory content at execution time, which further corroborate the role of the exchanged messages in improving agents' coordination.

## 5 Conclusions

In this work, we have introduced MD-MADDPG, a multi-agent reinforcement learning framework that uses a shared memory device as an intra-agent communication channel to improve coordination skills. The memory content contains a learned representation of the environment that is used to better inform the individual policies. The memory device is learnable end-to-end without particular constraints other than its size, and each agent develops the ability to modify and interpret it. We empirically demonstrated that this approach leads to better performance in small-scale (up to 6 agents in our experiments) cooperative tasks where coordination and synchronization are crucial to a successful completion of the task and where world visibility is very limited. Furthermore, we have visualised and analyzed the dynamics of the communication patterns that have emerged in several environments. This exploration has indicated that, as expected, the agents have learned different communication protocols depending upon the complexity of the task. In this study we have mostly focused on two-agent systems to keep the settings sufficiently simple to understand the role of the memory. Very competitive results have been obtained when more agents are used.

In future work, we plan on studying the role played by the sequential order in which the memory is updated, as the number of agents grows. A possible approach may consist of deploying agent selection mechanisms, possibly based on attention, so that only a relevant subset of agents can modify the memory at any given time, or impose master-slave architectures. A possible solution would be to have an agent acting as "scheduler"that controls the access to the memory, decides which information can be shared and provides scheduling for the writing accesses. Introducing such a scheduling agent would allow to keep the current framework unaltered, e.g. the sequential access to the memory would be retained. Although the scheduling agent would add an additional layer of complexity, this might reduce the number of memory access required in larger scale systems and improve the overall scalability. In future work, we will also apply MD-MADDPG on environments characterized by more structured and high-dimensional observations (e.g. pixel data) where collectively learning to represent the environment through a shared memory should be particularly beneficial.

## References


Sanjeevan Ahilan and Peter Dayan. Feudal multi-agent hierarchies for cooperative reinforcement learning. *arXiv preprint arXiv:1901.08492*, 2019.

Jeannette Brosig, Axel Ockenfels, Joachim Weimann, et al. *Information and communication in sequential bargaining*. Citeseer, 2003.

Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015.

Mark G Chen. Communication, coordination, and camaraderie in world of warcraft. *Games and Culture*, 4(1):47–73, 2009.

Xiangxiang Chu and Hangjun Ye. Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1710.00336*, 2017.

Louise K Comfort. Crisis management in hindsight: Cognition, communication, coordination, and control. *Public Administration Review*, 67:189–197, 2007.

Russell Cooper, Douglas V DeJong, Robert Forsythe, and Thomas W Ross. Communication in the battle of the sexes game: some experimental results. *The RAND Journal of Economics*, 20(4):568, 1989.

Russell Cooper, Douglas V De Jong, Robert Forsythe, and Thomas W Ross. Forward induction in coordination games. *Economics Letters*, 40(2):167–172, 1992.

Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

Robert H Crites and Andrew G Barto. Elevator group control using multiple reinforcement learning agents. *Machine learning*, 33(2-3):235–262, 1998.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. *arXiv preprint arXiv:1810.11187*, 2018.

Jan Peter De Ruiter, Matthijs L Noordzij, Sarah Newman-Norlund, Roger Newman-Norlund, Peter Hagoort, Stephen C Levinson, and Ivan Toni. Exploring the cognitive infrastructure of communication. *Interaction Studies*, 11(1):51–77, 2010.

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

Stefano Demichelis and Jorgen W Weibull. Language, meaning, and games: A model of communication, coordination, and evolution. *American Economic Review*, 98(4):1292–1311, 2008.

Richard Evans and Jim Gao. Deepmind ai reduces google data centre cooling bill by 40 https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40s, 2016. Accessed: 17-09-2018.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

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

Jakob N Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1811.01458*, 2018.

Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous robots*, 8(3):325–344, 2000.

Aaron French, Marcelo Macedo, John Poulsen, Tyler Waterson, and Angela Yu. Multivariate analysis of variance (manova). *San Francisco State University*, 2008.

Riccardo Fusaroli, Bahador Bahrami, Karsten Olsen, Andreas Roepstorff, Geraint Rees, Chris Frith, and Kristian Tylén. Coming to terms: quantifying the benefits of linguistic coordination. *Psychological science*, 23(8):931–939, 2012.

Bruno Galantucci. An experimental study of the emergence of human communication systems. *Cognitive science*, 29 (5):737–767, 2005.

Simon Garrod, Nicolas Fay, Shane Rogers, Bradley Walker, and Nik Swoboda. Can iterated learning explain the emergence of graphical symbols? *Interaction Studies*, 11(1):33–50, 2010.

Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, volume 2, pages 227–234. Citeseer, 2002.

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

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. *ICML*, 2019.

Takayuki Itō, Minjie Zhang, Valentin Robu, Shaheen Fatima, Tokuro Matsuo, and Hirofumi Yamaki. *Innovations in Agent-Based Complex Automated Negotiations*. Springer-Verlag Berlin Heidelberg, 2011.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Nathanaël Jarrassé, Themistoklis Charalambous, and Etienne Burdet. A framework to describe, analyze and generate interactive motor behaviors. *PloS one*, 7(11):e49945, 2012.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *arXiv preprint arXiv:1805.07733*, 2018.

Michael Kearns. Experiments in social computation. *Communications of the ACM*, 55(10):56–67, 2012.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. *ICLR*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Xiangyu Kong, Bo Xin, Fangchen Liu, and Yizhou Wang. Revisiting the master-slave architecture in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1712.07305*, 2017.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Harold D Lasswell. The structure and function of communication in society. *The communication of ideas*, 37(1): 136–39, 1948.

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

Guillaume J Laurent, Laëtitia Matignon, Le Fort-Piat, et al. The world of independent learners is not markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(1):55–64, 2011.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

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

Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

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

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

Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat. 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.*, pages 64–69, 2007.

John H Miller and Scott Moser. Communication and coordination. *Complexity*, 9(5):31–40, 2004.

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

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

Frans A Oliehoek and Nikos Vlassis. Q-value functions for decentralized pomdps. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 220. ACM, 2007.

Norihiko Ono and Kenji Fukumoto. Multi-agent reinforcement learning: A modular approach. In *Second International Conference on Multiagent Systems*, pages 252–258, 1996.

Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

Dawn C Parker, Steven M Manson, Marco A Janssen, Matthew J Hoffmann, and Peter Deadman. Multi-agent systems for the simulation of land-use and land-cover change: a review. *Annals of the association of American Geographers*, 93(2):314–337, 2003.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017.

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

Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 489–496. Morgan Kaufmann Publishers Inc., 2000.

Alberto Petrillo, Alessandro Salvi, Stefania Santini, and Antonio Saverio Valente. Adaptive multi-agents synchronization for collaborative driving of autonomous vehicles with multiple communication delays. *Transportation research part C: emerging technologies*, 86:372–392, 2018.

Manisa Pipattanasomporn, Hassan Feroze, and Saifur Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pages 1–8. IEEE, 2009.

Wei Ren and Nathan Sorensen. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4):324–333, 2008.

Luca Scardovi and Rodolphe Sepulchre. Synchronization in networks of identical linear systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 546–551. IEEE, 2008.

Jurgen Schmidhuber. A general method for multi-agent reinforcement learning in unrestricted environments. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 84–87, 1996.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

Reinhard Selten and Massimo Warglien. The emergence of simple languages in an experimental coordination game. *Proceedings of the National Academy of Sciences*, 104(18):7361–7366, 2007.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *ICLR*, 2019.

Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier, 1994.

Peter Stone and Manuela Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, 48(1):83–104, 1998.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Szabolcs Számadó. Pre-hunt communication provides context for the evolution of early human language. *Biological Theory*, 5(4):366–382, 2010.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

Carrie Ann Theisen, Jon Oberlander, and Simon Kirby. Systematicity and arbitrariness in novel communication systems. *Interaction Studies*, 11(1):14–32, 2010.

Karl Tuyls and Gerhard Weiss. Multiagent learning: Basics, challenges, and prospects. *Ai Magazine*, 33(3):41, 2012.

George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

Yevgeniy Vorobeychik, Zlatko Joveski, and Sixie Yu. Does communication help people coordinate? *PloS one*, 12(2): e0170780, 2017.

Guanghui Wen, Zhisheng Duan, Wenwu Yu, and Guanrong Chen. Consensus in multi-agent systems with communication constraints. *International Journal of Robust and Nonlinear Control*, 22(2):170–182, 2012.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207*, 2019.

Tim Wharton. Natural pragmatics and natural codes. *Mind & language*, 18(5):447–477, 2003.

Michael Wunder, Michael Littman, and Matthew Stone. Communication, credibility and negotiation using a cognitive hierarchy model. In *Workshop# 19: MSDM 2009*, page 73, 2009.

Keyou You and Lihua Xie. Network topology and communication data rate for consensusability of discrete-time multi-agent systems. *IEEE Transactions on Automatic Control*, 56(10):2262, 2011.

# Supplementary Material

## A Meta-agent

In the meta-agent MADDPG (MA-MADDPG) each agent can observe the observations of all other agents to address issues related to partial observability. The policy of each agent is defined as $\boldsymbol{\mu}_i' = \mathcal{O}_1, \times \mathcal{O}_2 \times \cdots \times \mathcal{O}_N$ and the gradient is:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_{\theta_i}) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}} \Big[ \nabla_{\theta_i} \boldsymbol{\mu}_{\theta_i}(\mathbf{x}) \nabla_{a_i} Q^{\boldsymbol{\mu}_{\theta_i}}(\mathbf{x}, a_1, \ldots, a_N)|_{a_i = \boldsymbol{\mu}_{\theta_i}(\mathbf{x})} \Big].$$

where $\mathbf{x} = \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_N$ and $Q^{\boldsymbol{\mu}_{\theta_i}}$ is updated according to 10.

## B Additional Experiments

### B.1 Corrupting the memory

Table B1 shows the performance of MD-MADDPG when a Gaussian noise (mean 0 and standard deviation 1) is added to the memory content $\mathbf{m}$ at execution time. It can be noted that the corruption of the communication channel causes a general worsening of the performance across metrics. This shows that the messages exchanged by the agents are crucial to achieving good performance, and that corrupting the memory hinders the communication which has a negative effect on synchronization.

| **Environment** | **Metric** | **MD-MADDPG - noise** |
|---|---|---|
| CN | Reward | $-2.28 \pm 0.1$ |
| | Average distance | $0.15 \pm 0.051$ |
| | # collisions | $0.11 \pm 0.76$ |
| PO-CN | Reward | $-2.68 \pm 0.45$ |
| | Average distance | $0.34 \pm 0.22$ |
| | # collisions | $0.32 \pm 1.14$ |
| Sync CN | Reward | $187.17 \pm 41.84$ |
| | # sync occup. | $33.27 \pm 39.07$ |
| | # not sync occup. | $102.61 \pm 41.43$ |
| Sequential CN | Reward | $124.72 \pm 30.28$ |
| | Average distance | $111.27 \pm 52.66$ |
| Swapping CN | Reward | $124.93 \pm 44.48$ |
| | Average distance | $112.44 \pm 83.88$ |
| Waterworld | Reward | $24.07 \pm 26.61$ |
| | # food targets | $1.65 \pm 1.33$ |
| | # poison targets | $10.37 \pm 3.91$ |

Table B1: Performance of MD-MADDPG when Gaussian noise is added to the memory content at test time.

### B.2 Increasing the number of agents - Cooperative Navigation

Table B2 shows the comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on Cooperative Navigation when the number of agents increases. MD-MADDPG have the best performance achieving the highest reward on all the scenarios. MAAC shows higher performance in collision avoidance and CommNet in distance travelled (five and six agents).

| # agents | Metric | MADDPG | MA-MADDPG | CommNet | MAAC | MD-MADDPG |
|---|---|---|---|---|---|---|
| 3 | Reward | $-4.02 \pm 0.32$ | $-4.03 \pm 0.29$ | $-4.66 \pm 0.35$ | $-7.38 \pm 1.28$ | $\mathbf{-3.75 \pm 0.21}$ |
| | Average distance | $0.34 \pm 0.1$ | $0.34 \pm 0.09$ | $0.53 \pm 0.11$ | $1.45 \pm 0.43$ | $\mathbf{0.25 \pm 0.07}$ |
| | # collisions | $1.24 \pm 3.76$ | $1.18 \pm 2.2$ | $5.94 \pm 11.0$ | $2.95 \pm 5.04$ | $\mathbf{1.15 \pm 2.34}$ |
| 4 | Reward | $-6.86 \pm 0.68$ | $-7.0 \pm 0.77$ | $-7.47 \pm 0.64$ | $-12.82 \pm 1.87$ | $\mathbf{-6.12 \pm 0.52}$ |
| | Average distance | $0.7 \pm 0.17$ | $0.73 \pm 0.19$ | $0.81 \pm 0.18$ | $2.19 \pm 0.47$ | $\mathbf{0.51 \pm 0.12}$ |
| | # collisions | $7.44 \pm 14.82$ | $9.47 \pm 19.14$ | $23.05 \pm 31.04$ | $\mathbf{4.43 \pm 6.01}$ | $7.3 \pm 17.53$ |
| 5 | Reward | $-11.46 \pm 1.35$ | $-11.94 \pm 1.38$ | $-11.52 \pm 1.1$ | $-16.92 \pm 3.41$ | $\mathbf{-11.44 \pm 1.48}$ |
| | Average distance | $1.26 \pm 0.27$ | $1.35 \pm 0.29$ | $\mathbf{1.21 \pm 0.22}$ | $2.37 \pm 0.68$ | $1.26 \pm 0.29$ |
| | # collisions | $13.88 \pm 19.94$ | $16.94 \pm 28.21$ | $42.52 \pm 36.86$ | $\mathbf{5.24 \pm 6.53}$ | $13.73 \pm 22.98$ |
| 6 | Reward | $-18.23 \pm 2.48$ | $-19.07 \pm 2.06$ | $-18.21 \pm 2.24$ | $-29.11 \pm 5.46$ | $\mathbf{-18.08 \pm 2.32}$ |
| | Average distance | $2.0 \pm 0.41$ | $2.13 \pm 0.35$ | $\mathbf{1.93 \pm 0.37}$ | $3.83 \pm 0.91$ | $1.96 \pm 0.38$ |
| | # collisions | $23.43 \pm 26.02$ | $30.72 \pm 33.47$ | $59.9 \pm 30.03$ | $\mathbf{11.04 \pm 10.11}$ | $31.06 \pm 35.49$ |

Table B2: Comparison of MADDPG, MA-MADDPG, CommNet, MAAC, and MD-MADDPG on Cooperative Navigation when increasing the number of agents.

### B.3 Increasing the number of agents - Partial Observable Cooperative Navigation

Table B3 presents the comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on Partially Observable Cooperative Navigation when the number of agents increases. It can be noted that MD-MADDPG still achieves good performance and in some scenarios (e.g. number of agents = 5) it outperforms other methods.

| # of agents | Metric | MADDPG | MA-MADDPG | CommNet | MAAC | MD-MADDPG |
|---|---|---|---|---|---|---|
| 3 | Reward | $\mathbf{-4.66 \pm 0.76}$ | $-4.96 \pm 0.95$ | $-5.15 \pm 0.86$ | $-6.73 \pm 1.44$ | $-4.97 \pm 0.94$ |
| | Average distance | $\mathbf{0.54 \pm 0.25}$ | $0.65 \pm 0.31$ | $0.7 \pm 0.29$ | $1.21 \pm 0.47$ | $0.65 \pm 0.31$ |
| | # collisions | $2.35 \pm 4.76$ | $1.54 \pm 4.5$ | $4.18 \pm 7.76$ | $9.61 \pm 13.28$ | $\mathbf{2.35 \pm 4.61}$ |
| 4 | Reward | $-8.11 \pm 1.37$ | $\mathbf{-7.56 \pm 1.17}$ | $-9.05 \pm 1.49$ | $-10.79 \pm 2.07$ | $-8.17 \pm 1.44$ |
| | Average distance | $1.02 \pm 0.34$ | $\mathbf{0.88 \pm 0.29}$ | $1.19 \pm 0.36$ | $1.68 \pm 0.51$ | $1.04 \pm 0.36$ |
| | # collisions | $4.97 \pm 7.56$ | $\mathbf{2.82 \pm 5.47}$ | $29.5 \pm 28.71$ | $7.03 \pm 7.18$ | $2.89 \pm 5.24$ |
| 5 | Reward | $-16.33 \pm 3.06$ | $-16.56 \pm 2.53$ | $-15.79 \pm 2.61$ | $-16.59 \pm 3.11$ | $\mathbf{-15.29 \pm 3.07}$ |
| | Average distance | $0.66 \pm 0.18$ | $1.68 \pm 0.5$ | $1.48 \pm 0.51$ | $6.13 \pm 7.44$ | $\mathbf{0.61 \pm 0.18}$ |
| | # collisions | $7.86 \pm 8.38$ | $15.64 \pm 17.38$ | $67.01 \pm 44.81$ | $\mathbf{2.31 \pm 0.62}$ | $3.38 \pm 4.2$ |
| 6 | Reward | $-18.69 \pm 3.18$ | $-20.24 \pm 2.62$ | $\mathbf{-17.11 \pm 2.86}$ | $-21.3 \pm 4.61$ | $-39.83 \pm 2.29$ |
| | Average distance | $2.09 \pm 0.53$ | $2.31 \pm 0.43$ | $\mathbf{1.72 \pm 0.47}$ | $2.53 \pm 0.76$ | $5.63 \pm 0.38$ |
| | # collisions | $13.14 \pm 13.21$ | $35.38 \pm 19.84$ | $76.64 \pm 48.27$ | $8.99 \pm 8.63$ | $\mathbf{6.47 \pm 6.1}$ |

Table B3: Comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on Partial Observable Cooperative Navigation when increasing the number of agents.

### B.4 Ablation study

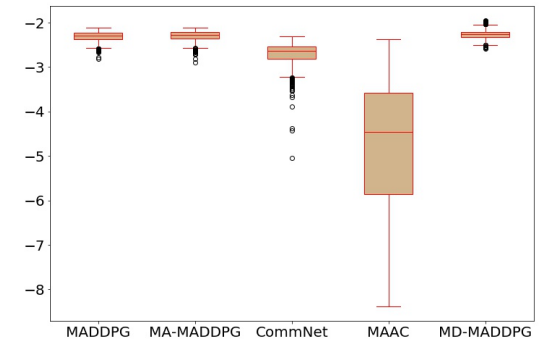
In the experiments presented here we study the benefits of each specific component, such as the context vector and the modules required for communicating, on final performance. To assess the importance of the context vector (Eq. 3.2) we have run a set of experiments removing $\mathbf{h}_i$ from the reading module of the agents. Table B4 shows that by using only $\mathbf{e}_i$ without $\mathbf{h}_i$ during the reading phase, the performance overall degrades on almost all the environments. We have noticed that the role played by the context vector becomes more critical as the level of communication requires by the underlying task increases, like in Sequential CN, Synchronous CN and Waterworld. We also run experiments to assess the performance of the components involved in the functioning of communication. It resulted that removing either the reading or the writing modules the performance significantly worsened on every scenario.

| **Environment** | **Metric** | **no context** | **no read** | **no write** | **MD-MADDPG** |
|---|---|---|---|---|---|
| CN | Reward | $-2.28 \pm 0.1$ | $-35.13 \pm 2.07$ | $-9.04 \pm 5.37$ | $\mathbf{-2.27 \pm 0.10}$ |
| | Average distance | $0.14 \pm 0.05$ | $16.57 \pm 1.04$ | $3.51 \pm 2.68$ | $\mathbf{0.13 \pm 0.05}$ |
| | # collisions | $\mathbf{0.08 \pm 0.59}$ | $0.2 \pm 0.96$ | $1.52 \pm 3.32$ | $0.12 \pm 0.82$ |
| PO-CN | Reward | $-2.68 \pm 0.45$ | $-5.4 \pm 0.75$ | $-5.93 \pm 0.07$ | $-2.68 \pm 0.46$ |
| | Average distance | $0.34 \pm 0.22$ | $1.7 \pm 0.38$ | $1.96 \pm 0.03$ | $0.34 \pm 0.22$ |
| | # collisions | $0.32 \pm 1.14$ | $0.56 \pm 1.61$ | $0.58 \pm 1.56$ | $\mathbf{0.26 \pm 1.06}$ |
| Sync CN | Reward | $\mathbf{189.8 \pm 38.39}$ | $-16.5 \pm 2.69$ | $-16 \pm 0.1$ | $92.90 \pm 69.78$ |
| | # sync occup. | $\mathbf{51.61 \pm 58.65}$ | $0.1 \pm 0.72$ | $0.2 \pm 0.92$ | $31.6 \pm 19.34$ |
| | # not sync occup. | $103.36 \pm 61.23$ | $21.22 \pm 59.75$ | $2.68 \pm 2.9$ | $\mathbf{17.58 \pm 12.00}$ |
| Sequential CN | Reward | $113.33 \pm 48.11$ | $-13.59 \pm 0.77$ | $-13.85 \pm 0.12$ | $\mathbf{130.15 \pm 35.19}$ |
| | Average distance | $129.79 \pm 83.56$ | $377.73 \pm 35.6$ | $391.75 \pm 5.87$ | $\mathbf{99.15 \pm 50.59}$ |
| Swapping CN | Reward | $75.76 \pm 66.49$ | $-15.53 \pm 0.37$ | $-12.54 \pm 2.52$ | $\mathbf{129.63 \pm 47.26}$ |
| | Average distance | $158.39 \pm 108.42$ | $579.23 \pm 10.02$ | $416.8 \pm 86.15$ | $\mathbf{53.21 \pm 40.80}$ |
| Waterworld | Reward | $31.31 \pm 77.31$ | $-1.5 \pm 2.33$ | $21.95$ | $\mathbf{503.96} \pm 103.91$ |
| | # food targets | $1.8 \pm 4.04$ | $0.21 \pm 0.11$ | $1.18 \pm 1.12$ | $\mathbf{26.25 \pm 5.41}$ |
| | # poison targets | $\mathbf{5.22 \pm 2.95}$ | $3.4 \pm 2.22$ | $3.98 \pm 2.28$ | $7.77 \pm 2.95$ |

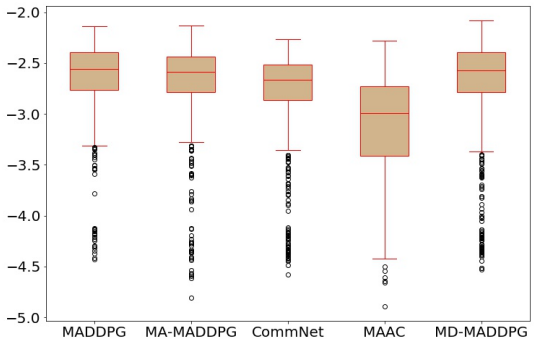
Table B4: An assessment of MD-MADDPG without context vector, MD-MADDPG without reading operation, MD-MADDPG without writing operation compared with the standard version MD-MADDPG.

### B.5 Main results visualization

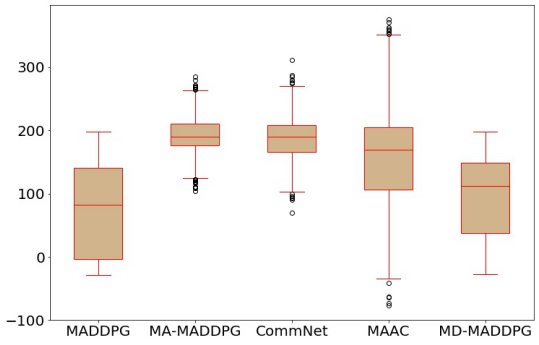
Figure B1 provides boxplot visualization of the main results already presented in Table 1.



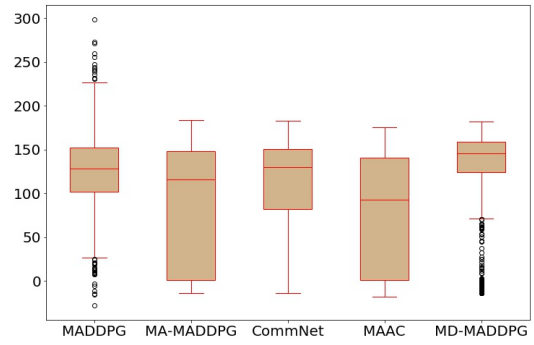

(a) Cooperative Navigation



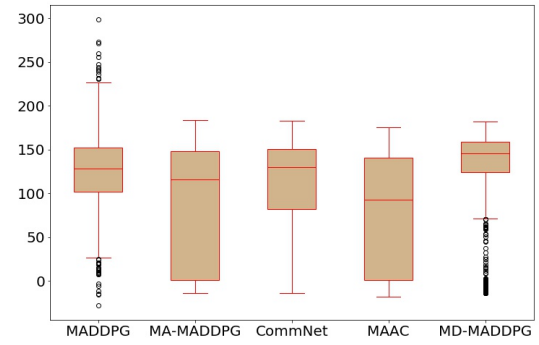

(b) Partial Observable CN



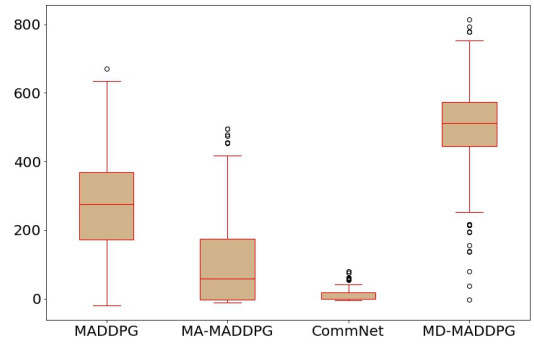

(c) Synchronous CN




(d) Sequential CN




(e) Swapping CN




(f) Waterworld

Figure B1: Boxplot representing the main results in Table 1.

### B.6 Multiple seeds

In this section, we investigate the sensitivity of MD-MADDPG on changes in random seeds used for setting the initial conditions of the randomness in the learning process. To show that the presented results are not affected by a particular choice of the seed that can significantly condition the final performance, we report the outcome of varying different seeds. Figures B2 and B3 show respectively the results of MD-MADDPG on Swapping CN and Sequential CN when changing the seed for training and testing the model. It can be noted that in both cases, models are not seed-sensitives, indeed varying the seed does not significantly affect the final results. In order to investigate the statistical significance of these results, we carried out a MANOVA (Multivariate ANOVA) (French et al., 2008), assessing the null hypothesis that all the population means are the same. In both scenarios, there is no enough evidence to conclude that there is a difference in means at the $0.001$ significance level ($p$-values is $0.1267$ on Swapping CN and $0,8357$ on Sequential CN).

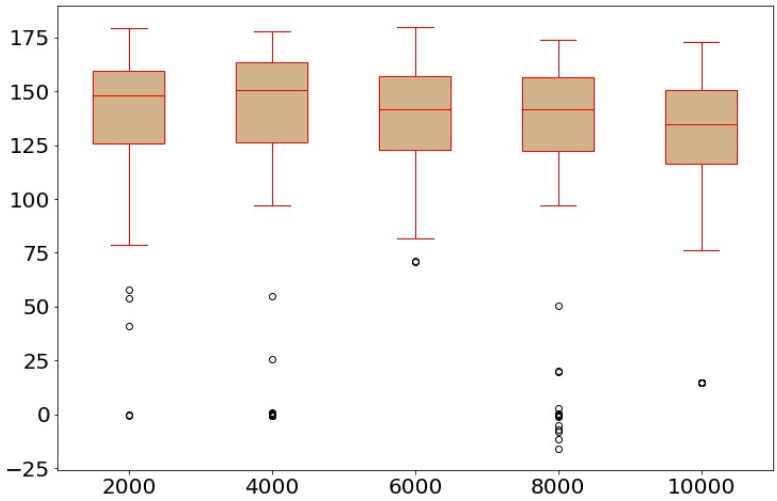



Figure B2: Boxplot representing the results of MD-MADDPG on Swapping CN when changing different seeds. The horizontal axis shows the seed and the vertical axis the reward. The MANOVA test returned a $p$-value of $0.1267$. This confirms that there is no enough evidence to conclude that the means are different.

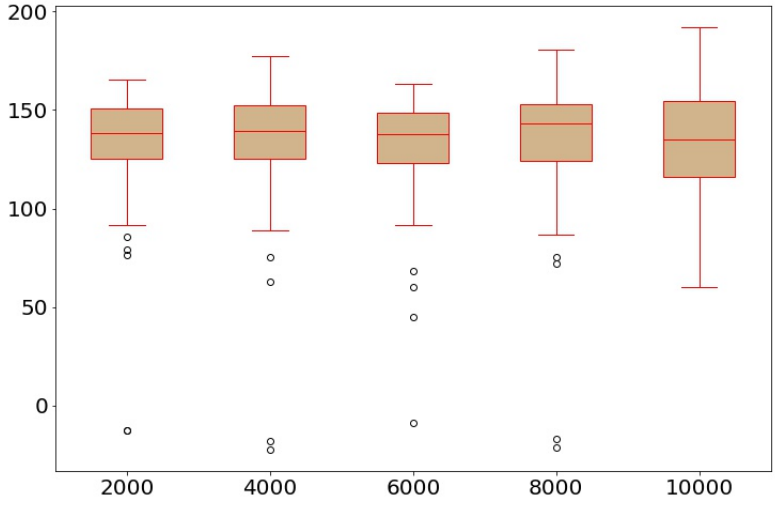



Figure B3: Boxplot representing the results of MD-MADDPG on Sequential CN when changing different seeds. The horizontal axis shows the seed and the vertical axis the reward. The MANOVA test returned a $p$-value of $0.8357$. This confirms that there is no enough evidence to conclude that the means are different.

### B.7 Multiple memory sizes

Figure B4 represents how the memory size affects the resulting reward on Swapping Cooperative Navigation. It can be noted that given the selected architecture, the higher reward is obtained using a memory size of 200.

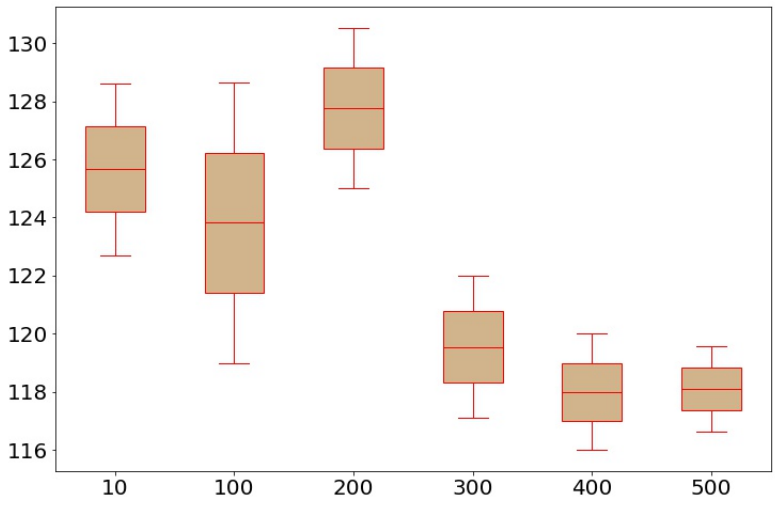
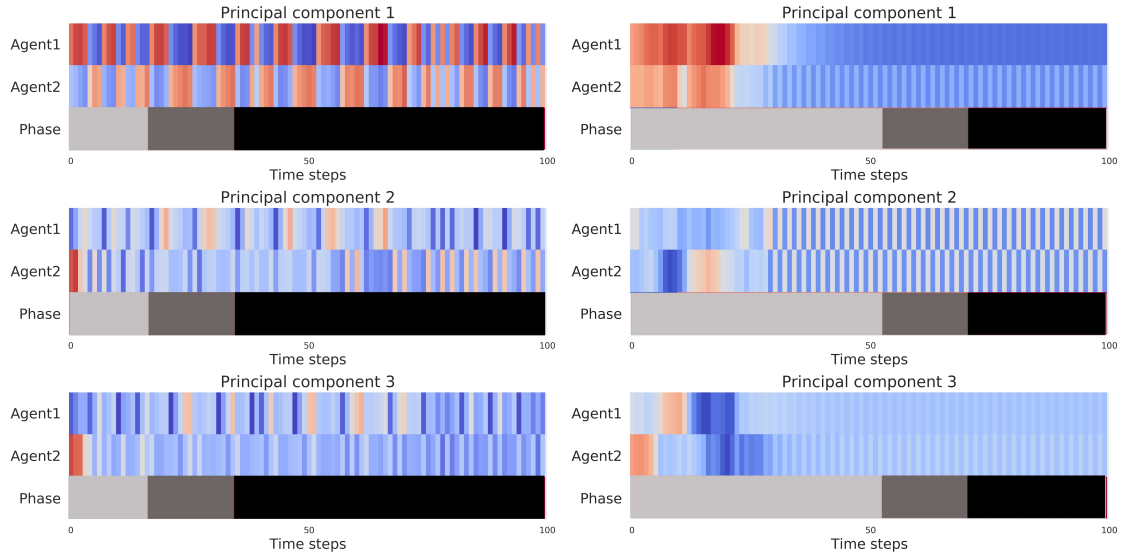



Figure B4: Results obtained on Swapping CN using different memory dimensions. The horizontal axis report the memory size and the vertical axis the reward.

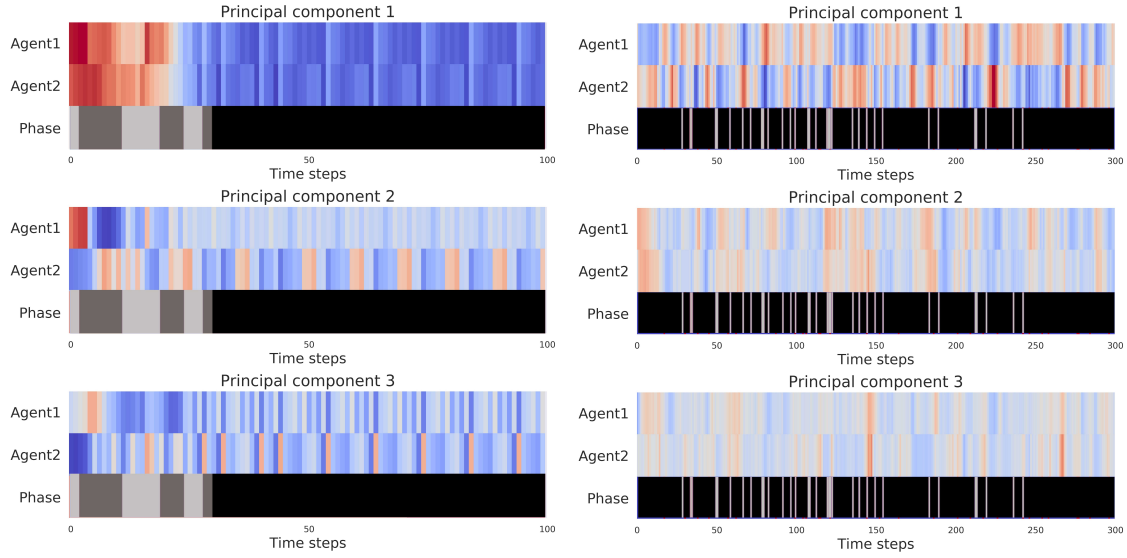
### B.8 Communication analysis - Read Vector

Figure B5 shows the results of a communication analysis analogous to Section 4.4 but for the read vector. As for the write vector, communication patterns emerge and seem to point out that communication is more intense when coordination is highly required. It can be noted that the read vectors still correlate with the phases. For example, in Synchronous Cooperative Navigation, the first principal component is highly activated during phases 1 and 2. This suggests that agents intensely communicate to reach the landmarks simultaneously. A different behaviour emerges for others environments like Swapping Cooperative Navigation where the reading vector is highly activated during the first phase, probably because the agents received the information about the next landmark to swap (e.g. coordinates). This analysis has been conducted to better present the behaviour of the agents and the interactions with their internal components. We believe that the content of the message, which corresponds to the write vector, is more informative since it is what the agents are explicitly communicating. On the other side the read vector can be more difficult to interpret since it is internally used by the agents together with other components to achieve communication.




(a) Sequential CN

(b) Swapping CN




(c) Synchronous CN

(d) Waterworld

Figure B5: Visualisation of communication strategies learned by the agents in four different environments: the three principal components provide orthogonal descriptors of the read vector content of the agents and are being plotted as a function of time. Within each component, the highest values are in red, and the lowest values are in blue. The bar at the bottom of each figure indicates which phase (or sub-task) was being executed within an episode.