Communication-based Cooperative Tasks: how the Language Expressiveness affects Reinforcement Learning

Jacopo Talamini  
DIA - Università di Trieste  
Trieste, Italy  
jacopo.talamini@phd.units.it

Eric Medvet  
DIA - Università di Trieste  
Trieste, Italy  
emedvet@units.it

Alberto Bartoli  
DIA - Università di Trieste  
Trieste, Italy  
bartoli.alberto@units.it

ABSTRACT
We consider a cooperative multi-agent system in which cooperation may be enforced by communication between agents but in which agents must learn to communicate. The system consists of a game in which agents may move in a 2D world and are given the task of reaching specified targets. Each agent knows the target of another agent but not its own, thus the only way to solve the task is for the agents to guide one another using communication and, in particular, by learning how to communicate. We cast this game in terms of a partially observed Markov game and show that agents may learn policies for moving and communicating in the form of a neural network by means of reinforcement learning. We investigate in depth the impact on the learning quality of the expressiveness of the language, which is a function of vocabulary size, number of agents and number of targets.

CCS CONCEPTS
• Computing methodologies → Cooperation and coordination; Multi-agent systems; Neural networks; • Theory of computation → Multi-agent reinforcement learning; • Computer systems organization → Robotic autonomy;

KEYWORDS
Multi-Agent Systems, Reinforcement Learning, Neural Networks, Agent Communication, Language Expressiveness

ACM Reference Format:

1 INTRODUCTION
Artificial intelligence nowadays plays an increasingly pervasive role in everyday life and what was only a research topic a few years ago is now an essential part of the technology industry [18]. An important goal in this scenario consists in the creation of artificial agents able to solve complex problems requiring forms of perception of the surrounding environment. Multi-agent systems, in particular, involve many artificial agents interacting in the same environment with common and/or conflicting goals, resulting in cooperative and/or competitive scenarios.

For example, the controller of an autonomous vehicle can be modeled as a multi-agent system of sensors and actuators that work together, so that collective performance depends on the cooperation among all of them. The cooperation involves the exchange of information between agents along with the evaluation of the action to take, based on the received information.

Language interpretability and efficiency are important requirements in this context. Natural language has the advantage of being understandable by humans, but its complexity and the fact that the meaning of a term may depend on the context in which it appears (i.e., on nearby terms) may nullify the potential benefits of the learning process [19].

A symbol-based language, with no predefined meaning for symbols, is not easily interpretable by humans, but could lead to more efficient cooperation among artificial agents and could be more appropriate for a learning task than natural language.

Motivated by recent works on multi-agent systems [13, 14], we consider a symbol-based approach to solve a cooperative multi-agent task in which agents learn to communicate using reinforcement learning (RL). We consider a multi-agent scenario in which agents may move in a 2D world and are given the task of reaching specified targets. Each agent knows the target of another agent but not its own, thus the only way to solve the task is for the agents to guide one another using communication and, in particular, by learning how to communicate.

We use the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) learning algorithm [13] for learning the control policy of the agents in a fully decentralized way. We focus on investigating the relation between vocabulary size (number of communication symbols available), number of agents, and number of possible targets. To this end, we define a single expressiveness numerical index that depends on these quantities and investigate its effect on the degree of accomplishment of the cooperative task at the end of the learning process.

Given an instance of the problem, with a fixed number of agents and targets, we find a vocabulary size that leads to best rewards w.r.t. the number of agents and targets. We compare these results to two baselines that model the two extreme scenarios: one in which agents cannot communicate and another in which agents have a predefined ability to communicate. Finally we show that, if expressiveness is beyond a certain threshold, the learning process struggles to learn how to communicate and, consequently, performs poorly.
2 RELATED WORK
Our work concerns the language used for communication among agents in a cooperative scenario where the learning of agent controllers is tackled with RL. In the following sections, we briefly survey relevant previous studies related this scenario.

In particular, we focus on language and communication in multi-agent systems (Section 2.1) and on RL as tool for training the agent controllers in these settings (Section 2.2). Indeed, most of the cited works involve both aspects. We decide to present each work in the section related to the work most relevant part of the contribution.

It is worth to note that our work somewhat investigates the broader field studying how the (natural) language has evolved. The reader may find in [4] a comprehensive analysis of that field.

2.1 Language and communication
In many works the agents are viewed as a complex adaptive system which collectively solves the problem of developing a shared communication model [17]. To do so, the community must reach an agreement on a repertoire of forms, of meanings, and of form-meaning pairs (the lexicon and grammar). In general, this task may be hard: the authors of [10] discuss the learning bottleneck and its relation with the phrase structure and, more broadly, with language compositionality.

In the communication-oriented multi-agent scenario of [8], the authors implement the agents through recurrent neural networks, allowing language compositionality and variability. Their focus is on the development of natural language, and the results are obtained by fixing the vocabulary size to an arbitrary value. The works cited above are based on language compositionality and grammar complexity, an aspect we do not deal with.

In [20] the authors train a robotic agent controlled by a recurrent neural network to learn the relationship between sequences of words and corresponding actions. They consider a symbol-based language in which every word is encoded as one-hot vector and visualize the agent internal representation of the state after the training. In the cited work there is no multi-agent interaction, as in our work, but a single agent model is trained to learn sentence-action associations. Moreover, rather than focusing on expressiveness, [20] assesses the quality of the learning w.r.t. the presence of specific logical expression in the uttered word sequences.

In [1] the authors claim that language is a mechanism that emerges for coordinating the solution of complex tasks among agents. In [2] the same authors show that different level of “friendship” between agents co-evolve with a system of linguistic conventions: agents decisions during individual interactions influence the overall social structure of the population. Both works consider a tool, called Language Evolution Workbench (LEW), which defines the task, the scenario, and the learning procedure. Based on simulations using LEW, they study the impact of the probability of communication success on the task achievement: moreover, they characterize the former in terms of lexicon size and amount of lexicon actually used. This analysis is similar to the one we do, even if the scenario and the task are different.

2.2 Reinforcement learning for agent controllers
RL has reached outstanding results in single-agent control domain [6] and several recent works have extended this approach to the training of cooperative multi-agent systems, also when communication among agents plays a role [17]. In [17] and in [8] communication is enforced in a referential game between two cooperating agents: one agent goal is to explain which image the other agent should select from a pool of images.

In [7] the authors investigate the process of learning to communicate and how this occurs among agents. To this extent they propose two multi-agent frameworks for communicative tasks based on reinforcement learning.

The focus of [16] is on the meaning-word association in a population of autonomous real robots. The robots play a guessing game in which one has to communicate the other that he identified an object, acquired through a camera. The works cited above use RL to learn the agent controllers, including the communication part, but focus on different scenarios: in particular, they deal with cases where the communication involves two agents.

Recently, another RL variant tailored to communication-based tasks has been presented in [3]. The authors propose to approximate the reward function, used to assess the agent behaviour, with a discriminator neural network trained on a data set of states. In this way, the representation of the solved task is separated from how to solve it. This approach has been showed to outperform vanilla RL on a range of tasks, and combines RL with supervised learning.

Another fully decentralized multi-agent RL approach for communication-based tasks has been proposed by [5]. The cited work shows that agents may indeed learn to communicate without any prior information. Agents learn a low-level wireless protocol with bidirectional communication. As in our work, the authors also provide baselines for both transmitter and receiver, and they learn only one of the two. Training quality is assessed with varying levels of noise in the communication channel.

The approach that we use for training the agents is Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [13], a multi-agent variation of deterministic policy gradient [15]. The cited work shown that MADDPG fits several different RL scenarios, and proposed its application in many cooperative and competitive multi-agent settings.

In [14] and [8] the authors present multi-agent scenarios with communication, in which an end-to-end differentiable model is trained through backpropagation. In particular in [8] this type of learning is compared with standard RL.

We consider a scenario similar to the one in [14], in which agents move in a 2D word and communicate using symbol-based language and agent controllers are implemented by neural networks. In the cited work, the authors consider an end-to-end differentiable approach for training: they assume differentiable system dynamics, so that the networks can be updated back-propagating an arbitrary reward scalar. This work is different from our work in terms of learning approach. Also the cited work does not evaluate the impact of expressiveness on the training results, focusing instead on analyzing the actual number of words emitted by the agents with varying vocabulary size.
A mixed cooperative-competitive scenario has been studied by [11], where two opposing agents have to negotiate a deal, in order to maximize their opposite goals. The two agents use natural language for the communication and the baseline for the training is supervised learning using a data set of dialogues on negotiation tasks. The agents are then trained from this baseline by using RL and self-play. Again we do not use supervised learning, nor natural language, and the agents in our work play a cooperative game, instead of competing against each other.

3 BACKGROUND

3.1 Markov game

We consider a multi-agent version of a Markov Decision Process (MDP), called partially observed Markov game [12] (or, here briefly, Markov game), in which a number of agents interact in the same discrete-time environment.

A Markov game involving \( n \) agents is described by a tuple \((S, \phi, \rho, O, \mathcal{A}, \Omega, \Pi, R)\), where \( S \) is the set of possible states of the game, \( \phi \) is the stochastic state transition function, \( \rho \) is the stochastic initial state function, \( O = (O_1, \ldots, O_n) \) is the set of agent-dependent observations sets, \( \mathcal{A} = (A_1, \ldots, A_n) \) is the set of agent-dependent actions sets, \( \Omega = (\omega_1, \ldots, \omega_n) \) is the set of agent-dependent stochastic observation functions, \( \Pi = (\pi_1, \ldots, \pi_n) \) is the set of agent-dependent stochastic policies, and \( R = (r_1, \ldots, r_n) \) is the set of agent-dependent reward functions.

The state transition function \( \phi : S \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \to [0, 1] \) stochastically determines the evolution of the game, i.e., \( \phi(s' | s, a_1, \ldots, a_n) \) is the probability that the game goes from state \( s \) to state \( s' \) given that the agents performed the actions \( a_1, \ldots, a_n \). It holds that \( \forall s \in S, \forall(a_1, \ldots, a_n) \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_n : \sum_{s' \in S} \phi(s' | s, a_1, \ldots, a_n) = 1 \).

The initial state function \( \rho : S \to [0, 1] \) stochastically determines the initial state of the game, i.e., \( \rho(s) \) is the probability that the game starts from the state \( s \). It holds that \( \sum_{s \in S} \rho(s) = 1 \).

Each observation function \( \omega_i : O_i \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \to [0, 1] \) stochastically determines the observation of the corresponding agent, i.e., \( \omega_i(o, a) \) is the probability that the \( i \)th agent observes the observation \( o \) in \( O_i \), given that it performed the action \( a \in \mathcal{A}_i \) with the game being in state \( s \). It holds that \( \forall i \in \{1, \ldots, n\}, \forall o \in O_i, \forall a \in \mathcal{A}_i : \sum_{o' \in O_i} \omega_i(o', a) = 1 \).

Each policy \( \pi_i : O_i \times \mathcal{A}_1 \to [0, 1] \) stochastically determines the behavior of the corresponding agent, i.e., \( \pi_i(o, a) \) is the probability that the \( i \)th agent performs the action \( a \in \mathcal{A}_i \) given that it observed the observation \( o \in O_i \). It holds that \( \forall i \in \{1, \ldots, n\}, \forall o \in O_i : \sum_{a \in \mathcal{A}_i} \pi_i(o, a) = 1 \).

Each reward function \( r_i : S \times \mathcal{A}_1 \to \mathbb{R} \) determines the reward an agent receives, i.e., \( r_i(s, a) \) is the reward the \( i \)th agent receives for having performed the action \( a \in \mathcal{A}_i \) with the game in state \( s \in S \).

3.2 Policy learning

Let \( \gamma \in (0, 1) \) be a predefined constant called discount factor and let \( T \) be a predefined time length called the time horizon. The policy learning problem consists in finding, given \( S, \mathcal{O}, \mathcal{A}, R, \) and \( \gamma, T \), the \textit{optimal policies} \( \Pi^* \) which maximize the expectation \( \mathbb{E}[r_h] \), of the overall reward \( r_h \) for any \( t_0 \), defined as

\[
r_{t_0}^* = \sum_{i \in \{1, \ldots, n\}} \sum_{t = t_0}^{t_0 + T} \gamma^t r_i(s^t, a_i^t)
\]

where \( s^t \) is the state of the game at the \( t \)th step and \( a_i^t \) is the action performed by the \( i \)th agent at the \( t \)th step. Parameters \( \gamma \) and \( T \) specify to which degree the agents employing the optimal policies \( \Pi^* \) act towards an immediate reward (short \( T \), small \( \gamma \)) or a future reward (long \( T \), \( \gamma \approx 1 \)).

It can be noted that the state transition function \( \phi \), the initial state function \( \rho \), and the observation functions \( \Omega \) are not available in the policy learning problem. Instead, it is possible to sample the corresponding distribution by playing the game as many times as needed. That is, given a policies set \( \Pi \), it is possible to play the game for a finite number \( T_{\text{episode}} \) of time steps (i.e., an episode) and obtaining the corresponding values of \( s^t, a_i^t, r_i^t \) for \( t = 0, \ldots, T_{\text{episode}} \)—and, as a consequence, the rewards \( r_i(s^t, a_i^t) \) for \( t = 0, \ldots, T_{\text{episode}} \) and the overall rewards \( r_i^\Pi \) for \( t = 0, \ldots, T_{\text{episode}} - T \).

In many cases of interest, the policies to be learned have all the same form (e.g., a neural network) which can be modeled by a finite set \( \Theta \) of numerical \textit{policy parameters}. In those cases, the policy learning problem consists in learning the optimal values \( \Theta^* = \{ \theta^*_1, \ldots, \theta^*_n \} \) of the policy parameters.

The policy learning problem can be solved using RL, a form of machine learning which tries to balance exploration (i.e., sampling \( \phi \), \( \rho \), and \( \Omega \) and exploitation (i.e., maximizing the overall rewards \( r_i^\Pi \)) while searching in the space of the policy parameters by varying \( \Theta \). In this work, we use a RL technique called Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [13] which has been shown to be effective for scenarios similar to the one considered in this work.

4 THE COOPERATIVE UNKNOWN TARGET GAME

We consider a cooperative game similar to the one presented in [14], which we call the \textit{Cooperative Unknown Target (CUT) game}. The game is based on 2D world in which \( n_t \) robots move and communicate. The goal of each robot is to reach one among \( n_t \) target positions in the world, the association between robot and target being statically determined before the game starts. Each robot knows the target of one robot but \textit{not} its own target. It is guaranteed that the target of each robot is known to exactly one robot. Robots communicate by sending and receiving symbols of a finite alphabet \( W \) (the language). At each time step, a robot broadcasts exactly one symbol: each sent symbol is received by every robot and the sender is known to the receivers. Robots are stateless: in particular, they do not remember the symbols they received before the current time step.

Communication among robots is thus essential for allowing each robot to reach its own target, which characterizes the CUT game as a cooperative one. The game is challenging because a robot should learn to: (i) move toward its target, to be deduced from what it hears; (ii) broadcast a symbol describing the single robot-target association that it knows.
The CUT game can be modeled as a Markov game. In the following sections, we detail how the CUT game maps to the Markov game abstraction and which is the form we chose for the corresponding policies, that we then learned using MADDPG. As anticipated in Section 1, the aim of the present paper is to investigate the impact of the language size on the effectiveness and efficiency of the policy learning; we do not insist on discussing the optimality of the game mapping, nor on thoroughly evaluating the MADDPG performance on the CUT game.

4.1 The CUT game as a Markov game

The key idea behind the mapping of the CUT game to the Markov game abstraction is to represent each robot in the former as two separate agents in the latter: one (the movement-agent) determines the movements of the robot, the other (the communication-agent) determines the output communication (i.e., the symbols sent by) the robot. The motivation for this choice is to allow for the decoupling of the two different goals of the robot: (i) moving towards its target and (ii) communicating the known robot-target association.

In particular, the two corresponding reward functions can be hence defined to reflect the two goals.

More in detail, the state of the CUT game encodes:

- the positions $\vec{x}_{r,1}, \ldots, \vec{x}_{r,n_r}$ of the $n_r$ robots, with $\vec{x}_{r,i} \in \mathbb{R}^2$ for each $i$;
- the speeds $\vec{v}_{1}, \ldots, \vec{v}_{n_r}$ of the $n_r$ robots, with $\vec{v}_i \in \mathbb{R}^2$ for each $i$;
- the positions $\vec{x}_{t,1}, \ldots, \vec{x}_{t,n_t}$ of the $n_t$ targets, with $\vec{x}_{t,i} \in [0,1]^2$ for each $i$;
- the robot-target associations $(\tau_{r,1}, \tau_{t,1}), \ldots, (\tau_{r,n_r}, \tau_{t,n_t})$ known by each robot, with $\tau_{r,i} \in \{1, \ldots, n_t\}$ and $\tau_{t,i} \in \{\tau_{r,1}, \ldots, \tau_{r,n_r}\}$ for each $i$;
- the symbols $w_1, \ldots, w_{n_t}$ sent by $n_t$ robots at the previous time step, with $w_i \in W \cup \varnothing$ for each $i$.

All the robots in the CUT game can move and communicate in the same way. Let $A^\text{move} = \{\uparrow, \downarrow, \leftarrow, \varnothing\}$ be the set containing the actions which can be performed by the movement-agent of a robot. The observation functions of the Markov game are not actually stochastic and depend only on the state (not on the performed action). We denote the state with $s$. For movement-agents, $w^\text{move}_i : s \mapsto O^\text{move}$ gives the offset of the $i$th agent from each target along the two axes (a pair of values in $\mathbb{R}$ for each target) and the symbols emitted at the previous step by each agent (one symbol in $W \cup \varnothing$):

$$w^\text{move}_i(s) = (\vec{x}_{t,1} - \vec{x}_{r,i}, \ldots, \vec{x}_{t,n_t} - \vec{x}_{r,i}, w_1, \ldots, w_{n_t})$$

For communication-agents, $w^\text{comm}_i : s \mapsto O^\text{comm}$ gives the robot-target association known to the $i$th robot:

$$w^\text{comm}_i(s) = (\tau_{r,i}, \tau_{t,i})$$

The robot-target associations $\tau_{r,i}, \tau_{t,i}$ never change, thus every communication-agent observes a constant observation during the game.

Finally, the reward functions of the Markov game are defined so as to capture the goal of the CUT game. For the movement-agents, $r^\text{move}_i$ rewards the agent when it is able to get closer to its target. For communication-agents, $r^\text{comm}_i$ rewards the agent when it is able to make its "recipient" $j$th to get closer to its target, with $j = \tau_{r,i}$.

In detail, $r^\text{move}_i : s \mapsto \mathbb{R}$ gives the opposite of the Manhattan distance (i.e., the closer, the greater the reward) of the $i$th robot from its target:

$$r^\text{move}_i(s) = \sum_{j=1}^{n_t} -d_1(\vec{x}_{r,i}, \vec{x}_{t,\tau_{t,i}}) I_1(s, j)$$

where:

$$I_1(s, j) = \begin{cases} 1 & \text{if } \tau_{r,j} = i \\ 0 & \text{otherwise} \end{cases}$$

The reward function $r^\text{comm}_i : s \mapsto \mathbb{R}$ gives the opposite of the Manhattan distance of the robot whose target association is known to the $i$th robot from its target:

$$r^\text{comm}_i(s) = -d_1(\vec{x}_{r,\tau_{r,i}}, \vec{x}_{t,\tau_{t,i}})$$

Both reward functions depend only on the state (not on the performed action) and are deterministic.

4.2 Policies form

We consider policies in the form of a neural network, thus the policy learning problem consists in learning the optimal values $\Theta^* = \{\theta^*_1, \ldots, \theta^*_n\}$ of the parameters of the network. We opted for a Multi-Layer Perceptron (MLP) as the form of the policies for the movement- and communication-agents, which can hence be denoted as $\pi^\text{move}_\theta : A^\text{move} \mapsto A^\text{move}$ and $\pi^\text{comm}_\theta : A^\text{comm} \mapsto A^\text{comm}$, respectively. We chose to use the same MLP topology for all the
movement-agents policies and the same MLP topology for all the communication-agents policies.

The input of $\pi_{0i}^\text{move}$ consists of a tuple of $2n_t + n_r$ elements, as defined in Section 4.1. The first $2n_t$ elements are the offset of the $i$th agent from each target along the two axes; each of these elements is mapped to the input layer directly. The remaining $n_r$ elements are the symbols emitted at the previous step by each agent (i.e., one symbol in $W \cup \emptyset$); each of these elements is represented in one-hot encoding, thus each of these elements correspond to $|W| + 1$ input values. The resulting size for the input layer will be $2n_t + n_r(|W|+1)$. The output of $\pi_{0i}^\text{move}$ is an element of $A^\text{move}$ (Section 4.1). This element is represented in one-hot encoding, thereby resulting in $|A^\text{move}| = 5$ output neurons.

The input of $\pi_{0i}^\text{comm}$ consists of a pair of elements corresponding to the robot-target association known to the $i$th agent $\tau_{r,i}, \tau_{t,i}$, as defined in Section 4.1. Both elements are represented in one-hot encoding, resulting in $n_r + n_t$ input values. The output of $\pi_{0i}^\text{comm}$ is an element of $A^\text{comm} = W \cup \emptyset$ (Section 4.1). This element is mapped to $|W| + 1$ output neurons with one-hot encoding.

For both the policies, we set 2 fully connected hidden layers, each with 64 neurons, and we used the Rectifier Linear Unit (ReLU) activation function as done in [13]. Finally, the policies output are sampled from the Gumbel-Softmax distribution [9] which makes the policies actually stochastic: this, in turn, allows for a better exploration of the search space while learning the policies with MADDPG.

### 4.3 Hand-made communication-agents policies

In order to allow for a more insightful investigation of the impact of the language size on the policy learning effectiveness and efficiency, we designed two communication-agents policies to use as baselines. The two policies represent two extreme cases, one of non-communicating agents and one of agents which optimally communicate with a language of predefined size.

The first hand-made policy, which we denote with NoComm, is defined by:

$$\pi_{\text{NoComm}}(\tau_{r,i}, \tau_{t,i}) = \emptyset$$  \hspace{1cm} (7)

Note that, since the output of $\pi_{\text{NoComm}}$ is always $\emptyset$, this policy works with any $W$: in particular, it works with $W = \emptyset$.

The second hand-made policy, which we denote with Opt, is defined by:

$$\pi_{\text{Opt}}(\tau_{r,i}, \tau_{t,i}) = W_{\tau_{r,i}, \tau_{t,i}}^{\text{Opt}}$$  \hspace{1cm} (8)

where $w_{\tau_{r,i}, \tau_{t,i}}^{\text{Opt}}$ is a symbol of a language $W^{\text{Opt}}$ that can express all the possible robot-agent associations. That is, in $W^{\text{Opt}}$ each robot-agent association is unequivocally encoded by one symbol: $W^{\text{Opt}} = \{ w_{i,j}, 0 \leq i \leq n_r, 0 \leq j \leq n_t \}$.

### 5 EXPERIMENTAL EVALUATION

#### 5.1 Procedure

We emphasize that each agent learns to communicate and to understand the received symbols independently of the other agents. Thus, a given symbol could be associated with different meanings by the communication-agent policy $\pi_{\text{comm}}$ of different agents. It is thus important to gain insights into the relation between the number of symbols available for communication $|W|$, the number of robots $n_r$, and the number of targets $n_t$.

In order to decouple the experimental findings from the size of the policy learning problem (i.e., the number $n_r$ of robots and the number $n_t$ of targets), we define the expressiveness of the language as $\varepsilon = \frac{|W|}{n_r \times n_t}$. Others may define expressiveness in different ways, for instance as the number of times symbols have been uttered. We investigated the impact of expressiveness on the $\varepsilon$ and efficiency of the policy learning in the CUT problem.

We performed policy learning of both $\pi_{\text{move}}$ and $\pi_{\text{comm}}$ for a number of different combinations of $(n_r, n_t, |W|)$ corresponding to $\varepsilon \in [0, 4]$, as follows. We considered all combinations of $(n_r, n_t) \in \{2, 3, 4\} \times \{2, 3, 4\}$; for each such combination we considered all values for $|W| \in \{0, \ldots, 4n_r n_t \}$.

For each combination of $(n_r, n_t, |W|)$, we executed an experiment consisting of 20 000 learning episodes followed by the evaluation of the resulting policies for 100 validation episodes. The learning parameters are listed in table 1.

For the baseline policies we performed policy learning only for the agent movement policy $\pi_{\text{move}}$ because agent communication policies $\pi_{\text{comm}}^{\text{NoComm}}$ and $\pi_{\text{comm}}^{\text{Opt}}$ were specified in advance. We considered all combinations of $(n_r, n_t) \in \{2, 3, 4\} \times \{2, 3, 4\}$, each combination with a single value for $|W|$, as follows. With $\pi_{\text{comm}}^{\text{NoComm}}$ the language is $W = \emptyset$, thus $|W| = \varepsilon = 0$. With $\pi_{\text{comm}}^{\text{Opt}}$ the language size is $|W| = n_r n_t$, thus the expressiveness is $\varepsilon = 1$.

#### 5.2 Results and discussion: effectiveness

In this section we assess the effectiveness of the policy learning in all the problem settings of the experimental campaign. To this end, we computed the validation reward $R_{\text{val}}$ for each experiment, defined as the average overall reward of the agents on the validation episodes. Figure 1 shows the normalized validation reward averaged across all the experiments—i.e., for each $(n_r, n_t)$ the values of $R_{\text{val}}$ for different $\varepsilon$ are adjusted to have null mean and standard deviation equal to 1.

The most important finding is that the highest validation reward corresponds to values of $\varepsilon$ close to 1, i.e., when $|W|$ is close to $n_r n_t$. On the other hand, when $\varepsilon \gg 1$ the action space grows

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discount factor $\gamma$</td>
<td>0.95</td>
</tr>
<tr>
<td>Time horizon $T$</td>
<td>25</td>
</tr>
<tr>
<td>Episode time steps $T_{\text{episode}}$</td>
<td>25</td>
</tr>
</tbody>
</table>

### Table 1: Parameters.

...
Figure 1: Normalized validation reward $R_{\text{val}}$ vs. the expressiveness $e$, average across all the experiments.

unnecessarily and the learning process is negatively affected. In principle, one could try to tackle the growth of the action space by increasing the size of the MLP: we did not perform any experiments to investigate this opportunity.

Figure 1 also shows that the validation reward with $e = 0$ is poor. This is not surprising, because when the language does not allow to communicate, an agent does not have any information about its target, thus the learned policy can only tend to minimize the distance between a robot and all targets. In summary, this figure shows that the policy learning process is indeed effective in learning how to communicate, in order to solve the specific cooperative task. Furthermore, it shows that the policy learning is most effective when the size of the language is sufficiently large to (potentially) allow associating approximately one symbol with each possible robot-target association.

Figure 2 provides the relation between validation reward and expressiveness in a more granular form. The figure contains a curve for each pair $(n_r, n_t)$ considered (in each curve, the number of symbols $|W|$ varies so as to span the interval $[0, 4]$ for the expressiveness). The validation reward for $\pi^\text{comm}_\text{Opt}$ is represented by a dot lying at $e = 1$.

The impact of expressiveness on validation reward is more evident for larger values of $n_r$ and $n_t$: it can be seen that the upper lines (corresponding to lower values of $n_r$ and $n_t$) tend to exhibit a constant validation reward, except for very low values of expressiveness, while the bottom lines show a degradation behavior when expressiveness grows. It can also be seen that the gap between $\pi^\text{comm}_\text{Opt}$ and the learned policies is higher for larger values of $n_r$ and $n_t$.

Further insights on the relation between validation reward and expressiveness can be obtained from Table 2 and Figure 3. In particular, Table 2 shows that when $e$ is sufficiently large (i.e., when $|W|$ is at least equal to $n_r n_t$) $R_{\text{val}}$ is higher if $n_t$ is low and viceversa. This result confirms the intuition that cooperating toward reaching less targets is easier than toward reaching more targets. On the other hand, when the language is not expressive enough (i.e., when $e$ is too small), the resulting behavior is unpredictable in the sense that there is no intuitive relationship between $n_r n_t$ and $R_{\text{val}}$. Interestingly, though, Figure 3 shows that there is significant variability in the outcome of policy learning, more so when expressiveness is large: the first quartile of the distribution for $e = 3.0$ spans nearly the same range of values as the whole distribution.

Table 2: Validation reward for selected values of expressiveness.

<table>
<thead>
<tr>
<th>$n_r$</th>
<th>$n_t$</th>
<th>$e = 0.25$</th>
<th>$e = 1.0$</th>
<th>$e = 3.0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>−37.84</td>
<td>0.10</td>
<td>−13.80</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>−40.89</td>
<td>0.12</td>
<td>−15.48</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>−28.80</td>
<td>0.03</td>
<td>−16.02</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>−53.87</td>
<td>0.17</td>
<td>−24.92</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>−36.75</td>
<td>0.12</td>
<td>−30.32</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>−47.66</td>
<td>0.10</td>
<td>−41.72</td>
</tr>
</tbody>
</table>

5.3 Results and discussion: efficiency

In this section we assess the efficiency of the policy learning. To this end, we computed the average episode reward (the learning reward $R_{\text{learn}}$) every 512 episodes of the learning process.
Figure 3: Boxplot of the normalized validation reward $R_{val}$ for selected values of expressiveness $e$.

Figure 4 shows how $R_{learn}$ changes during the learning for 3 combinations of $n_r$, $n_t$ (one for each plot). For each combination we considered 3 values for expressiveness and the baseline policies. With $n_r = n_t = 2$, all policies reach the respective maximal learning reward relatively quickly, thus very efficiently. It can also be observed a sharp separation between two groups of policies, one including $\pi_{\text{NoComm}}$ and the learned policy with $e = 0.25$ and another including all the other policies. With $n_r = n_t = 3$, the baseline policies exhibit the same efficiency as in the previous scenario, while the learned policies require more episodes to converge to their maximal learning reward. The fact that the baseline policies converge more quickly may be explained with the fact that these policies have to learn only a movement policy. Interestingly, in this case the learned policies always exhibit a learning reward in between the two baselines. With $n_r = n_t = 4$, on the other hand, $\pi_{\text{Opt}}$ is much slower for reaching its maximal learning reward, while all the other policies converge more quickly. We interpret this result as a combination of the larger size of the search space coupled with the ability of $\pi_{\text{Opt}}$ to indeed cope with such a larger space better than the other policies, i.e., still being able to learn.

Table 3 shows similar information of Figure 3 for all the experiments. For each combination of $n_r$, $n_t$ and each value of $e$ in a set of three selected values (and each of the two baselines), the table shows the number of episodes (in thousands) the learning took to reach 95% of the final value of the learning reward $R_{learn}$.

### Table 3: Number of episodes ($10^3$) required for reaching 95% of the final learning reward.

<table>
<thead>
<tr>
<th>$n_r$</th>
<th>$n_t$</th>
<th>NoComm</th>
<th>Opt</th>
<th>$e$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>13</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>11</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>10</td>
<td>12</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>14</td>
<td>16</td>
<td>13</td>
<td>3</td>
</tr>
</tbody>
</table>

6 CONCLUDING REMARKS

We considered a cooperative multi-agent system in which communication between agents is required for accomplishing the task but in which agents must learn to communicate. Specifically, each agent must learn its target from what it hears, learn to move toward that target, and learn to broadcast information useful to other agents. We have considered a symbol-based language and investigated the impact of the expressiveness of the language, which is a function of vocabulary size, number of agents, and number of targets, on both effectiveness and efficiency of the learning process. We have shown that agents including two separate neural networks, one for encoding a movement policy and another for encoding a communication policy, may indeed learn policies for moving and communicating by means of reinforcement learning. We have also shown that the best effectiveness is obtained when the vocabulary size is close to the product between number of agents and number of targets: a smaller vocabulary is not expressive enough while a larger vocabulary makes it more difficult to explore the resulting larger search space.

### REFERENCES


Figure 4: Learning reward $R_{\text{learn}}$ during the learning, one curve for each of three selected expressiveness values $e$ and each baseline, one plot for each combination of $n_r, n_t$.

