Undesirable communication behaviour in cooperative agents with decentralised critics

Antonia Boca
Department of Computer Science
University of Cambridge
aib36@cam.ac.uk

Abstract

Multi-Agent Reinforcement Learning (MARL) systems have proven useful in various applications, such as game theory, traffic control, and supply-chain optimisation. Many works in the field of MARL have been focusing on implementing decentralised multi-agent systems for real-world problems that are generally too complex for centralised models to handle. Given such systems, it is not often straightforward how decentralised agents should communicate in a shared environment. In this project, I add a differentiable communication channel to a cooperative environment and then investigate the communication behaviour of agents with decentralised critics in this environment. Results indicate that introducing communication does not lead to a performance increase, with a post-hoc analysis showing that these agents may learn to lie to each other. This behaviour persists in a low-visibility setting where agents may benefit from exchanging topological information.

The code can be found at: `https://github.com/semiluna/self_interested_comms`.

1 Introduction

Multi-agent reinforcement learning (MARL) is a subfield of reinforcement learning (RL) that deals with the interaction between multiple agents, where each agent learns to make decisions based on its local observations and rewards in a shared environment. In MARL, the agents can be autonomous and have their own learning policies, which can be independent or coordinated. MARL systems can be further split into *centralised* and *decentralised* systems; the latter is particularly useful in complex environments where agents may have different access to information and global state aggregation is not possible, like in traffic control.

Graph neural networks (GNNs) are a type of neural network that can operate on data represented in the form of graphs to perform tasks such as node and graph classification, and link prediction. All GNNs use a form of *neural message passing*, in which vector messages are exchanged between nodes and updated using neural networks [Gilmer et al., 2017]. Due to their powerful inductive bias towards handling relational data, they have become a popular choice to model communication between agents in a shared-environment [Li et al., 2019, Khan et al., 2020, Tolstaya et al., 2020]. Recent works have been focusing on improving the GNN-driven approach to communication, with Kortvelesy and Prorok [2021] proposing a more powerful and scalable GNN architecture, Siedler [2021] introducing the choice to communicate as part of the action space, and Blumenkamp et al. [2022] showing how GNN-based policies can be implemented in physical systems that may deal with network issues.

Some of the works above focus on improving communication in cooperative settings; however, the cooperative setting itself is usually based on the actor-critic framework [Konda and Tsitsiklis, 1999], assuming decentralised actors and a centralised critic. While the practitioner's intuition is that centralised critics ease learning and fight non-stationarity issues, this claim was contested by Lyu

Preprint. Under review.

et al. [2021], who argue both theoretically and empirically that both centralised and decentralised critics have their pros and cons.

Many real-world problems can be more readily modelled as individual agents with local critics that learn to cooperate in a shared environment; network routers from different ISPs and self-driving cars from different companies are two such examples. In such scenarios, it is unrealistic to assume to presence of a centralised critic, but may be beneficial to allow such agents to communicate to improve the overall performance of the system. As such, this project builds on top of the implementation and methods of Blumenkamp and Prorok [2021] to study how communication affects learning in cooperative MARL agents that have *decentralised critics*. This is contrast to the original approach of the paper, where cooperative agents have a centralised critic and share GNN parameters in order to develop communication strategies.

1.1 Contributions

In this work, I analyse the behaviour of a group of heterogeneous agents that receive individual rewards while trying to maximise the cummulative reward of the group. In particular, I run the following experiments:

- In the first setting, the agents have full map visibility. In one version, they have an explicit communication channel via a GNN; in the second, they are not allowed to communicate.
- In the second setting, the agents have limited map visibility. Similarly to above, in one version they can communicate, while in the second they cannot.

Therefore, the scope of this project is to investigate the extent to which proper communication strategies can be achieved when agents are heterogeneous, i.e. do not share any network parameters, but are rewarded as a collective for their actions. Additionally, to further understand the communication behaviours achieved in the above settings, I train an "inverse" CNN to interpret agent messages and discover that communication helps some agents learn how to lie about their state.

2 Theoretical background

2.1 Communication via Graph Neural Networks

Graph neural networks are characterised by the *neural message passing* framework, in which vector messages are exchanged between nodes and then updated using neural networks. At message passing iteration k, a hidden embedding $\mathbf{h}_u^{(k)}$ corresponds to each node $u \in \mathcal{V}$, and an update happens according to information aggregated from u's graph neighbourhood $\mathcal{N}(u)$, as seen in Equation 2.

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)}(\mathbf{h}_u^{(k+1)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k+1)}, \forall v \in \mathcal{N}(u)\})) && (1)\\ &= \text{UPDATE}^{(k)}(\mathbf{h}_u^{(k+1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}) && (2)\end{aligned}$$

Note that in general the AGGREGATE function must be permutation-invariant.

Local AGNNs. In this work, I follow the approach presented in Blumenkamp and Prorok [2021] and consider an *Aggregation Graph Neural Network* (AGNN) that operates over a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ that models communication between agents. Each agent is represented by a node in $\mathcal{V} = \{1, \dots, N\}$ and communication links are represented as the edge set $\mathcal{E} = \mathcal{V} \times \mathcal{V}$. Then, $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$ represents the one-hop neighbourhood of node i. Note that in this work I consider bidirectional communication links, meaning that $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$.

The set of initial messages sent by all agents is denoted $\mathbf{X} \in \mathbb{R}^{N \times F}$. Then, $[\mathbf{X}]_i$ contains local information pertaining to agent i. For $k \in 0, \dots, K$ and layer l, agent i has a set of parameters $\mathbf{H}_{k,i}^{(l)} \in \mathbb{R}^{F \times F'}$ that are used to perform a linear transformation on aggregated neighbour data from k hops away. For a given adjacency matrix $\mathbf{S}$, raising $\mathbf{S}$ to the power k allows us to access the neighbours of node i that are k hops away. We can then define the local AGNN update for layer l on node i as follows:[1]

$$[\mathbf{X}]_i^{(l)} = \sigma\Big(\sum_{k=0}^{K}[\mathbf{S}^k]_i\mathbf{X}^{(l-1)}\mathbf{H}_{k,i}^{(l)}\Big) \text{ with } \mathbf{X}^{(0)} = \mathbf{X} \tag{3}$$

where σ is a non-linearity (in this project, the $ReLU$ function). Note that we work with *heterogeneous agents*, hence each node will have its own set of parameters $\mathbf{H}_{k,i}^{(l)}$. In the specific case when the number of layers L is equal to 1, the only information that agents share is the initial message matrix $\mathbf{X}$.

2.2 Problem formulation

We can formulate the problem setting as a Markov Decision Process (MDP) $\langle \mathcal{V}, \mathcal{S}, \mathcal{A}, P, \{R^i\}_{i\in\mathcal{V}}, \{\mathcal{Z}^i\}_{i\in\mathcal{V}}, Z, \mathcal{T}, T, \gamma\rangle$, where agents belong to the set $\mathcal{V} \equiv \{1, \ldots, N\}$ and the environment has true state $s_t \in \mathcal{S}$. At timestep t, each agent performs action $a_t^i \in \mathcal{A}$, forming joint action $\mathbf{a}_t \in \mathcal{A}^N$. The state transition probability function is $P(s_{t+1}|s_t, \mathbf{a}_t) : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \to [0, 1]$. The agents have partial observability, so agent i draws observations $z_t^i \in \mathcal{Z}^i$ according to its observation function $Z^i(s_t) : \mathcal{S} \to \mathcal{Z}$. Each agent also observes a local reward r_t^i drawn from $R^i(s_t, a_t^i) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and makes use of a communication topology $T(s_t) : \mathcal{S} \to \mathcal{T}$ with $\mathcal{T} \in \mathcal{E}_t$ (i.e. the global topology of the entire graph at timestep t).

Agent Policy. An agent has a local observation z_t^i that it passes through an encoder $f_{\nu_i}(z_t^i) : \mathcal{Z} \to \mathbb{R}^F$ parameterised by ν_i. When communication is turned on, these space representations make up the initial messages $\mathbf{X} = [f_{\nu_1}(z_t^1), \ldots, f_{\nu_N}(z_t^N)]^\top$ that are passed through each agent's local AGNN parameterised by η_i, resulting in the final representations $\mathbf{R} = \mathbf{X}^{(L)}$.[2] Finally, Each agent will have function $h_{\mu_i}([\mathbf{R}]_i) : \mathbb{R}^{F'} \to \Delta^{|A|}$ that outputs a distribution over the set of possible actions. Overall, the local policy of agent i is defined by $\pi_\theta^i(a_t^i|z_t, \mathcal{E}_t)$ with $\theta = \{\nu_1, \ldots, \nu_N, \eta_1 \ldots, \eta_N, \mu_1 \ldots, \eta_N\}$. Note that with explicit communication, the policy of each agent depends on the parameters of all other agents, since messages exchanged between agents depend on each other.

Policy Optimisation. I use Proximal Policy Approximation (PPO) [Schulman et al., 2017] to optimise agents' policies for a cooperative task. Formally, agent i learns a policy $\pi_\theta^i(a_t^i|z_t) : \mathcal{Z} \times \mathcal{A} \to [0, 1]$, where $\theta = \{\nu_1, \ldots, \nu_N, \eta_1 \ldots, \eta_N, \mu_1 \ldots, \eta_N\}$ represents the set of all parameters of all agents; note that this is the case because agents communicate with each other, so one agent's network parameters will influence the policies of other agents. Then, the discounted return is given by $G_t^i = \sum_{j=0}^{\infty} \gamma^l r_{t+j}^i$. The value function of each agent i in state s_t is $V^{\pi^i}(s_t) = \mathbb{E}_t[G_t^i|s_t]$ and the action-value function is given by $Q^{\pi^i}(s_t, a_t^i) = \mathbb{E}_t[G_t^i|s_t, a_t^i]$.

Using these notations, we define the advantage function for each agent i to be $A^{\pi^i}(s_t, a_t^i) = Q^{\pi^i}(s_t, a_t^i) - V^{\pi^i}(s_t)$. This function is estimated using the *Generalised Advantage Estimate* [Schulman et al., 2015] to yield $\hat{A}^{\pi^i}(s_t, a_t^i)$. This estimate is used to compute each agent's objective L_i at timestep $(k + 1)$:

$$L_i = \min\left(\frac{\pi_\theta^i(a_t^i|s_t)}{\pi_{\theta_k}^i(a_t^i|s_t)}\hat{A}^{\pi_{\theta_k}^i}(s_t, a_t^i),\ \ \text{clip}\Big(\frac{\pi_\theta^i(a_t^i|s_t)}{\pi_{\theta_k}^i(a_t^i|s_t)}, 1-\epsilon, 1+\epsilon\Big)\hat{A}^{\pi_{\theta_k}^i}(s_t, a_t^i)\right) \tag{4}$$

I then aggregate each L_i to produce the overall objective for the cooperative task:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{5}$$

[1]Note that $\mathbf{S} = \mathbf{S}^\top$ since the communication graph is bidirectional; hence this notation differs slightly from Blumenkamp and Prorok [2021].

[2]When communication is turned off, however, we set $\mathbf{R} = [f_{\nu_1}(z_t^1), \ldots, f_{\nu_N}(z_t^N)]^\top$.

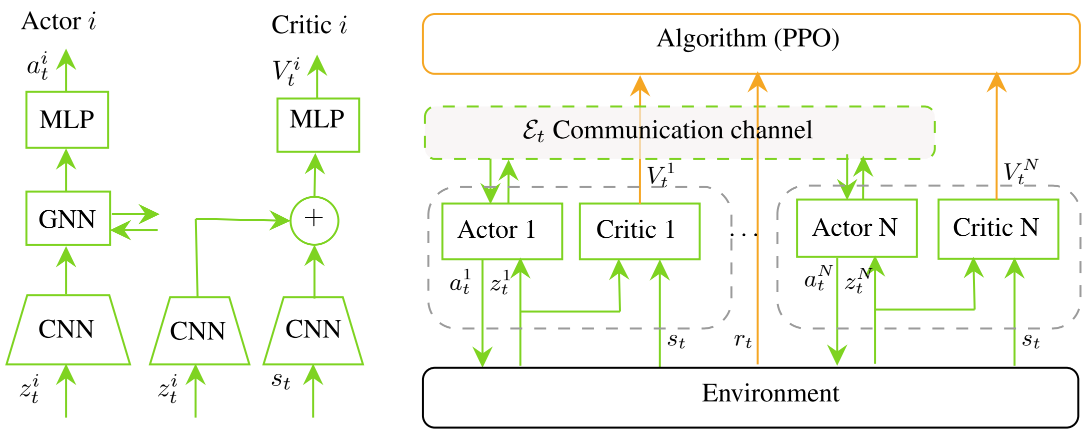


Figure 1: Architectural overview. We have $N = 4$ agents, each being composed of actor and a critic. All green components can be executed locally; each actor has its own local critic that generates value V_t^i to be used in the loss computation. Orange denotes components that are required for centralised training. Agents communicate with each other using the communication channel $\mathcal{E}_t$. The Environment sends reward information r_t to the Algorithm.

3 Implementation

My codebase is heavily inspired from the codebase of Blumenkamp and Prorok [2021].[3] Due to several compatibility and deprecation issues, I decided to re-implement both the models and the environment used in the paper with Ray `2.1.0`. The project uses `Ray[Rllib]` [Liang et al., 2017], as well as Pytorch [Paszke et al., 2019].

3.1 Problem setting

I train 4 agents to cover all free cells of a 16×16 grid map with obstacles. Each agent receives a reward of $+1$ for every new cell it covers (and that hasn't been covered by any other agent so far). The goal of the agents, as a group, is to cover as many cells as possible. While this is a similar setting to Blumenkamp and Prorok [2021], one crucial difference is that each agent has their own neural network to predict an action and their own local critic. Therefore, there is *no parameter sharing* between the agents; the only information they can share is the one they choose to share by exchanging messages that are then aggregated via each agent's local GNN architecture.

Agent architecture. Figure 1 summarises the architecture setup. Using the policy formulation presented in 2.2, the encoder $f_{\nu_i}(z_t^i) : \mathcal{Z} \to \mathbb{R}^F$ is a CNN; the final representations $\mathbf{R}$ are generated using a local 1-layer AGNN presented in 2.1; the function $h_{\mu_i}([\mathbf{R}]_i) : \mathbb{R}^{F'} \to \Delta^{|A|}$ that outputs a distribution over the set of possible actions for each agent i is an MLP.

Environment. As mentioned above, the environment is represented by a 16×16 grid map with obstacles resembling a non-convex area. Obstacles represent approximately 40% of the grid. Figure 2 shows some examples of maps covered by agents during evaluation. The observation of each agent is described by a three-channel tensor of size $16 \times 16 \times 3$ consisting of the local obstacle map, the agent's own coverage, and other agents' position within a specified circular field-of-vision (set to a radius of 8 around the agent). Each observation matrix listed above is shifted so that the agent is in the middle of the frame. An episode is terminated if the entire area has been covered, the number of timesteps is greater than 153[4], or if any agent has not covered a new cell for the last 10 timesteps. The communication graph is **fully-connected** at all times. This last choice was made in order to

[3]The original github repo of Blumenkamp and Prorok [2021] can be found here: `https://github.com/proroklab/adversarial_comms`.

[4]$16 \times 16 \times 0.6 = 153.6$

properly assess whether *any* type of communication helps, and therefore a fully-connected topology allows the agents to focus on any of the connections they consider necessary.

I run 4 experiments in total; in two of these, the agents have “low visibility”: observations about the local obstacle map are limited to cells within a Manhattan distance of 2 around the agent. The reason for this will be explained in detail in section 4.

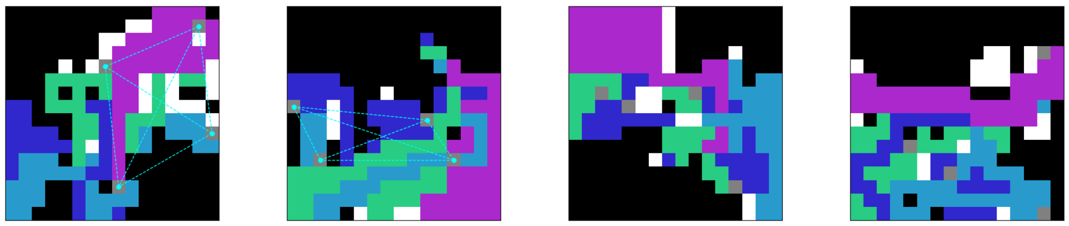


Figure 2: Coverage experiments. Black cells represent obstacles, gray cells represent agents, white cells represent non-covered cells. The other 4 colours represent the coverage achieved by the four agents at the end of an episode. Maps (a) and (b) are taken from experiments in which agents are allowed to communicate. Maps (c) and (d) are taken from experiments in which agents are not allowed to communicate.

3.2 Training setup and hyperparameters

Training and model hyperparameters are inspired from Blumenkamp and Prorok [2021]. I run 4 experiments, two on my personal laptop (Apple M2, 24GB RAM and 8 cores) and two on the `dev-gpu` provided by the Computer Science Department for Masters’ students (NVIDIA A100, 128 cores). Due to computational and time constraints, I run each of the 4 experiments for 900 training iterations. While this is a significantly lower number of iterations than in the original paper, I found that due to the smaller map size, the mean cumulative reward of every episode plateaus rapidly.

PPO hyperparameters. I train with a discount factor $\gamma = 0.9$, a clipping parameter $\epsilon = 0.2$ and the GAE bias-variance parameter $\lambda = 0.95$. The training batch size is 1000 and the SGD mini-batch size is 100. The number of SGD iterations per training batch is 5.

4 Experiments

The main goal of my experiments is to study communication benefits in a cooperative setting where agents do not share a centralised critic; I believe that in such a setting, the pressure for the agents to employ communication strategies is higher than in a setting where a centralised critic can nudge actors towards policies that are better for the group as a whole.

Experiment settings. To properly assess the benefit of communication I run two types of experiments:

1. **Agents have high map visibility**, meaning that they can see all map obstacles within a Manhattan distance of 8 (e.g., when they are in the centre of the map, they can see the map in its entirety, but if they are in a corner, they will only see a quarter);
2. **Agents have low map visibility**, meaning that they can see all obstacles within a Manhattan distance of 2. I hypothesise that in this case, sharing information is necessary for the group to achieve a high cumulative reward, since agents cannot assess “the bigger picture” by themselves.

Figure 3 shows an example of the obstacle map perceived by two agents in the two settings.

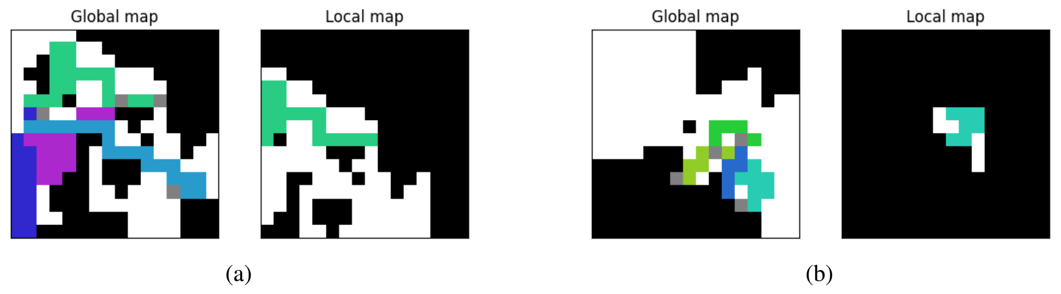


Figure 3: Example of agent visibility. Subfigure 3a shows how the agent perceives the obstacle map in the high visibility setting; subfigure 3b shows how the local obstacle map looks like for the agent in the low visibility setting. In both cases, own historical coverage is overlapped.

4.1 Results

As detailed in Section 3.2, I run 4 experiments for 900 training steps. Figure 4 shows the mean cumulative reward throughout training. We can see that in the low visibility setting, the agents perform slightly worse than in the high visibility setting. Perhaps more interestingly, there aren't any significant differences between agents that employ communication and agents that do not.

I further perform an evaluation run for which the results are summaried in Table 1. The evaluation is run for 10k environment steps on each of the 4 learned policies. I further discuss gained insights from these experiments.

Discussion. The evaluation run suggests that **there aren't any major differences** between the experiments w/ comms and the experiments w/o comms. There may be many interpretations for this behaviour:

- **The training run is too short for meaningful differences to arise**. Indeed, the high standard deviation ($\approx \pm 28$) present across all experiments indicate that longer training may be necessary for the learned policies to stabilise, or that better hyperparameters should be used. Alternatively, a high standard deviation could be a result of having decentralised critics and therefore slower learning. As mentioned earlier in this report, the training duration limitation is present due to both time and computational constraints. While future work should focus on re-running these experiments for longer, I hope the insights gained through this report are still meaningful for developing a better understanding of communication in cooperative tasks.
- **The environment is too small for cooperation via communication to be necessary**. Even in a low visibility setting, agents manage to cover most of the cells. With 153 empty cells to cover, each agent can cover on average $\frac{153}{4} \approx 38$ cells. In the low visibility setting, the agent sees $5 \times 5 = 25$ cells. Therefore, we can argue that an agent, on average, does not need to explore a great deal of the map to reach a good enough reward. Future work should focus on increasing the map size and reducing the number of obstacles in order to analyse agent exploration behaviour in more detail. Additionally, a baseline for agents with a centralised critic should also be implemented, such that cooperative behaviour can be quantitatively compared to the behaviour induced by a centralised critic.

Finally, while the evaluation run indicates that there may be a improvement when adding communication in the high visibility setting, a longer training run should be performed before drawing any conclusions. As discussed above, the reward difference is within the standard devation of the total reward of the two experiments, so longer and bigger experiments are necessary for a clear indication of communication benefits.

5 White box analysis

Even though preliminary results indicate that there aren't any major benefits to cooperation via communication in agents with decentralised critics, I perform a white box analysis of agents' latent

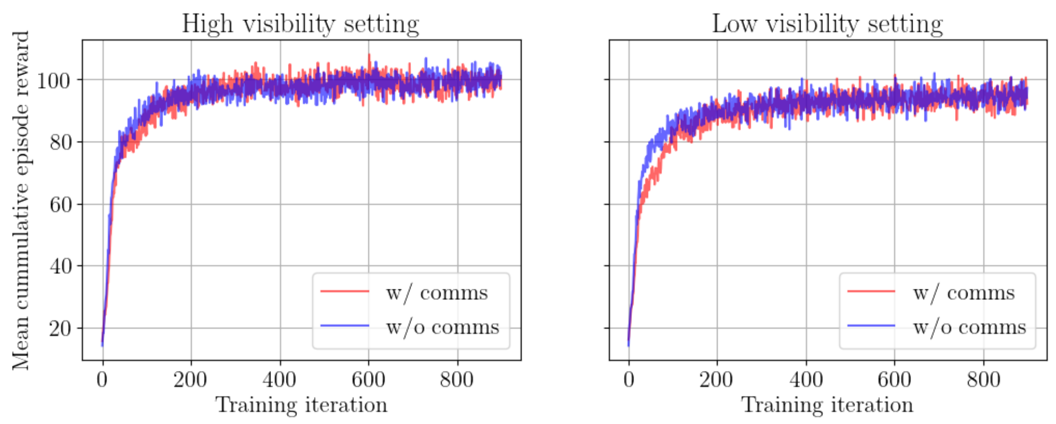


Figure 4: Mean cumulative reward across training iterations for the 4 experiments. On the left we see the reward curve of two experiments in which agents have *high obstacle visibility*, meaning that they can see obstacles within a Manhattan distance of 8. On the right, the agents have *low obstacle visibility*, meaning they can only see obstacles within a Manhattan distance of 2.

Table 1: Evaluation results for 4 experiments. Evaluation is run over 10k environment steps. For the total reward, I report both the mean and the standard deviation.

Visibility	Comms	Mean total reward	Max total reward	Average agent reward	Average max agent reward
High	Enabled	$106.23_{\pm 27.33}$	144	26.55	32.57
	Disabled	$98.50_{\pm 29.75}$	144	24.62	30.65
Low	Enabled	$96.27_{\pm 29.63}$	145	24.06	30.12
	Disabled	$97.06_{\pm 30.31}$	144	24.26	30.52

features to understand what messages (if any) they choose to send to their neighbours. Inspired by Blumenkamp and Prorok [2021], I design an *inverse* CNN that takes a latent feature and maps it to a possible 16×16 map. The upsampling CNN has two subnetworks; the first subnetwork is trained to predict the local obstacle map of each agent, while the second subnetwork is trained to predict the historical local coverage of the agent. I train the network on features of the agents with high visibility and no communication, since these representations are not influenced by the existence of a communication channel. Figure 5 shows an example of a global map, a local ground truth map of an agent and the predicted local maps from the internal features it generates.

The upsampling CNN is trained on 50K examples from high visibility environments w/o comms. Interestingly, while it achieves a good performance when predicting obstacles, it struggles with predicting coverage. This may be the case because an agent's own history of coverage is not as relevant for exploring new parts of the map. Since the agent doesn't need to communicate this information, it decides it is not worth encoding in the latent space.

5.1 Interpreting communication

I use the interpreter for messages sent by agents allowed to communicate. Perhaps surprisingly, many environment snaphots paint an intriguing picture: some agents will malform their internal representations of obstacles and coverage. This could be interpreted as lying, or as an attempt to hide their true internal state. Figure 6 represents one such snapshot.

A more interesting situation can be observed for agents in the low visibility experiment w/ comms. In particular, by taking snapshots of the environment, we can notice how not a single agent is truthful about its coverage (see Figure 7), even though the agents are penalised as a group for their actions via the loss function defined in Equation 5.

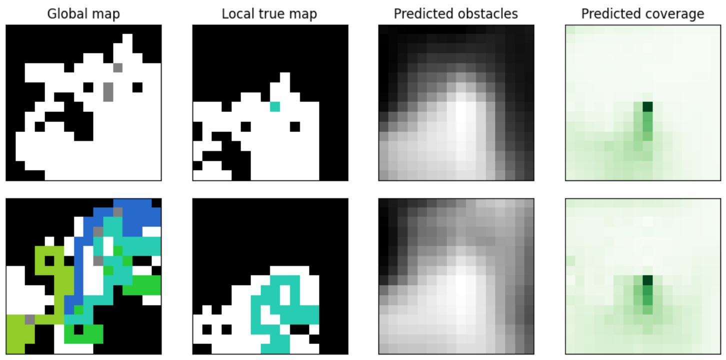


Figure 5: Examples of predicted obstacle and coverage maps from the latent features of agents in an environment w/o comms. As we can see, the interpreter does a good job with predicting obstacles, but struggles with coverage.

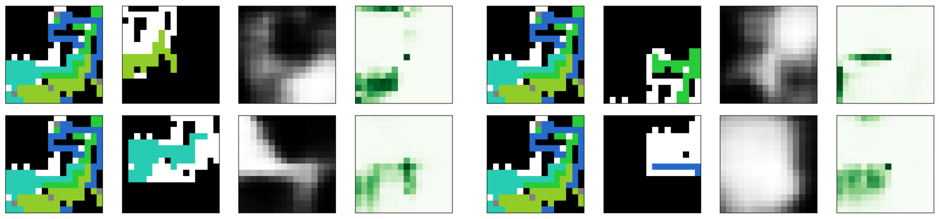


Figure 6: Environment snapshot of all 4 agents from a high-visibility experiment w/ comms. We can see that some agents lie about their coverage, while others lie about the obstacles they observe.

In the hope of gaining more insight into how agents counteract this liar effect, I plotted the value of all parameters within each agent's AGNN; although I had expected agents to protect themselves from other agents' lies by ignoring incoming messages, this does not necessarily appear to be the case. Figure 8 shows the parameters of the two filters present in each agent's AGNN. `H_0` is a filter applied to the agent's own latent features, while `H_1` is a filter applied to all messages in a 1-hop neighbourhood. Given that our graph is fully connected, the second layer takes into account incoming messages from all other agents.

I believe that while this behaviour is surprising, a more thorough analysis should be done on other types of environments before drawing results. In particular, it would be interesting to assess whether this mildly adversarial behaviour is optimal to the group as a whole, given the environment is small enough that the agents do not need to communicate factual information (especially in the high visibility setting).

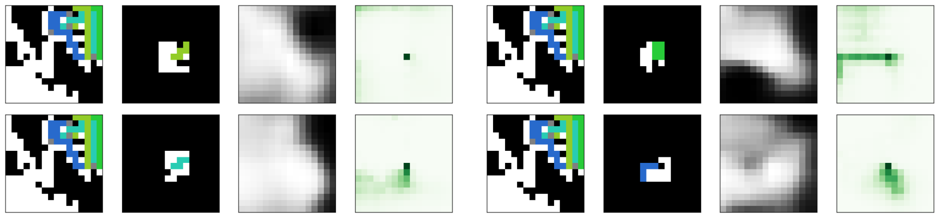


Figure 7: Environment snapshot of all 4 agents from a low-visibility experiment w/ comms. We can see that all agents lie about their true coverage.

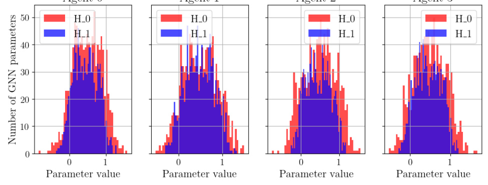
6 Conclusion

This project has investigated communication behaviour in cooperative agents with decentralised critics. While one could argue that the empirical results gained in this project raise more questions than they answer, I wish to underline some key takeaways:

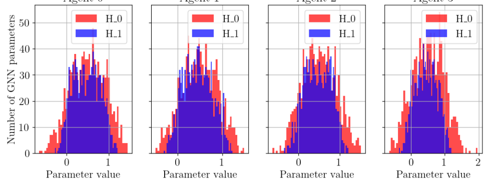
- **Introducing communication channels does not immediately yield performance improvements**. While these improvements may be harder to achieve due to the small size of this experiment, we notice that the standard deviation of the mean result in the experiment w/ comms is as big as in the experiments w/o comms. This may indicate, at least, that communication does not necessarily stabilise training.
- **When given communication channels, some agents choose to lie, while others tell the truth**. Although it is not immediately clear why this may be the case, this can be either an indication that decentralised agents need a better reward function to adopt fully cooperative behaviours, or an example of a situation where adversarial behaviour is optimal for the group as a whole.

Overall, the biggest limitation I found in this project is the lack of computational resources to run bigger and longer experiments, whose results may provide more convincing answers regarding the behaviours observed in this report. Additionally, an interesting path for future work is an investigation into the benefits of communication in adversarial MARL, and how the introduction of a communication channel changes the dynamics of environments with *self-interested agents*.

MARL with differentiable communication channels is an active research area, and many real-world applications can be modelled via agents with decentralised critics that communicate, such as network routers belonging to different Internet Service Providers, self-driving cars belonging to different companies, or flying drones belonging to different owners. At first glance, it seems almost obvious that such systems would benefit from a communication channel that helps them maximise "public goods". As such, I hope this report has brought in useful insights about the possible behaviours of decentralised agents in a cooperative environment.



(a) High visibility experiment.



(b) Low visibility experiment.

Figure 8: Histograms showing the distribution of values of parameters in each agent's AGNN. `H_0` is a filter applied to an agent's own message. `H_1` is a filter applied to all other neighbours.

References

Jan Blumenkamp and Amanda Prorok. The emergence of adversarial communication in multi-agent reinforcement learning. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1394–1414. PMLR, 16–18 Nov 2021. URL `https://proceedings.mlr.press/v155/blumenkamp21a.html`.

Jan Blumenkamp, Steven Morad, Jennifer Gielis, Qingbiao Li, and Amanda Prorok. A framework for real-world multi-robot systems running decentralized gnn-based policies. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8772–8778. IEEE, 2022.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Arbaaz Khan, Ekaterina Tolstaya, Alejandro Ribeiro, and Vijay Kumar. Graph policy gradients for large scale robot control. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 823–834. PMLR, 30 Oct–01 Nov 2020. URL `https://proceedings.mlr.press/v100/khan20a.html`.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Ryan Kortvelesy and Amanda Prorok. Modgnn: Expert policy approximation in multi-agent systems with a modular graph neural network architecture. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9161–9167, 2021. doi: 10.1109/ICRA48506.2021.9561386.

Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. Graph neural networks for decentralized multi-robot path planning, 2019. URL `https://arxiv.org/abs/1912.06095`.

Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning, 2017. URL `https://arxiv.org/abs/1712.09381`.

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Philipp Dominic Siedler. The power of communication in a distributed multi-agent system. *arXiv preprint arXiv:2111.15611*, 2021.

Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference* *on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 671–682. PMLR, 30 Oct–01 Nov 2020. URL `https://proceedings.mlr.press/v100/tolstaya20a.html`.