HAMMER: Multi-Level Coordination of Reinforcement Learning Agents via Learned Messaging

Nikunj Gupta
IIIT-Bangalore
Nikunj.Gupta@iiitb.org

G. Srinivasaraghavan
IIIT-Bangalore
gsr@iiitb.ac.in

Swarup Kumar Mohalik
Ericsson Research
swarup.kumar.mohalik@ericsson.com

Nishant Kumar
IIT-BHU
nishantkr.ece18@itbhu.ac.in

Matthew E. Taylor
University of Alberta & Alberta
Machine Intelligence Institute (Amii)
matthew.e.taylor@ualberta.ca

ABSTRACT

Cooperative multi-agent reinforcement learning (MARL) has achieved significant results, most notably by leveraging the representation-learning abilities of deep neural networks. However, large centralized approaches quickly become infeasible as the number of agents scale, and fully decentralized approaches can miss important opportunities for information sharing and coordination. Furthermore, not all agents are equal — in some cases, individual agents may not even have the ability to send communication to other agents or explicitly model other agents. This paper considers the case where there is a single, powerful, *central agent* that can observe the entire observation space, and there are multiple, low-powered *local agents* that can only receive local observations and are not able to communicate with each other. The central agent's job is to learn what message needs to be sent to different local agents based on the global observations, not by centrally solving the entire problem and sending action commands, but by determining what additional information an individual agent should receive so that it can make a better decision. In this work we present our MARL algorithm HAMMER, describe where it would be most applicable, and implement it in the cooperative navigation and multi-agent walker domains. Empirical results show that 1) learned communication does indeed improve system performance, 2) results generalize to heterogeneous local agents, and 3) results generalize to different reward structures.

KEYWORDS

Multi-agent Reinforcement Learning, Learning to Communicate, Heterogeneous Agent Learning

1 INTRODUCTION

The field of multi-agent reinforcement learning (MARL) combines ideas from single-agent reinforcement learning (SARL), game theory, and multi-agent systems. Cooperative MARL calls for simultaneous learning and interaction of multiple agents in the same environment to achieve shared goals. Applications like distributed logistics [60], package delivery [42], and disaster rescue [35] are domains that can be modeled naturally using this framework. However, even cooperative MARL suffers from several complications inherent to multi-agent systems, including non-stationarity [3], a potential need for coordination [3], the curse of dimensionality [44], and global exploration [29].

Multi-agent reasoning has been extensively studied in MARL [29, 34] in both centralized and decentralized settings. While very small systems could be completely centralized, decentralized implementation becomes indispensable as the number of agents increase, and to cope with the exponential growth in the joint observation and action spaces. However, it often suffers from synchronization issues [29] and complex teammate modeling [1]. Moreover, independent learners may have to optimize their own, or the global, reward from only local, private observations [52]. In contrast, centralized approaches can leverage global information and mitigate non-stationarity through full awareness of all teammates.

In MARL, communication has been shown to be an important aspect, especially in tasks requiring coordination. For instance, agents tend to locate each other or the landmarks more easily using shared information in navigation tasks [12]. Communication can also influence the final outcomes in group strategy coordination [15, 59]. There have been significant achievements using explicit communication in video games like StarCraft II [36] as well as in mobile robotic teams [28], smart-grid control [37], and autonomous vehicles [4]. Communication can be in the form of sharing experiences among the agents [61], sharing low-level information like gradient updates via communication channels [8] or sometimes directly advising appropriate actions using a pretrained agent (teacher) [54] or even learning teachers [33, 45].

Inspired by the advantages of centralized learning and communication for synchronization we propose multi-level coordination among intelligent agents via messages learned by a separate agent to ease the localized learning of task-related policies. We propose a single *central agent* designed to learn high-level messages based on complete knowledge of all the local agents in the environment. These messages are communicated to the *independent learners* who are free to use or discard them while learning local policies to achieve a set of shared goals. By introducing centralization in this manner, the supplemental agent can play the role of a *facilitator* of learning for the rest of the team. Furthermore, the independent learners need not be as powerful as required if they must train to communicate or model other agents alongside learning task-specific policies.

A hierarchical approach to MARL is not new — we will contrast with other existing methods in Section 2. However, the main insight of our algorithm is to learn to communicate relevant pieces of information from a global perspective to help agents with limited capabilities improve their performance. Potential applications

include autonomous warehouse management [7] and traffic light control [26], where there can be a centralized monitor. After we introduce our algorithm, HAMMER, we will show results in two very different simulated environments to showcase its versatility. OpenAI's multi-agent cooperative navigation lets agents learn in a continuous state space with discrete actions and global team rewards. In contrast, Stanford's multi-agent walker environment has a continuous action space and agents can receive only local rewards.

The main contributions of this paper are to explain a novel and important setting that combines agents with different abilities and knowledge (Section 3), introduce the HAMMER algorithm that addresses this setting (Section 4), and then empirically demonstrate that HAMMER can make significant improvements in learning decision policies for agents (Section 6) in two multi-agent domains.

2 BACKGROUND AND RELATED WORK

This section will provide a summary of background concepts necessary to understand the paper and a selection of related work.

2.1 Proximal Policy Optimization

A popular choice for solving RL tasks is to use policy gradients, where the parameters of the policy θ are directly updated to maximize an objective $\mathbf{J}(\theta)$ by moving in the direction of $\nabla\mathbf{J}(\theta)$. In our work, we make use of Proximal Policy Optimization (PPO) [41], which reduces challenges like exhibiting high variance gradient estimates, being sensitive to the selection of step-size, progressing slowly, or encountering catastrophic drops in performance. Moreover, it is relatively easy to implement. Its objective function, well-suited for updates using stochastic gradient descent, can be defined as follows:

$$L^{CLIP}(\theta) = \mathbf{E}_t\left[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)\right],$$

where r_t is the ratio of probability under the new and old policies respectively, A_t is the estimated advantage at time t, and ϵ is a hyperparameter.

2.2 Multi-Agent Reinforcement Learning

Single-agent reinforcement learning can be generalized to competitive, cooperative, or mixed multi-agent settings. We focus on the fully cooperative multi-agent setting, which can described as a generalization of MDPs to stochastic games (SGs). A SG for n local agents, and an additional centralized agent in our case, can be defined as the tuple $\langle S, U, O_1, \ldots, O_n, A_1, \ldots, A_n, P, R, \gamma \rangle$, where S is the set of all configurations of environment states for all agents, U is the set of all actions for the central agent, $O_1, \ldots, O_n$ represent the observations of each local agent, $A_1, \ldots, A_n$ correspond to the set of actions available to each local agent and P is the state transition function. γ is the discount factor. In case all the agents have the same common goal (i.e., they aim to maximize the same expected sum) the SG becomes fully cooperative.

The state transitions in a multi-agent case are a result of the joint action of all the agents $\mathrm{U} \times A_1 \times \cdots \times A_n$. The policies join together to form a joint policy h: $\mathrm{S} \times \mathrm{U} \times \mathbf{A}$. There can be two possible reward structures for the agents. First, they could share a common team reward signal, $R : S \times A \rightarrow \mathfrak{R}$, defined as a function of the state s ∈ S and the agents joint action A: $A_1 \times ... \times A_n$. In the case of such shared rewards, agents aim to directly maximize the returns for the team. Second, each agent could receive its own reward $R_i : O_i \times A_i \rightarrow \mathfrak{R}$. A localized reward structure means that agents maximize their own individual expected discounted return $\mathrm{r} = \sum_{t=0}^{\infty} \gamma^t r^t$.

2.3 Relevant Prior Work

Recent works in both SARL and MARL have employed deep learning methods to tackle the high dimensionality of the observation and action spaces [10, 25, 30, 36, 51].

Several works in the past have taken advantage of hierarchical approaches to MARL. The MAXQ algorithm was designed to provide for a hierarchical break-down of the reinforcement learning problem by decomposing the value function for the main problem into a set of value functions for the sub-problems [6]. Tang et al. [53] used temporal abstraction to let agents learn high-level coordination and independent skills at different temporal scales together. Kumar et al. [18] present another framework benefiting from temporal abstractions to achieve coordination among agents with reduced communication complexities. These works show positive results from combining centralized and decentralized approaches in different manners and are therefore closely related to our work. Vezhnevets et al. [57] introduce Feudal networks in hierarchical reinforcement learning and employ a Manager-Worker framework. However, there are some key differences. In their case, the manager directly interacts with the environment to receive the team reward and accordingly distributes it among the workers (analogous to setting their goals), whereas in our work the central agent interacts indirectly and receives the same reward as the local agents. Here, the central agent is only allowed to influence the actions of independent learners rather than set their goals explicitly. One further distinction is that they partly pretrain their workers before introducing the manager into the scene. In contrast, our results show that when the central agent and the independent learners simultaneously learn, they achieve better performance.

Some works have developed architectures that use centralized learning but ensure decentralized execution. COMA [10] used a centralized critic to estimate the Q-function along with a counterfactual advantage for the decentralized actors in MARL. The VDN [49] architecture trained individual agents by learning to decompose the team value functions into agent-wise value functions in a centralized manner. QMIX [38] employs a mixing network to factor the joint action-value into a monotonic non-linear combination of individual value functions for each agent. Another work, MADDPG [27], extends deep deterministic policy gradients (DDPG) [25] to MARL. They learn a centralized critic for each agent and continuous policies for the actors and allow explicit communication among agents. Even though the prior research works mentioned here address a similar setting and allow for using extra globally accessible information like in our work, they mainly aim at decentralized execution which applies to domains where global view is unavailable. In contrast, we target domains where a global view is accessible to a single agent, even during execution.

There has been considerable progress in learning by communication in cooperative settings involving partially observable environments. Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL) [8] use neural networks to output communication messages in addition to the agent's Q-values. RIAL used a shared network to learn a single policy whereas DIAL used gradient sharing during learning and communication actions during execution. Both methods use discrete communication channels. On the other hand, CommNet [48], used continuous vectors, enabled multiple communication cycles per time step and the agents were allowed to freely enter and exit the environment. Lazaridou et al. [21] and Mordatch et al. [32] trained the agents to develop an emergent language for communication. Furthermore, standard techniques used in deep learning, such as dropout [47] have inspired works where messages of other agents are dropped out during learning to work well even in conditions with only limited communication feasible [17]. However, in all these works, the goal is to learn inter-agent communication alongside local policies that suffer from the bottleneck of simultaneously achieving effective communication and global collaboration [43]. They also face difficulty in extracting essential and high-quality information for exchange among agents [43]. Further, unlike HAMMER, these works expect that more sophisticated agents are available in the environment in terms of communication capabilities or the ability to run complex algorithms to model other agents present — which might not always be feasible.

One of the popular ways to carry out independent learning is by emergent behaviours [23, 24, 51], where each agent learns its own private policy and assumes all other agents to be a part of the environment. This method disregards the underlying assumptions of single-agent reinforcement learning, particularly the Markov property. Although this may achieve good results [29], it can also fail due to non-stationarity [20, 56]. Self-play can be a useful concept in such cases [2, 46, 55], but it is still susceptible to failures through the loss of past knowledge [19, 22, 39]. Gupta et al. [13] extend three SARL algorithms, Deep Q Network (DQN) [31], Deep Deterministic Policy Gradient (DDPG) [25], and Trust Region Policy Optimization (TRPO) [40], to cooperative multi-agent settings.

3 SETTING

This section details a novel setting in which multiple agents — the central agent and the local agents — with different capabilities and knowledge are combined (see Figure 1). The HAMMER algorithm is designed for this type of cooperative multi-agent environment.

Consider a warehouse setting where lots of small, simple, robots fetch and stock items on shelves, as well as bring them to packing stations. If the local agents could communicate among themselves, they could run a distributed reasoning algorithm, but this would require more sophisticated robots and algorithms. Or, if the observations and actions could be centralized, one very powerful agent could determine the joint action of all the agents, but this would not scale well and would require a very powerful agent (to address an exponential growth in the observation and actions spaces with the number of agents). Section 6 will make this more concrete with two multi-agent tasks. Now, assume there is an additional central agent in the team, that has a global perspective, unlike the local agents, who receive only local observations. Further, the central agent is more powerful — not only does it have access to more information, but it can also communicate messages to all local agents. The local agents can only blindly transmit their observations and actions to the central agent and receive messages in return. in this manner, the local agents can simply rely on the communicated messages to get briefed about the other agents in the environment and make informed decisions (actions) accordingly. Consequently, the central agent must learn to encapsulate the available information in its, possibly large, inputs into smaller vectors (messages) to facilitate the learning of the local agents.

Having described an overview of the setting, we can take a closer look at the inputs, outputs, and roles of the agents in the heterogeneous system as described in Figure 1. The centralized agent receives a global observation $s \in S$ on every time step and outputs a unique message (equivalently, executes an action), $u_i \in U$, to each of the local agents, where i is the agent identifier. Its global observation s is the union of all the local observations $o_i \in O_i$ and actions of the independent learners $a_i \in A_i$ — can either be obtained from the environment or transmitted to it by the local agents at every time step. u_i encodes a message vector that a local agent can use to make better decisions. Local agents receive a partial observation o_i, and a private message u_i from the central agent. Based on o_i and u_i, at each time step, all n local agents will choose their actions simultaneously, forming a joint action $(A_1 \times \cdots \times A_n)$ and causing a change in the environment state according to the state transition function P.

Upon changing the dynamics of the environment, a reward $r \in R$ — which could be team-based or localized — is sent back to the local agents, using which they must learn how to act. If the messages from the central agent were not useful, local agents could learn to ignore them. Every time the central agent communicates a message u_i to a local agent, it learns from the same reward as is obtained by that local agent on performing an action a_i in the environment. In other words, the central agent does not directly interact with the environment to get feedback, instead, it learns to output useful messages by looking at how the independent agents performed in the environment using the messages it communicated. In domains with localized rewards for agents, the central agent gets a tangible feedback for sent messages, whereas, in the case of team rewards, it needs to learn using comparatively abstract feedback. In Section 6 we show that HAMMER generalizes to both the reward structures.

4 THE HAMMER ALGORITHM

This section introduces HAMMER, the *Heterogeneous Agents Mastering Messaging to Enhance Reinforcement learning* algorithm, designed for the cooperative MARL setting discussed above.

There are multiple local and independent agents in an environment that are given tasks. Depending on the domain, they may take discrete or continuous actions to interact with the environment. In HAMMER, we introduce a single, relatively powerful central agent into the environment, capable of 1) obtaining a global view of all the other agents present in an environment, and 2) transmitting messages to them. It learns a separate policy and aims to support the local team by communicating short messages to them. It is designed to use both a global or local reward structure. As a result,

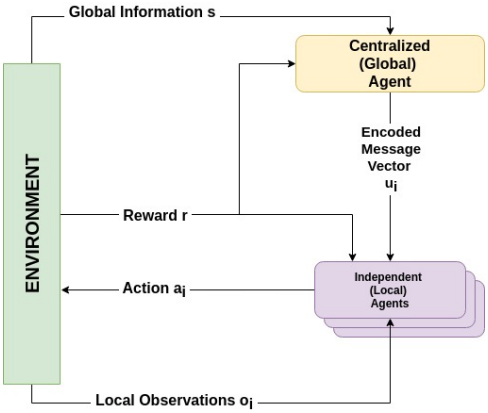

Figure 1: Our cooperative MARL setting: a single global agent sends messages to help multiple independent local agents act in an environment

local agents' private observations will now have additional messages sent to them by the central agent and they can choose to use or discard this information while learning their policies.

As described by Algorithm 1, in every iteration, the centralized agent receives the union of private observations of all local agents (line 3). It encodes its input and outputs an individual message vector for each agent (line 7). These messages from the central agent are sent to the independent learners, augmenting their private partial observations obtained from the environment (line 9). Then, they output an action (line 10) affecting the environment (line 12). Reward for the joint action performed in the environment is returned (line 12) and is utilized as feedback by all the agents to adjust their parameters and learn private policies (lines 15—17).

There are multiple techniques for training the central agent to learn how to communicate. One strategy could be to employ any RL algorithm to train HAMMER's central agent to learn its policy, and use the local agents' reward as a gradient signal. A second strategy would be to push gradients from the local agent's network to HAMMER's central agent network, by directly connecting the latter's outputted communication actions to the input of the local agent's network. This strategy is inspired from a number of other relevant works [9, 48]. Another strategy could be to allow messages m_i output by the central agent to first be processed by a regularisation unit, like RU(m_i) = Logistic(**N** (m_i, σ)), where σ is the standard deviation of the noise added to the channel, and then be passed to the local agent's network [5, 9, 14]. In Section 6, we compare our results for all of these strategies.

The independent learners follow the formerly proposed idea of allowing all of them to share the parameters of a single policy, hence enabling a single policy to be learned with experiences from all the agents simultaneously [9, 13]. This still ensures different behavior among agents because each of them receive different observations. Parameter sharing has been successfully applied in several other multi-agent deep reinforcement learning settings [11, 36, 38, 48, 50]. Learning this way makes it centralized, however, execution can be decentralized by making a copy of the policy and using them individually in each of the agents. Furthermore, in cooperative multi-agent settings, concurrent learning often does not scale up to

Algorithm 1: HAMMER

1 **Initialize** Actor-Critic Network for independent agents (shared network) (IA), Actor-Critic Network for the central agent (CA) or a multi-layered perceptron if gradients from IA are backpropagated to CA, and two experience replay memory buffers (B and B');
2 **for** *episode e = 1 to TOTAL_EPISODES* **do**
3 Fetch combined initial random observations $s = [o_1, ..., o_i]$ from environment (o_i is agent i's local observation);
4 *Input:* Concat ($s = [o_1, ..., o_i]$) $\rightarrow$ CA ;
5 **for** *time step t = 1 to TOTAL_STEPS* **do**
6 **for** *each agent* n_i **do**
7 *Output*: message vector $u_i \leftarrow$ CA, for agent n_i;
8 Pass each message vector u_i through a regularization unit ;
9 *Input:* Concat ($o_i \in s, u_i$) $\rightarrow$ IA ;
10 *Output*: local action $a_i \leftarrow$ IA ;
11 **end**
12 Perform sampled actions in environment and get next set of observations s' and rewards r_i for each agent ;
13 Add experiences in B and B' for CA and IA respectively;
14 **if** *update interval reached* **then**
15 Sample random minibatch b $\in$ B and b' $\in$ B';
16 Train IA on b' using stochastic policy gradients ;
17 Train CA on b or directly by backpropagating gradients from IA ;
18 **end**
19 **end**
20 **end**

the number of agents which can make the environment dynamics non-stationary, and difficult to learn [13]. Hence, in HAMMER, a single policy is learned for the local agents via the centralized learning, decentralized execution setting.

Note that while we focus on having a shared network for independent agents and using PPO methods for policy learning, HAMMER can be implemented with other MADRL algorithms. Moreover, even though we test with a single central agent in this paper, it is entirely possible that multiple central agents could better assist independent learners. This is left to future works.

5 TASKS AND IMPLEMENTATION DETAILS

This section details the two multi-agent environments used to evaluate HAMMER. In addition to releasing our code after acceptance, we fully detail our approach so that results are replicable.

5.1 Cooperative Navigation

Cooperative navigation is one of the Multi-Agent Particle Environments [27]. It is a two-dimensional cooperative multi-agent task with a continuous observation space and a discrete action space consisting of n movable agents and n fixed landmarks. Figure 2

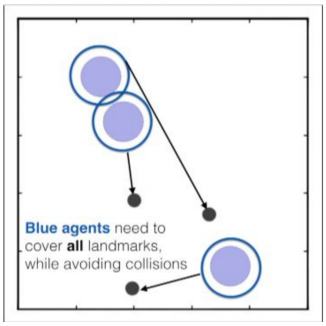

Figure 2: Cooperative Navigation

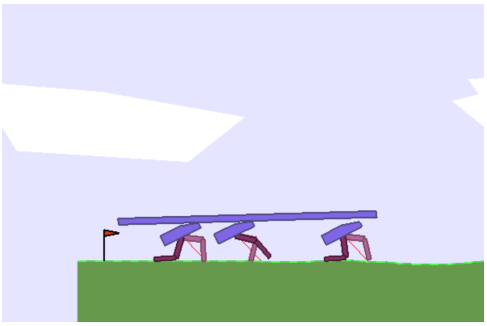

Figure 3: Multi-Agent Walker

shows the case where $n = 3$. Agents occupy physical space (i.e., are not point masses), perform physical actions in the environment, and have the same action and observation spaces. The agents must learn to cover all the landmarks while avoiding collisions, without explicit communication. The global reward signal, seen by all agents, is based on the proximity of any agent to each landmark and the agents are penalized upon colliding with each other. The team reward can be defined by the equation:

$$R = [-\sum_{n=1,l=1}^{N,L} min(dist(a_n, l))] - c,$$

where N is the number of agents and L is the number of landmarks in the environment. The function *dist()* calculates the distance in terms of the agents' and landmarks' (x_i, y_i) positions in the environment. The number of collisions, c, among the agents and is set as a penalty of -1 for each time two agents collide. The action set is discrete, corresponding to moving in the four cardinal directions or remaining motionless. Each agent's observation includes the relative positions of other agents and landmarks within the frame. Note that the local observations do not convey the velocity (movement direction) of other agents. Consistent with past work, the initial positions of all the agents and the landmark locations are randomized at the start of each episode, and each episode ends after 25 time steps.

To test HAMMER, we modify this task so that a centralized agent receives the union of local agents' observations at every time step. We also conduct relevant ablation studies on modified versions of this environment (exhibiting graceful degradation to the system, particularly to what the local agents can observe) to understand the contribution of HAMMER's central agent to the overall system. We are most interested in this environment because of the motivating robotic warehouse example.

5.2 Multi-Agent Walker

Multi-agent walker is a more complex, continuous control benchmark locomotion task [13]. A package is placed on top of n pairs of robot legs which can be controlled. The agents must learn to move the package as far to the right as possible, without dropping it. The package is large enough (it stretches across all of the walkers) that the agents must cooperate. The environment demands high inter-agent coordination for them to successfully navigate a complex terrain while keeping the package balanced. This environment supports both team and individual rewards, but we focus on the latter, to demonstrate the effectiveness of HAMMER in individual feedback settings (as team rewards were already explored in the cooperative navigation task). Each walker is given a reward of -100 if it falls and all walkers receive a reward of -100 if the package falls. Throughout the episode, each walker is given an additional reward of +1 for every step taken. However, there is no supplemental reward for reaching the destination. By default, the episode ends if any walker falls, the package falls, after 500 time steps, or if the package successfully reaches the destination. Each agent receives 32 real-valued numbers, representing information about noisy Lidar measurements of the terrain and displacement information related to the neighbouring agents. The action space is a 4-dimensional continuous vector, representing the torques in the walker's two joints of both legs. Figure 3 is an illustration of the environment with $n = 3$ agents.

Similar to the Cooperative Navigation domain, this environment is tweaked so that all local agents in it can transmit their private observations to the central agent. This environment was chosen primarily to test our approach in a continuous action domain and in cases where individual agents receive their own local rewards instead of global team rewards. It is also significantly more challenging than the cooperative navigation domain.

5.3 Implementation Details

In both domains, two networks were used — one for the central agent and the second for the independent agents. PPO was used to update the decision-making policy for the local agents. For the actor and critic networks of these individual agents, 2 fully-connected multi-layer perceptron layers were used to process the input layer and to produce the output from the hidden state. Three different variants were use to train the central agent. The first variant, HAMMERV1, had its own actor-critic network and learned its policy to communicate using PPO. In this case, the central agent's rewards were the same as those received by the local agents from the environment. In the second variant variant, HAMMERV2, the central agent used a multi-layer perceptron to output real-valued messages for the local agents and learned directly via the gradients passed back to it from the local agent network using backpropagation. In the final variant, HAMMERV3, instead of directly passing the messages to the local agent, we pre-processed the messages using a regularisation unit, RU (as described earlier in Section 4). The

central agent's network is capable of producing a vector of floating points scaled to $[-1, 1]$ and is consistent across domains. A tanh activation function is used everywhere except for the local agent's output. Local agents in cooperative navigation use a soft-max over the output, corresponding to the probability of executing each of the five discrete actions. Local agents in the multi-agent walker task use 4-output nodes, each of which model a multivariate Gaussian distribution over torques.

Trials were run for 500,000 episodes in the cooperative navigation and 35,000 in the multi-agent walker environments. The training times of baselines and HAMMER were similar — 12 hours and 14 hours, respectively, for 500,000 episodes on cooperative navigation and 14 hours and 18 hours, respectively, for 35,000 episodes in multi-agent walker. We set $\gamma = 0.95$ for all experiments. After having tried multiple learning rates $\{0.01, 0.001, 0.002, 0.005, 3 \times 10^{-4}, 1 \times 10^{-2}, 3 \times 10^{-3}\}$ and training batch sizes in PPO — we found 3×10^{-3} and 800, respectively, to work best for both the centralized agent and the independent agents in both domains. Five independent trials with different random seeds were performed on both environments to establish statistical significance. The clip parameter for PPO was set to 0.2. Empirically, we found that a message length of 1 was enough to perform well in both the navigation and multi-agent walker tasks.

6 RESULTS

This section describes the experiments conducted to test HAMMER's potential of encapsulating and communicating learned messages and speeding up independent learning on the two environments — cooperative navigation and multi-agent walker — whose details are described in the previous section. All the curves are averaged over five independent trials. Additional ablative studies were performed on the modified versions of cooperative navigation environment.

6.1 Cooperative Navigation Results

We investigated in detail the learning of independent learners in the cooperative navigation environment under different situations. Learning curves used for evaluation are plots of the average reward per local agent as a function of episodes. Moreover, to smooth the curves for readability, a moving average with a window size of 1500 episodes was used for each of the cases.

At first, we let the local agents learn independently, without any aid from other sources. The corresponding curve (red), as shown in Figure 4, can also act as our baseline to evaluate learning curves obtained when the learners are equipped with additional messages. As described earlier, the experimental setup is consistent with earlier work [13].

Next, we used a central agent to learn messages, as described by HAMMER, and communicated them to the local learners to see if the learning improved. As described earlier (Section 4) HAMMER's central agent is trained using more than one strategies and we compare the results here. First, the central agent learned to communicate by employing PPO, and used the local agents' team-based reward as its feedback (HAMMERV1). HAMMERV1 performed only slightly better than independent learners alone, over a training period of 500,000 episodes, as can be seen in Figure 4 (purple). This training strategy does provide a small amount of improvement, which could be because of two types of problems: *(Case 1)* The central agent may emit a relevant message to a local agent, but the local agent is unable to intercept it correctly resulting in a poor reward. In such a case, the central agent gets penalized, even though it performed its part well. *(Case 2)* The central agent emits relevant messages to some local agents, and non-relevant messages to others. In such a case, the central agent is penalized in spite of emitting relevant messages for some agents.

Now, to avoid the problems encountered in the previous case, gradients from the local agent's network were pushed to HAMMER's central agent network, by directly connecting the outputted communication vectors to the input of the local agent's network (HAMMERV2). Letting gradients flow in this manner, gives the central agent richer feedback, thus reducing the required amount of learning by trial and error, and easing the discovery of effective real-valued messages. Since these messages function as any other network activation, gradients can be passed back to the central agent's network, allowing end-to-end backpropagation across the entire framework. As shown in Figure 4, HAMMERV2 (green) performs significantly better than unaided independent learners as well as HAMMERV1. However, HAMMER agents seem to slow after 250,000 episodes. We speculate that a larger action space (real-valued messages) causes the central agent learns to "over-encode" the global information available to it. Having included this extra information along with the relevant pieces would have masked the private observations of the local agents, slowing their progress.

Finally, addressing the problem of "over-encoding" faced in the previous case, the emitted messages were first processed by a regularisation unit — $\text{RU}(m_i) = \text{Logistic}(\mathbf{N}(m_i, \sigma))$, to encourage discretization of the messages (by pushing the activations of the communication vectors into two different modes during training, i.e., where the noise is minimized), and then passed to the local agent's network (HAMMERV3). Figure 4 illustrates that HAMMERV3 (blue) outperforms all the other cases. Hence, using a regularization unit to limit the central agent's action space was essential for learning a better communication policy. Using a noisy channel in this manner has been useful in other relevant works too — differentiable inter-agent learning [9], training document models [14] and performing classification [5]. For our experiments, we used a value of $\sigma = 0.2$.

From these experiments, our results show that: (1) HAMMER agents were able to learn much faster when compared to independent local agents, and (2) the central agent was able to successfully learn to produce useful smaller messages. Recall that the total global observation vector has $18 \times 3 = 54$ real-valued numbers (for 3 agents and 3 landmarks in the environment), and our message uses only 1.

To help evaluate the communication quality, random messages of the same length were generated and communicated to the local agents to see if the central agent was indeed learning relevant or useful messages. As expected, random messages induce a larger observation space with meaningless values and degrade the performance of independent learners (Figure 4, orange curve). This also supports the claim that the central agent is learning much better messages to communicate (rather than sending random values) as it outperforms the independent learning of agents provided with random messages. Note here that the local agents are free to learn to ignore unhelpful messages and independent agents' learning while receiving random messages only degrades performance slightly.

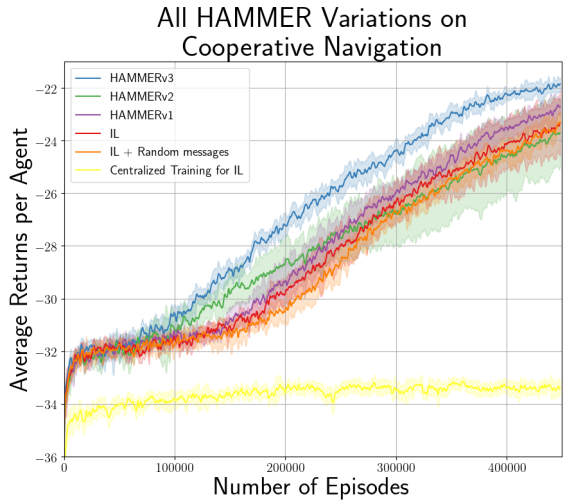

Figure 4: HAMMER agents outperform independent PPO learners in cooperative navigation. Using a similar framework as HAMMER, but providing the local agents with random messages, causes degraded performance (as expected). HAMMER also significantly outperforms centrally learned policy for independent agents.

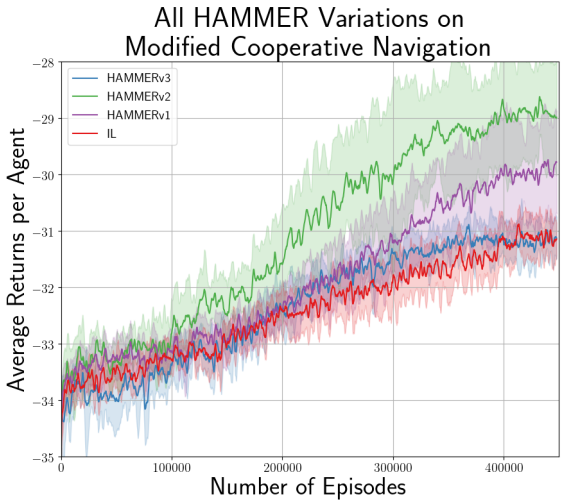

Figure 5: Results on a modified cooperative navigation environment — disallowing the local agents to be able to observe each other, hence necessitating the need for communication via the central agent to coordinate and cover the landmarks in the environment cooperatively. HAMMER's ability to perform in this setting shows that the central agent is indeed learning effective communication to help the local agents coordinate.

To confirm that the central agent is not simply forwarding a compressed version of the global information vector to all the agents, we let the independent agents learn in a fully centralized manner, using a joint observation space. The performance drops drastically in this case (Figure 4, yellow curve). This suggests that the central agent in HAMMER is learning to encapsulate only partial but relevant information as messages and communicates them to facilitate the learning of local agents. Complete information would have become too overwhelming for the localized agents to learn how to act and hence slowed the progress, as is clear by the curve shown in Figure 4. It has been shown previously that this approach does not perform well in domains such as pursuit, water world and multi-agent walker [13]. We confirm the same in the cooperative navigation environment for n=3 agents.

Ablation Studies. First, we perform an ablation experiment to validate that HAMMER's central agent is indeed facilitating independent agents' learning and helping them coordinate by sending out relevant messages. For this, we modify the environment preventing agents from seeing each other. Local agents be unable to observe other agents and would have to rely on the central agent for coordinating and covering respective landmarks. Results of HAMMER in this scenario, compared to unaided independent learners, are in Figure 5. As expected, it became difficult for the independent learners to learn the cooperation-intensive task on their own. On the other hand, HAMMER agents learned substantially better policies faster. Note here that HAMMERv2 in this case performs better than HAMMERv3. We speculate that this is due to local agents' greater dependency on central agent's communicated messages for coordinating, and hence, HAMMER requiring a larger spectrum to encode and communicate more information into its messages.

In our second ablation study, we validate whether HAMMER would generalize to a setting with heterogeneous local agents. This experiment uses an environment where one of the local agents was unable to observe other agents, while the other two agents still received their original observations. All local agents still learn and use the same policy, but one of them cannot see the others. As expected, independent learners had increased difficulty in learning to coordinate (Figure 6), and were outperformed by all variants of learning strategies for HAMMER. We conclude that HAMMER generalizes to heterogeneous settings too.

In summary, HAMMER successfully summarizes relevant global knowledge into small real-valued messages to individual learners that outperforms other cases — (1) unaided independent learning, (2) independent agents supplied with random messages, and (3) fully centralized training of independent agents. Specifically, HAMMERv3 (i.e., HAMMER trained by directly passing message gradients from the local agent's network to the central agent using backpropagation, and its messages preprocessed using a regularization unit) outperforms the other training strategies, learning a significantly better policy than local agents learning independently.

6.2 Multi-Agent Walker Results

This section shows that HAMMER also works in a multi-agent task with continuous control and individual rewards. Figure 7 shows that HAMMERv3 agents perform considerably better than unaided independent local agents. Like in the cooperative navigation task, results here are averaged over five independent trials, with an additional 5000-episode moving window to increase readability. HAMMER performing better in a domain like Multi-Agent Walker

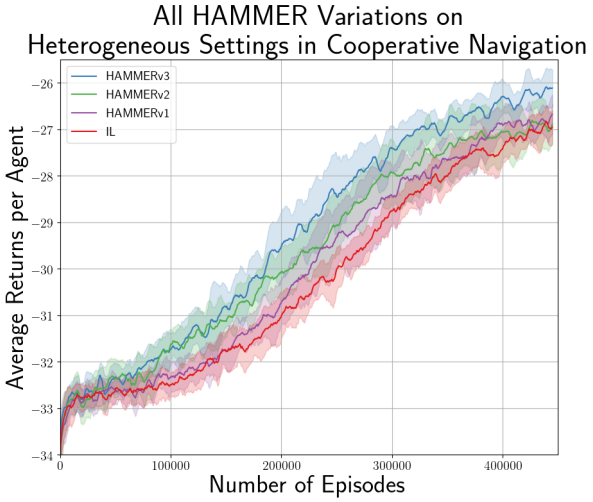

Figure 6: HAMMER applied on another modified version of the cooperative navigation environment such that the local agents in the environment were heterogeneous in nature — one of the local agents was unable to observe the other agents, whereas the other two agents could. Results show that HAMMER was able to generalize to these settings too.

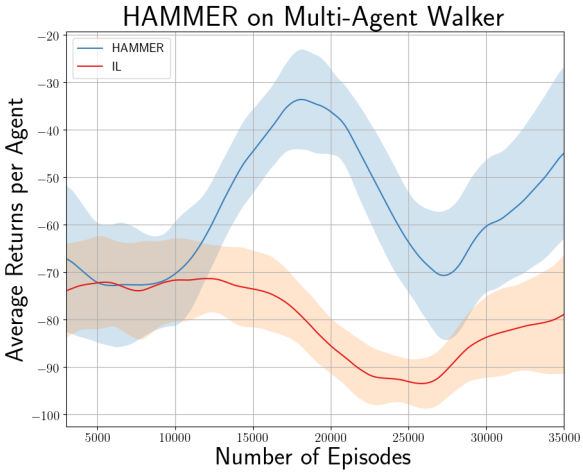

Figure 7: HAMMER considerably improves the performance over independent learners in the multi-agent walker task too.

confirms the generalization of the approach to continuous action spaces and different reward structures.*

The results for the two test domains show that (1) heterogeneous agents successfully learn messaging and enable multi-level coordination among the independent agents to enhance reinforcement learning using HAMMER approach, (2) HAMMER generalizes to heterogeneous local agents in the environment, (3) the approach works well in both discrete and continuous action spaces, and (4) the approach performed well with both individual rewards and global team rewards.

7 CONCLUSION

This paper presented HAMMER, an approach used to achieve multi-level coordination among heterogeneous agents by learning useful messages from a global view of the environment or the multi-agent system. Using HAMMER, we addressed challenges like non-stationarity and inefficient coordination among agents in MARL by introducing a powerful all-seeing central agent in addition to the independent learners in the environment. Our results in two domains, cooperative navigation and multi-agent walker, showed that HAMMER can be generalized to discrete and continuous action spaces with both global team rewards and localized personal rewards. HAMMER also proved to generalize to heterogeneous local agents in the environment We believe that the key reasons for the success of HAMMER were two-fold. First, we leveraged additional global information like global states, actions and rewards in the system. Second, centralizing observations has its own benefits, including helping to bypass problems like non-stationarity in multi-agent systems and avoiding getting stuck in local optima [16, 58]. Several works, related to ours (discussed earlier) demand powerful agents in the environment — ones that can transmit to other agents and/or are capable of modeling other agents in the system. This might not always be feasible, thus motivating HAMMER, where only one central agent needs to be powerful while the independent learners can be simple agents interacting with the environment based on their private local observations augmented with learned messages. Warehouse management and traffic lights management are two example applications that could fit these assumptions well.

There are several directions for future work from here. Both the domains used in this work involved low-dimensional observation spaces and seem to offer substantial global coverage on combining local observations. HAMMER's results in multi-agent settings with tighter coupling and further complex interactions involved among agents such as in autonomous driving in SMARTS [62] or heterogeneous multi-agent battles in StarCraft [36] could help better appreciate the significance of the method. Additionally, more complex hierarchies could be used, such as by making several central agents available in the system. In this work, we performed an initial analysis of message vectors communicated by HAMMER, but additional work remains to better understand if and how HAMMER tailors messages for the local agents using its global observation. It would also be interesting to further study how RL learns to encode information in messages and to understand what the encoding means. It would also be interesting to test the scalability of HAMMER to a larger number of local agents in the environment. Lastly, in our setting, communication is free — future work could consider the case where it was costly and attempt to trade off the number of messages sent with the learning speed of HAMMER. If the central agent had the ability to communicate small amounts of data occasionally, would it still be able to provide a significant improvement?

*We did not include the results corresponding to the other baselines here as Gupta et al. [13] already show that independent learners outperform a centralized policy in the multi-agent walker domain.

REFERENCES

[1] Stefano V Albrecht and Peter Stone. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence* 258 (2018), 66–95.

[2] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. 2015. Heads-up limit hold'em poker is solved. *Science* 347, 6218 (2015), 145–149.

[3] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[4] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2012. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9, 1 (2012), 427–438.

[5] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).

[6] Thomas G Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of artificial intelligence research* 13 (2000), 227–303.

[7] John J Enright and Peter R Wurman. 2011. Optimization and coordinated autonomy in mobile fulfillment systems. In *Workshops at the twenty-fifth AAAI conference on artificial intelligence*. Citeseer.

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

[9] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2137–2145.

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

[11] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1702.08887* (2017).

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

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

[14] Geoffrey Hinton and Ruslan Salakhutdinov. 2011. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science* 3, 1 (2011), 74–91.

[15] Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, Tokuro Matsuo, and Hirofumi Yamaki. 2010. *Innovations in agent-based complex automated negotiations*. Vol. 319. Springer.

[16] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. 2011. Accelerating best response calculation in large extensive games. In *IJCAI*, Vol. 11. 258–265.

[17] Woojun Kim, Myungsik Cho, and Youngchul Sung. 2019. Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6079–6086.

[18] Saurabh Kumar, Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. 2017. Federated control with hierarchical multi-agent deep reinforcement learning. *arXiv preprint arXiv:1712.08266* (2017).

[19] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in neural information processing systems*. 4190–4203.

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

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

[22] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. 2019. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742* (2019).

[23] Joel Z Leibo, Julien Perolat, Edward Hughes, Steven Wheelwright, Adam H Marblestone, Edgar Duéñez-Guzmán, Peter Sunehag, Iain Dunning, and Thore Graepel. 2018. Malthusian reinforcement learning. *arXiv preprint arXiv:1812.07019* (2018).

[24] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037* (2017).

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

[26] Mengqi Liu, Jiachuan Deng, Ming Xu, Xianbo Zhang, and Wei Wang. 2017. Cooperative deep reinforcement learning for tra ic signal control. In *The 7th International Workshop on Urban Computing (UrbComp 2018)*.

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

[28] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib. 2012. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *AAAI 2012*. p2017–2023.

[29] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. 2012. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. (2012).

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

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[32] Igor Mordatch and Pieter Abbeel. 2018. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[33] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P How. 2019. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6128–6136.

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

[35] James Parker, Ernesto Nunes, Julio Godoy, and Maria Gini. 2016. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics* 33, 7 (2016), 877–900.

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

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

[38] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485* (2018).

[39] Spyridon Samothrakis, Simon Lucas, ThomasPhilip Runarsson, and David Robles. 2012. Coevolving game-playing agents: Measuring performance and intransitivities. *IEEE Transactions on Evolutionary Computation* 17, 2 (2012), 213–226.

[40] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. 1889–1897.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[42] Sven Seuken and Shlomo Zilberstein. 2012. Improved memory-bounded dynamic programming for decentralized POMDPs. *arXiv preprint arXiv:1206.5295* (2012).

[43] Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. 2020. Learning Structured Communication for Multi-agent Reinforcement Learning. *arXiv preprint arXiv:2002.04235* (2020).

[44] Yoav Shoham, Rob Powers, and Trond Grenager. 2007. If multi-agent learning is the answer, what is the question? *Artificial intelligence* 171, 7 (2007), 365–377.

[45] Felipe Leno Da Silva, Garrett Warnell, Anna Helena Reali Costa, and Peter Stone. 2020. Agents teaching agents: a survey on inter-agent transfer learning. *Autonomous Agents and Multi-Agent Systems* (Jan 2020).

[46] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[48] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *Advances in neural information processing systems*. 2244–2252.

[49] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent

learning. *arXiv preprint arXiv:1706.05296* (2017).

[50] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward.. In *AAMAS*. 2085–2087.

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

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

[53] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Changjie Fan, and Li Wang. 2018. Hierarchical deep multiagent reinforcement learning. *arXiv preprint arXiv:1809.09332* (2018).

[54] Matthew E. Taylor, Nicholas Carboni*, Anestis Fachantidis, Ioannis Vlahavas, and Lisa Torrey. 2014. Reinforcement learning agents providing advice in complex video games. *Connection Science* 26, 1 (2014), 45–63. https://doi.org/10.1080/09540091.2014.885279 arXiv:http://dx.doi.org/10.1080/09540091.2014.885279

[55] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (1995), 58–68.

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

[57] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161* (2017).

[58] Shimon Whiteson, Brian Tanner, Matthew E Taylor, and Peter Stone. 2011. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*. IEEE, 120–127.

[59] Michael Wunder, Michael Littman, and Matthew Stone. 2009. Communication, credibility and negotiation using a cognitive hierarchy model. In *AAMAS Workshop*, Vol. 19. Citeseer, 73–80.

[60] Wang Ying and Sang Dayong. 2005. Multi-agent framework for third party logistics in E-commerce. *Expert Systems with Applications* 29, 2 (2005), 431–436.

[61] Mohamed Salah Zaïem and Etienne Bennequin. 2019. Learning to Communicate in Multi-Agent Reinforcement Learning: A Review. *arXiv preprint arXiv:1911.05438* (2019).

[62] Ming Zhou, Jun Luo, Julian Villela, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. 2020. SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. *arXiv preprint arXiv:2010.09776* (2020).