Specializing Communication in Heterogeneous Deep Multi-Agent Reinforcement Learning using Agent Class Information

Douglas De Rizzo Meneghetti **Reinaldo Augusto da Costa Bianchi**
FEI University Center
Humberto de Alencar Castelo Branco Ave., 3972-B
São Bernardo do Campo, SP, Brazil, 09850-901
douglasrizzo@fei.edu.br, rbianchi@fei.edu.br

Abstract

Inspired by recent advances in agent communication with graph neural networks, this work proposes the representation of multi-agent communication capabilities as a directed labeled heterogeneous agent graph, in which node labels denote agent classes and edge labels, the communication type between two classes of agents. We also introduce a neural network architecture that specializes communication in fully cooperative heterogeneous multi-agent tasks by learning individual transformations to the exchanged messages between each pair of agent classes. By also employing encoding and action selection modules with parameter sharing for environments with heterogeneous agents, we demonstrate comparable or superior performance in environments where a larger number of agent classes operates.

Introduction

In partially observable multi-agent settings, communication among agents may help mitigate the uncertainty with relation to which state the agent currently is, given its observation. Communication has been achieved in deep multi-agent reinforcement learning by multiple means: with agents directly sharing their observations (Sunehag et al. 2018) or learning message passing mechanisms via backpropagation. This was initially achieved with multi-layer perceptrons (Sukhbaatar, Szlam, and Fergus 2016), bidirectional recurrent neural networks (Peng et al. 2017), considering communication as part of agent actions (Foerster et al. 2016) or allowing an agent to learn which agents to communicate with in the environment, without restraints (Hoshen 2017; Jiang and Lu 2018; Malysheva, Kudenko, and Shpilman 2019; Das et al. 2019).

A recent spike in interest in the areas of graph neural networks and geometric deep learning resulted in works which explore the use of operations previously targeted towards supervised and semi-supervised learning on graphs to achieve agent communication (Wang et al. 2018; Agarwal, Kumar, and Sycara 2019; Malysheva, Kudenko, and Shpilman 2019; Jiang et al. 2020). However, most of these works focus in tasks with homogeneous agents, that is, agents that have the same observation and action spaces and follow the same policy.

Many tasks present agents that would either benefit from learning specific policies or that have completely different observations and actions. An example of the former is the RoboCup robot soccer leagues (Chalup et al. 2019), in which teams of homogeneous robots may specialize into offensive, defensive and goalkeeping roles. Examples of the later include mixed robot teams, such as aerial drones used for mapping and terrestrial robots tasked with navigation, as well as real-time strategy (RTS) games. In an RTS game, each unit belongs to an unit type, which dictates its skills, strengths and weaknesses and, consequently, its policy.

Another property of the aforementioned examples is the fact that a collection of heterogeneous agents can be group into classes (robot soccer roles, such as offense, defense and goalie, or game unit types), where agents from the same class may be considered homogeneous among themselves.

Given these scenarios, this work proposes a neural network model that learns specialized communication protocols between classes of agents. The method first creates a directed labeled agent communication graph, in which node labels represent agent classes and edge labels represent a communication channel between agents of two classes. The neural network model then employs relational graph convolutions (Schlichtkrull et al. 2018) as a message passing mechanism between agents, before estimating Q values or all actions. We also show how heterogeneous classes of homogeneous agents can share parameters in parts of the model, accelerating learning.

Research Background

When focusing on fully-cooperative, partially observable multi-agent tasks, one way to formalize the environments is through decentralized partially observable Markov decision processes Dec–POMDPs (Amato 2015). A Dec–POMDP is defined as a tuple $< U, S, \mathcal{A}, \Omega, P, R, O >$, where

- U: a finite set of agents,
- S: a finite set of states,
- $\mathcal{A}_u$: finite set of actions for agent u,
- Ω_u: a finite set of observations for agent u,

- P: a transition probability function $P(s'|s, \boldsymbol{a})$ mapping the joint actions chosen by all agents in state s to the probability of transitioning to state s',
- R: a shared reward function $R(s, \boldsymbol{a})$,
- O: an observation model $O(\boldsymbol{o}|s', \boldsymbol{a})$ which dictates the probabilities that agents will observe $\boldsymbol{o}$ in s' after taking $\boldsymbol{a}$.

Given a pair of agents u_1 and u_2, they are considered homogeneous when $\mathcal{A}_{u_1} = \mathcal{A}_{u_2}$, $\Omega_{u_1} = \Omega_{u_2}$ and they can share the same policy without affecting outcomes. If at least one of these conditions is not true, then they are considered to be heterogeneous.

Deep Multi-Agent Reinforcement Learning

Recent advances in the area of deep multi-agent reinforcement learning that contributed to this work are the network architectures trained with off-policy algorithms to control multiple agents in fully cooperative, partially observable environments. Reinforced Inter-Agent Learning and Differentiable Inter-Agent Learning (Foerster et al. 2016) model agents as deep recurrent Q-Networks (DRQN) (Hausknecht and Stone 2015) to address partial observability, while mixing techniques such as value decomposition networks (VDN) (Sunehag et al. 2018) and QMIX (Rashid et al. 2018) train a multi-agent neural network by combining agents' Q values under the assumptions of additivity and monotonicity, respectively, to achieve superior performance in fully cooperative tasks. Agents then choose actions according to their individual Q values, achieving what is called centralized training and decentralized execution.

Especially in the case of VDN, the joint value function of a team of cooperative agents is given by

$$Q_U(\boldsymbol{o}, \boldsymbol{a}) = \sum_{u \in U} Q_u(o_u, a_u).$$

Graph Neural Networks

Graph neural networks (GNN) (Gori, Monfardini, and Scarselli 2005; Scarselli et al. 2009) are models specialized in processing input data structured as graphs or manifolds. GNNs are composed of multiple types of operations (Zhou et al. 2018). However, one class of operations over graphs that is particularly interesting in the MARL setting are graph convolutions (Defferrard, Bresson, and Vandergheynst 2017; Kipf and Welling 2017). These operations can be interpreted as message passing mechanisms between nodes of a graph (Gilmer et al. 2017) and, while their main applications have been in supervised and semi-supervised learning in graph data sets, they have also found use in reinforcement learning tasks, as message passing mechanisms between agents (Sukhbaatar, Szlam, and Fergus 2016; Malysheva, Kudenko, and Shpilman 2019; Jiang et al. 2020).

In the family of graph convolution operations, one that is particularly interesting in this work are relational graph convolutions (RGCN) (Schlichtkrull et al. 2018). RGCNs operate over graphs defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ represents a set of nodes containing individual feature vectors, $\mathcal{E}$ is a set of edges and $\mathcal{R}$ a set of possible relations between nodes.

The feature vector of node i in layer $l+1$ is given by

$$\vec{v}_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} \boldsymbol{W}_r^{(l)} \vec{v}_j^{(l)} + \boldsymbol{W}_0^{(l)} \vec{v}_i^{(l)}\right), \quad (1)$$

where σ is a nonlinear activation function, r is the index of the relation between nodes i and j, $\mathcal{N}_i^r$ are the neighbors of i under connected by edges with relation index r and $\boldsymbol{W}_r^{(l)}$ is a parameter matrix specific to relation r in layer l.

Related Work

Other works have already modeled agent communication capability as graphs. Agarwal, Kumar, and Sycara (2019) use a Transformer attention mechanism (Vaswani et al. 2017) to let agents learn the relevance of neighbors' messages, while DGN (Jiang et al. 2020) concatenate the outputs of multiple graph convolution layers to form agents' observations, under the rationale that each subsequent layer captures information from further away in the graph.

MAGNet (Malysheva, Kudenko, and Shpilman 2019) uses a neural network to generate weighted edges in the agent graph. This graph generation network is pre-trained in environments which possess default agents and its output is then used by the multi-agent reinforcement learning network to learn policies for all agents.

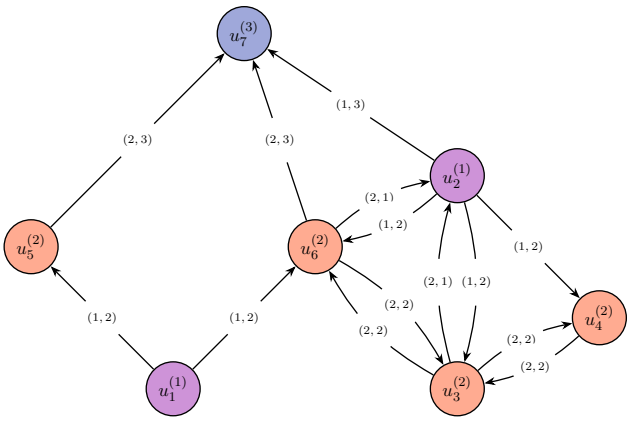
While the previous mentioned works have only dealt with homogeneous agents, i.e. agents that work with the same $\mathcal{A}$ and Ω sets as well as learn the same policy, NerveNet (Wang et al. 2018) models a creature's skeleton as a group of heterogeneous agents by categorizing joints with equal functionality as agents of the same type. A static graph of the creature's skeleton is then generated and messages are passed among nodes by using specialized MLPs for each pair of node types.

In previous work (Meneghetti and Bianchi 2020), a neural network architecture was presented to deal with graphs composed of heterogeneous agents and entities, in which node vectors represented an entity's features instead of its observations. Due to partial observability of the environment, the graph structure proposed in that work was not capable of representing all information necessary for agents to learn well-performing policies, an issue which has been addressed in this work.

The current work is different from Agarwal, Kumar, and Sycara (2019) and DGN by taking into account partial observations, using LSTM layers in the agent networks and training agents using an episodic replay buffer (Hausknecht and Stone 2015) instead of a buffer containing transitions (Mnih et al. 2015). This work also differs from MAGNet and other works in which an agent may freely choose to communicate with any other agent in the environment (Foerster et al. 2016; Hoshen 2017; Jiang and Lu 2018; Das et al. 2019), as well as NerveNet, in which the agent communication graph is fixed, given the environment. In our work, communication capabilities are dictated by the environment and may change between states, being out of the control of the agents.

Lastly, it is worth noting that the use of agent classes to model MDPs has previously been proposed in object-

Figure 1: A heterogeneous agent graph, represented as a directed labeled graph



oriented MDPs (Wasser, Cohen, and Littman 2008) and its multi-agent variant (da Silva, Glatt, and Costa 2019).

Representing states as heterogeneous agent graphs

In this section, we introduce the concept of a heterogeneous agent communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, C.\mathcal{R})$. Each node in the set of nodes $\mathcal{V}$ represents an agent in the set of agents U, so, for the rest of this work, individual agents and nodes will both be represented by u.

A directed arc $(u_1, u_2) \in \mathcal{E}$ denotes the capability of u_1 to communicate with u_2. By making $\mathcal{G}$ a directed graph, we are able to represent one-way communication between agents when necessary.

In some tasks, the team of agents may have full communication among themselves, making $\mathcal{G}$ a complete graph. In other tasks, an agent may be limited to communicate only with agents that are geographically close or with which some kind of communication channel is available. In this work, we make no assumptions regarding agent communication capabilities and assume they may change from state to state.

The set C represents a collection of agent classes. All agents $u \in U$ belong to a single class in C, in such a way that, while U may be considered a set of heterogeneous agents, the subset of all agents belonging to a class $c \in C$, denoted by U_c, is homogeneous.

Let $C(u)$ be a function that returns the class of u. An arc (u_1, u_2) is labeled $(C(u_1), C(u_2))$, indicating a communication channel from an agent of class $C(u_1)$ to an agent of class $C(u_2)$. This, coupled with the fact that agent class information is encoded in $\mathcal{G}$ through the use of node labels, makes $\mathcal{G}$ a labeled graph.

The set of all possible inter-agent communication channel types is denoted by $\mathcal{R}$ and is defined as all possible ordered pairs of agent classes $(c_1, c_2), c1 \in C, c2 \in C$, totaling $|C|^2$ possible types.

Figure 1 presents an example of such a graph. In it, node colors and superscripts represent an agent's class and bidirectional communication is represented by a pair of arcs.

Heterogeneous Communication in Deep Multi-Agent Reinforcement Learning

Figure 2 presents the proposed neural network architecture. It is composed of three modules. The input data is composed of $\boldsymbol{o}$, the observations of all agents. The encoding module applies a function ϕ to $\boldsymbol{o}$, normalizing its dimensions and enhancing the expressiveness of the model.

In the communication module, the agent graph structure is used for message passing among agents through the use of RGCN layers. Equation (1) shows how communication between different pairs of classes of agents is learned and performed differently. By representing each agent-class pair $(c_1, c_2), c1 \in C, c2 \in C$ as a relation $r \in \mathcal{R}$, an RGCN layer is able to model message passing between each pair of agent class using individual parameter matrices, allowing the overall network to shape the way agents of specific classes communicate with each other independently.

The resulting vectors after applying K graph convolution layers are taken as the combination of each agent's observation and the information received by its neighbors. They are then used as input to the action selection module, which also employs a single function that approximates Q values for all agents, sharing parameters in an analogous way to the encoding function ϕ.

Parameter sharing for teams of heterogeneous agents

In Meneghetti and Bianchi (2020), both the encoding and action selection modules were composed of multiple functions ϕ_c and Q_c,$c \in C$, specialized in agent classes with observation vectors of different dimensions, as well as action sets of different sizes. However, we found that using the same input and output dimensions for all agent classes and employing zero padding in the unused variables allowed for the use of a single function in each case, achieving parameter sharing and faster training of the model.

Furthermore, when using the same action selection network for multiple heterogeneous agents, we have to account for the fact that the network will estimate Q values for the joint set of actions of all agent classes. When choosing an action according to their policies, each agent must ignore values for actions outside of its action set. These Q values must also be ignored when calculating the temporal difference error that trains the network.

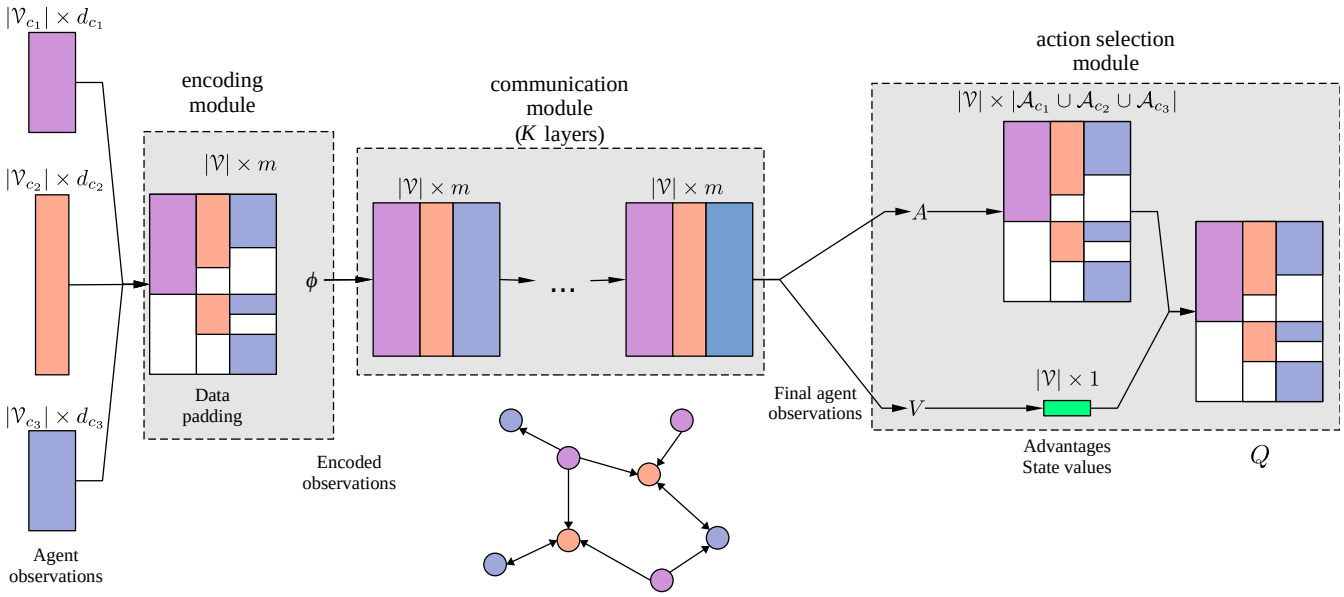
Let $\mathcal{A}_c$ denote the action set of agent class c and $\mathcal{A} = \bigcup_{c \in C} \mathcal{A}_c$ the joint set of agent actions. Whenever the network estimates Q for an action $a \in \mathcal{A} - \mathcal{A}_{C(u)}$ for an agent u, we ignore it during action selection and consider its contribution to the TD error as 0.

This parameter sharing approach for heterogeneous agents was formalized in Terry et al. (2020) for the general case in which all agents in a team are heterogeneous. Here we apply the same rationale when a heterogeneous group of agents can be grouped into classes of homogeneous agents.

Experiments

The proposed network architecture was tested in the *StarCraft Multi-Agent Challenge* (SMAC) environ-

Figure 2: Neural network architecture for the heterogeneous agent setting.



ment (Samvelyan et al. 2019). A collection of scenarios was chosen to evaluate the performance of the model under different circumstances, such as a scenario with a single agent class (3m), scenarios with increasing number of agent classes (3s5z and 1c3s5z) and scenarios in which both teams had the same number of units (MMM) vs an equivalent scenario in which the agent team had a disadvantage in number of units (MMM2).

Actions in the SMAC scenarios include: moving in one of four directions cardinal directions, attacking an enemy unit (select by an ID), stopping and the no-op action. The Medivac unit, present in the MMM and MMM2 scenarios, has actions to heal allied units instead of attacking enemies. Due to the parameterized nature of the attack and heal actions, units in different scenarios may have a varying number of actions.

The encoding function ϕ is a single layer perceptron with 96 neurons and tanh activation; the action selection module is implemented as a Dueling Deep Recurrent Q-Network (Mnih et al. 2015; Hausknecht and Stone 2015; Wang et al. 2016) with the Double Q-Learning loss function (van Hasselt, Guez, and Silver 2016).

We tested three different variants of the communication module. The first one uses the proposed heterogeneous agent communication graph and two RGCN layers (Schlichtkrull et al. 2018) with two bases. The second module employs two graph attention (GAT) layers (Veličković et al. 2018) with three attention heads. GAT layers do not explicitly specialize message passing according to agent classes, but may do so implicitly if agent class information is added to the agent observation. In both cases, each layer is composed of 96 hidden units and employs the LeakyReLU activation. The last variant of the communication module is no communication at all.

To determine the orthogonality of mixing strategies (which have been applied by other works in the SMAC environment) with the proposed agent communication method, we include in our analysis one method that approximates individual value functions for each agent, known as independent Q-Learning (IQL) (Tan 1993) and another one that learns the additive joint value function for the team of agents (VDN) (Sunehag et al. 2018).

Table 1: Hyperparameters used in the training setting

$\hat{\theta}$ update interval	250
Network learning rate	$2.5 * 10^{-4}$
L2 regularization coef.	10^{-5}
RL discount factor γ	0.99
ϵ_{min}	0.1
ϵ_{max}	0.95
N. steps to finish linear ϵ decay	50000

Agents use a joint ϵ-greedy policy, in which all agents either explore with probability ϵ or exploit with probability $1 - \epsilon$. A linear ϵ decay schedule is employed, whose relevant values, as well as other hyperparameters used in the experiments, are displayed in table 1.

Every 10 thousand steps, we execute an evaluation step in which the network follows a greedy policy for 32 episodes. We log the win rate in these episodes, as well as the average number of defeat enemies (as an intermediate metric, in case agents do not win a match).

Following recommendations from the literature (Rashid et al. 2020), we train each version of the model in each of the chosen maps five times for 1 million steps, reporting the 25, 50 and 75 percentiles of the aforementioned metrics in each combination. We also trained select versions of the model for 6 million steps on a single map (3s5z) to investigate the results of longer training periods. Each 1 million-step training procedure took from 5 to 11 eleven wall-clock hours in

an Nvidia GTX 1070 GPU.

Results

Figure 3 displays the win rate and average number of defeat enemies in the tested SMAC maps. While in other works (Rashid et al. 2018, 2020), authors present different results for equivalent algorithms (IQL, VDN without communication) in the same maps, this may be due to the use of a different version of StarCraft.

While the number of defeated enemies in the homogeneous 3m map (3b) grows as training progresses, all methods seem to display equivalent performance with relation to win rate (3a). The same can be said for the heterogeneous 3s5z map (which we investigate further in forthcoming results) and the MMM2 map, which is considered a hard map due to the asymmetry between the agent and enemy teams.

In the 1c3s5z which is a symmetric scenario with the most classes of agents (3), the methods display different performance. IQL with specialized communication achieved superior win rates and defeated more enemies than IQL with communication performed using an attention mechanism as well as no communication (figs. 3e, 3f). In this scenario, communication using an attention mechanism seemed to harm agent performance, performing the worst of all. Lastly, we can see that the model that used a combination of specialized communication and value decomposition achieved the highest values in our metrics, implying a benefit in using specialized communication in heterogeneous multi-agent settings over no communication or attention mechanisms.

Lastly, in the MMM scenario, which also has three agent classes, although the model with specialized communication and value decomposition tended to defeat the most number of agents in average (fig. 3h), all models displayed a tendency to reach equivalent final performance, implying that, while not negatively affecting performance, communication may not be necessary in all scenarios.

To discover whether there are relevant differences in the performance of the methods, we employed analysis of variance on all six tested methods in each of the scenarios separately, followed by a Tukey HSD test. Figure 4 presents results of Tukey's test applied to the data collected in the final five evaluation steps of all methods. We omit tests which are highly similar (scenarios with both a high win rate and number of defeated enemies) and tests performed in the 3m scenario, in which no statistically significant difference among the methods was observed (possibly due to agent homogeneity).

In some scenarios, there are indications that the use of specialized communication and additive mixing may be comparable (fig. 4a). Other measures indicate that the use of an attention mechanism for communication may have been detrimental in some scenarios, when compared to specialized or no communication (fig. 4b). Lastly, while in some scenarios it may be unclear which is the best performing method, methods that employ specialized communication are constantly in the groups of highest performance (fig. 4a, 4b, 4c, 4d).

Given the poor win rate achieved by all models in the symmetric and heterogeneous 3s5z scenario (fig. 3c, an additional experiment was conducted, in which a selection of the models was trained for a total of 6 million steps. Models were selected in an ablative fashion, to contrast: the contribution of specialized communication over no communication; the orthogonality of the use of a mixing strategy over IQL and the performance of specialized communication over an attention mechanism.

The win rate and number of defeat enemies for the 6 million training steps in the 3s5z map are displayed in figure 5. We can see that, in this scenario, agents only start to win episodes after approximately 3 million training steps (fig. 5a). Also, only the models that implemented specialized communication demonstrated winning behavior, with the attention model with value decomposition winning an episode only after 530 million steps.

In figure 5b, we see that the models with specialized communication achieve the highest performance, with the one employing value decomposition being the best of all. This suggests that the application of mixing strategies achieve an orthogonal increase in performance when applied with the proposed specialized agent communication method.

Conclusion

This work presented a method to model communication between heterogeneous agents in fully cooperative multi-agent deep reinforcement learning tasks through the use of relational graph convolutions. Agents are assigned to classes, in such a way that agents in the same class are homogeneous among themselves. The communication capabilities of the whole team in a given state is represented as a directed labeled agent graph, which encodes inter-agent class relations in the graph edges. Relational graph convolutions are used to specialize communication channels between pairs of agent classes using separate parameter matrices, which are reused every time two agents represented by the same ordered pair of classes communicates.

Experiments performed in a selection of scenarios from the StarCraft Multi-Agent Challenge show that the proposed method achieves equal or superior performance in all maps, when compared to models which use attention mechanisms for communication, or no communication at all. We also showed indications of how the proposed communication method can be combined with additive mixing, achieving even better results in the tested scenarios.

Acknowledgments

The authors acknowledge the São Paulo Research Foundation (FAPESP Grant 2019/07665-4) for supporting this project. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

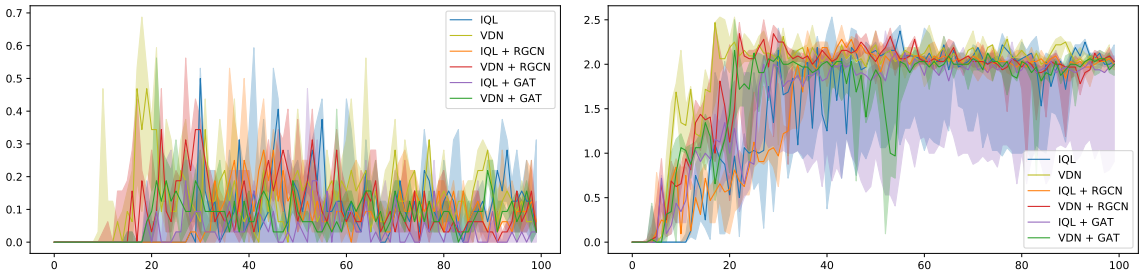
References

Agarwal, A.; Kumar, S.; and Sycara, K. 2019. Learning Transferable Cooperative Behavior in Multi-Agent Teams.

Figure 3: Win rate (left column) and number of defeat enemies (right column) for 1 million training steps (100 evaluation steps) in the tested SMAC maps

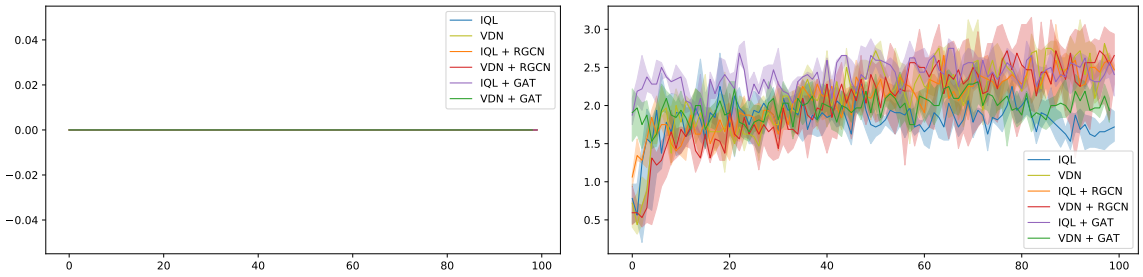
(a) Win rate – 3m

(b) Defeated enemies – 3m (max 3)



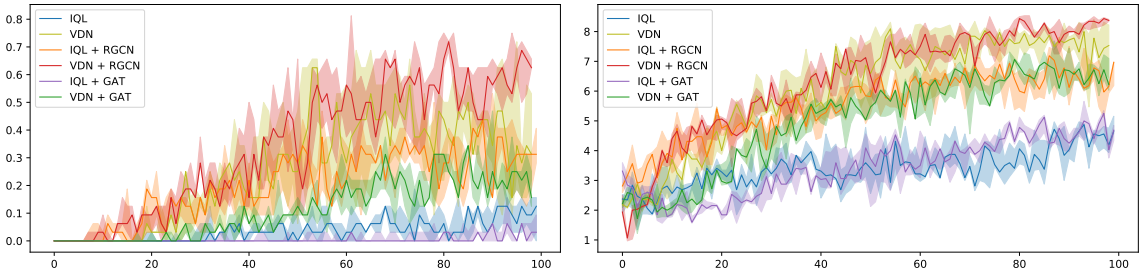
(c) Win rate – 3s5z

(d) Defeated enemies – 3s5z (max 8)



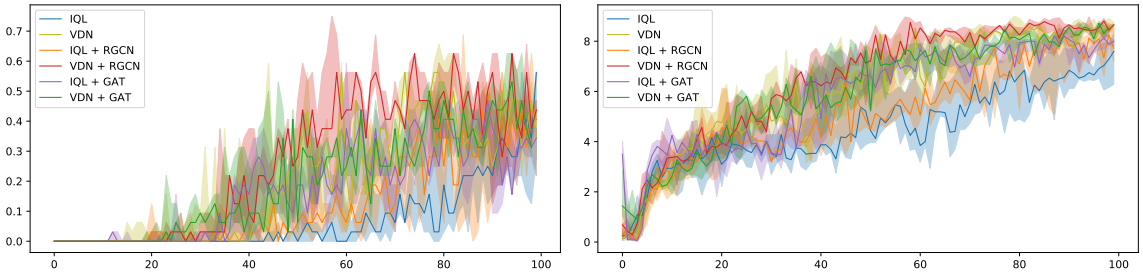
(e) Win rate – 1c3s5z

(f) Defeated enemies – 1c3s5z (max 8)



(g) Win rate – MMM

(h) Defeated enemies – MMM (max 10)



(i) Win rate – MMM2

(j) Defeated enemies – MMM2 (max 12)

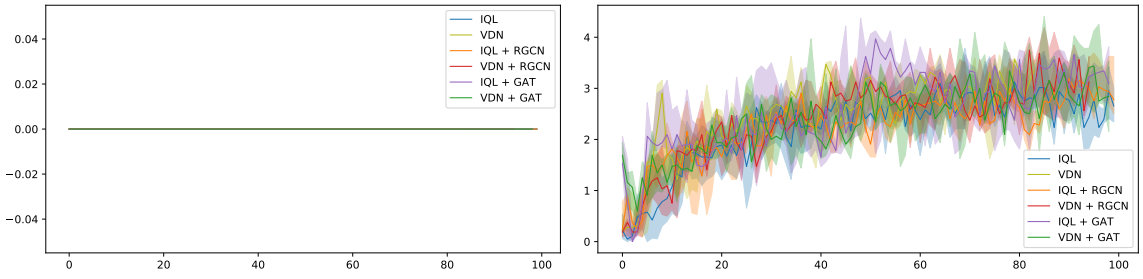
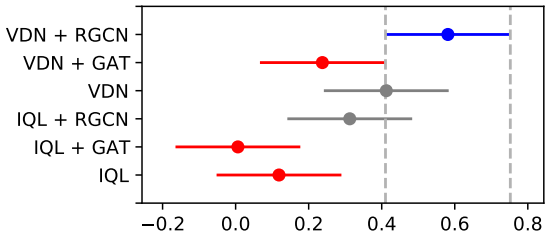
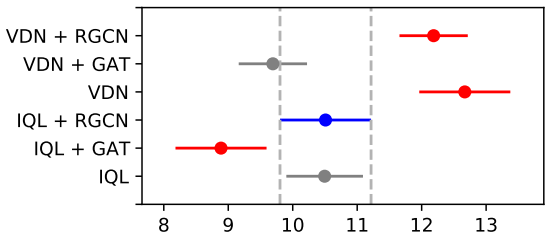


Figure 4: Tukey's HSD test in different performance measures of the different methods.

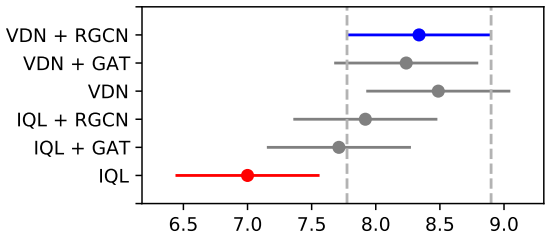
(a) Win rate – 1c3s5z



(b) Defeated enemies – 3s5z



(c) Defeated enemies – MMM



(d) Episode reward – MMM2

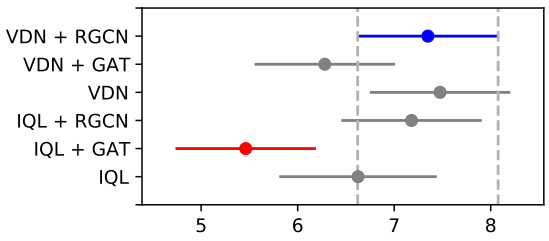
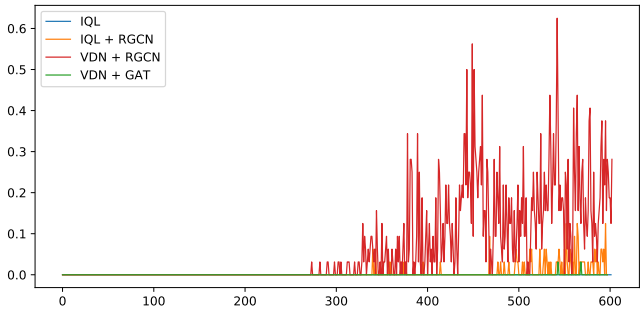
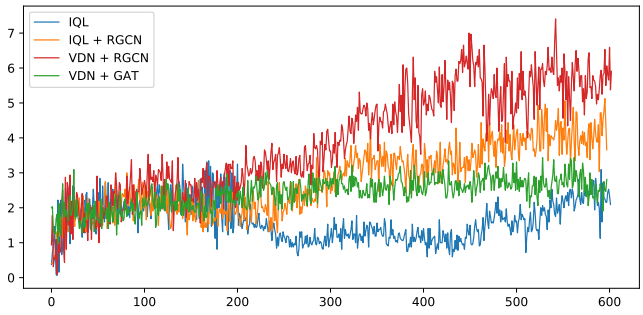


Figure 5: Win rate and defeated enemies for 6 million training steps (600 evaluation steps) in the `3s5z` map

(a) Win rate



(b) Defeated enemies (max 8)



In *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Representations*.

Amato, C. 2015. *Cooperative Decision Making*. The MIT Press. ISBN 978-0-262-33170-8. doi:10.7551/mitpress/10187.003.0011.

Chalup, S.; Niemueller, T.; Suthakorn, J.; and Williams, M.-A. 2019. *RoboCup 2019: Robot World Cup XXIII*, volume 11531. Springer Nature.

da Silva, F. L.; Glatt, R.; and Costa, A. H. R. 2019. MOO-MDP: An Object-Oriented Representation for Cooperative Multiagent Reinforcement Learning. *IEEE Transactions on Cybernetics* 49. doi:10.1109/tcyb.2017.2781130.

Das, A.; Gervet, T.; Romoff, J.; Batra, D.; Parikh, D.; Rabbat, M.; and Pineau, J. 2019. TarMAC: Targeted Multi-Agent Communication. *Proceedings of the 36th International Conference on Machine Learning* 97: 1538–1546.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2017. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *arXiv:1606.09375 [cs, stat]* .

Foerster, J.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2137–2145.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. *arXiv:1704.01212 [cs]* .

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A New Model for Learning in Graph Domains. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, 729–734. IEEE. doi:10.1109/ijcnn.2005.1555942.

Hausknecht, M.; and Stone, P. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI Fall Sym-*

posium - Technical Report, volume FS-15-06, 29–37. AI Access Foundation. ISBN 978-1-57735-752-0.

Hoshen, Y. 2017. VAIN: Attentional Multi-Agent Predictive Modeling. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 2701–2711. Curran Associates, Inc.

Jiang, J.; Dun, C.; Huang, T.; and Lu, Z. 2020. Graph Convolutional Reinforcement Learning. In *International Conference on Learning Representations*.

Jiang, J.; and Lu, Z. 2018. Learning Attentional Communication for Multi-Agent Cooperation. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 7254–7264. Curran Associates, Inc.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

Malysheva, A.; Kudenko, D.; and Shpilman, A. 2019. MAGNet: Multi-Agent Graph Network for Deep Multi-Agent Reinforcement Learning. In *Adaptive and Learning Agents Workshop at AAMAS (ALA 2019)*. Montreal, Canada.

Meneghetti, D. D. R.; and Bianchi, R. A. D. C. 2020. Towards Heterogeneous Multi-Agent Reinforcement Learning with Graph Neural Networks. In *Anais Do Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, 579–590. Porto Alegre, RS, Brasil: SBC. ISSN 0000-0000.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518(7540): 529–533. ISSN 0028-0836. doi:10.1038/nature14236.

Peng, P.; Wen, Y.; Yang, Y.; Yuan, Q.; Tang, Z.; Long, H.; and Wang, J. 2017. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-Level Coordination in Learning to Play StarCraft Combat Games .

Rashid, T.; Samvelyan, M.; de Witt, C. A. S.; Farquhar, G.; Foerster, J. N.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In Krause A., D. J., ed., *Proceedings of the 35th International Conference on Machine Learning*, volume 10 of *35th International Conference on Machine Learning, ICML 2018*, 6846–6859. International Machine Learning Society (IMLS). ISBN 978-1-5108-6796-3.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv:2003.08839 [cs, stat]* .

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. *arXiv:1902.04043 [cs, stat]* .

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20(1): 61–80. ISSN 1045-9227. doi:10.1109/tnn.2008.2005605.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference*, 593–607. Springer.

Sukhbaatar, S.; Szlam, A.; and Fergus, R. 2016. Learning Multiagent Communication with Backpropagation. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*, 2244–2252. Curran Associates, Inc.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; and Graepel, T. 2018. Value-Decomposition Networks for Cooperative Multi-Agent Learning Based on Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, 2085–2087. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Tan, M. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Machine Learning Proceedings 1993*, 330–337. Elsevier. doi:10.1016/b978-1-55860-307-3.50049-6.

Terry, J. K.; Grammel, N.; Hari, A.; and Santos, L. 2020. Revisiting Parameter Sharing in Multi-Agent Deep Reinforcement Learning. In *Accepted for Presentation at International Conference on Learning Representations (ICLR 2021)*.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, 2094–2100.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; ukasz Kaiser, \.; and Polosukhin, I. 2017. Attention Is All You Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 5998–6008. Curran Associates, Inc.

Veličković, P.; Casanova, A.; Liò, P.; Cucurull, G.; Romero, A.; and Bengio, Y. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

Wang, T.; Liao, R.; Ba, J.; and Fidler, S. 2018. Nervenet: Learning Structured Policy with Graph Neural Networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *International Conference on Machine Learning*, 1995–2003.

Wasser, C. G. D.; Cohen, A.; and Littman, M. L. 2008. An Object-Oriented Representation for Efficient Reinforcement Learning. In *Proceedings of the 25th International*

Conference on Machine Learning, 240–247. ACM. doi: 10.1145/1390156.1390187.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2018. Graph Neural Networks: A Review of Methods and Applications .