Multi-Agent Reinforcement Learning using Graph Networks

Manan Tomar Rohan Saphal

Abstract

Multi-agent reinforcement learning (MARL) has seen considerable developments over the past few years solving problems across a plethora of complex domains. Deep Reinforcement learning algorithms tend to perform poorly in environments that require multiple agents to coordinate, cooperate and compete with each other. MARL has potential applications across a variety of domains such as autonomous driving,robotic control, financial trading, etc. Modelling MARL in a structured way can result in significant improvement in performance. With the recent developments in Graph Networks, we demonstrate that modelling the multi-agent environment in a graph network paradigm can result in performance that is comparable or better than baseline deep reinforcement learning algorithms. We also show that our framework scales well with increase in the number of agents.

1. Introduction

Deep reinforcement learning (DRL) has seen considerable success over the past few years in a variety of fields such as game playing(Mnih et al., 2015; Silver et al., 2016; 2017), robotics(Levine et al., 2018), object detection(Mnih et al., 2014)etc. Deep RL algorithms have achieved super human performance starting from the seminal work on DQN(Mnih et al., 2015), that used pixel input and strategies such as experience replay and target networks.

Deep RL mainly has two paradigms, value based methods and policy gradient methods. There are multiple variants for each of these that have evolved over the years. Policy gradient methods address the optimization problem faced by value based methods in continuous action spaces. Out of the multiple variants of policy gradient methods, DDPG(Lillicrap et al., 2015) has been considered the baseline work in continuous control tasks, combining actor critic framework with DQN and Deterministic policy gradients(DPG)(Silver et al., 2014).

The study of multi-agent systems has received considerable attention in recent years and some of the most advanced autonomous systems in the world today are multi-agent in nature (e.g. assembly lines and warehouse management systems(Jaderberg et al., 2018; Leibo et al., 2017; Hughes et al., 2018; Peysakhovich & Lerer, 2017; Bansal et al., 2017; Lanctot et al., 2017). Deep RL algorithms that have been designed keeping in mind a single agent, tend to perform poorly in multi-agent environments owing to its non-stationary nature. Research involving sharing of information and parameters between multiple agents has led to recent developments in the field. One of the outstanding challenges in this domain is how to foster coordinated behavior among learning agents. In hand-engineered multi-agent systems (e.g. assembly lines), it is possible to obtain coordination by design, where expert engineers carefully orchestrate each agent's behavior and role in the system. However, these cannot scale to large number of agents or to mixed human-robot ensembles. There is thus an increasing focus on multi-agent systems that learn how to coordinate on their own

With the recent development in neural networks that perform relational reasoning using graph networks (Sanchez-Gonzalez et al., 2018), it is possible to build models that predict the forward dynamics of multi-agent systems. Previous works (Battaglia et al., 2016; Sukhbaatar et al., 2016) have shown considerable improvement in forward prediction using graph models that have also surpassed recurrent models. Recent work using graphical models have shown considerable performance for control tasks,both single agent(Sanchez-Gonzalez et al., 2018; Wang et al., 2018) and multi-agent systems(Hoshen, 2017; Jiang et al., 2018). Multi-Agent Deep Deterministic Policy Gradients (MADDPG) (Lowe et al., 2017) is a MARL algorithm built on top of the DDPG framework that shares parameters between multiple agents to allow for coordination, cooperation and competition between agents in different tasks.

An interesting and flexible platform for benchmarking and testing MARL algorithms is the Multi-agent particle environment (Mordatch & Abbeel, 2017). There are multiple tasks relating to cooperation, coordination, competition and communication that allows for testing the performance of different MARL algorithms. The platform is light-weight and is capable of scaling tasks with different permutations and combinations of agents. This helps to check for the robustness of the algorithms to permutations

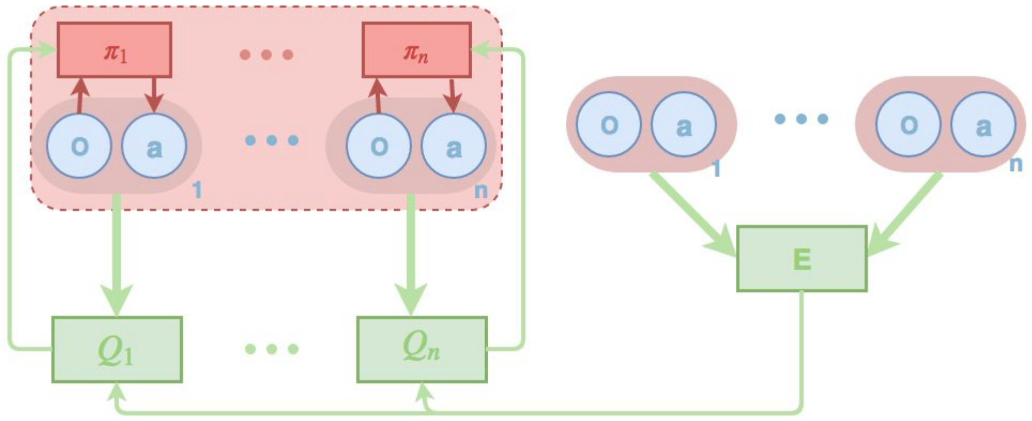

Figure 1. Proposed architecture for Multi-Agent Actor Critic using learnt GN predictions

in the agents and environment.
In this project, we develop an algorithmic framework that leverages the relational reasoning of graph networks to model the multi-agent environment. The framework aims to mitigate the difficulties and shortcoming of MADDPG.

2. Related work

MADDPG (Lowe et al., 2017) and Counter-factual multi-agent policy gradients (COMA) (Foerster et al., 2017) are the extension of actor-critic model for multi-agent environments, where MADDPG is designed for mixed cooperative-competitive environments and COMA is proposed to solve multi-agent credit assignment in cooperative settings. A centralized critic that takes as input the observations and actions of all agents are used in MADDPG and COMA.

Our work is primarily concerned with learning multi-agent interactions with graph structures. A notable iterative graph-like neural algorithm is the Neural-GPU(Kaiser & Sutskever, 2015).Notable works in graph NNs includes Spectral Networks (Gaiotto et al., 2013) and (Duvenaud et al., 2015) for fingerprinting of chemical molecules.

Two related approaches that learn multi-agent interactions on a graph structure are: Interaction Networks (Battaglia et al., 2016) which learn a physical simulation of objects that exhibit binary relations and Communication Networks (CommNets) (Sukhbaatar et al., 2016), presented for learning optimal communications between agents.

There are several models that have been proposed to learn multi-agent cooperation by communication. BiCNet (Peng et al., 2017) uses a reccurent neural network (RNN) as the communication channel to connect each individual agent's policy and value networks. ATOC (Jiang & Lu, 2018) enables agents to learn dynamic communication with nearby agents using attention mechanism.

Most existing models are limited to the scale of dozens of agents, while some consider large-scale MARL. When the number of agents increases, learning becomes hard due to the curse of the dimensionality and the exponential growth of agent interactions. Instead of considering the different effects of other individuals on each agent, Mean Field (Yang et al., 2018) approximates the effect of other individuals by their mean action. However, the mean action eliminates the difference among these agents in terms of observation and action and thus incurs the loss of important information that helps cooperative decision making

3. Background and Preliminaries

3.1. Graph Networks

In the Graph Network framework (Wang et al., 2018), (Battaglia et al., 2016), (Sanchez-Gonzalez et al., 2018) all objects in the environment can be represented as nodes of a directed graph, while edges define the relation between any two objects. As the system changes, whether due to external effects or control actions taken by the agent, the next state can be described through the interaction of all objects. When this happens, the node and edge values are updated based on how each node interacts with each other. This interaction is modeled as messages being passed, either coming in from neighboring nodes to a given node or vice versa. Aggregating this message passing for a fixed number of steps allows for the influences of each node to propagate throughout the graph.

Specifically, the following updates are applied in a GN block:

$$e_k^{'} = \phi^e(e_k, v_{r_k}, v_{s_k}, u) \tag{1}$$

$$v_i^{'} = \phi^v(\bar{e}_i^{'}, v_i, u) \tag{2}$$

$$u' = \phi^u(\bar{e}', \bar{v}') \tag{3}$$

, where (u, v, e) are the global attributes of the system, nodes and edges respectively in the defined graph structure. x' for any variable x here denotes its updated value after computation through update functions ϕ, while $\bar{x}'$ denotes the aggregation of the updated values across all nodes.

Graph networks help provide a relational inductive bias in terms of invariance to ordering of entities in an environment as well as the explicit physical connections between them. For instance, in the case of predicting the center of mass of a system of objects $x_1, x_2, ..., x_n$, a standard MLP will make different predictions for each different permutation of the input, i.e. the prediction for $(x_1, x_2, ..., x_n)$ as input will be different from the prediction for the input $(x_2, x_1, ..., x_n)$. This is not ideal as the sequencing does not have an effect on the prediction in this case. Using graphs to represent data allows us to preserve such structure in multi-agent systems.

3.2. Deep Deterministic Policy Gradients (DDPG)

DDPG (**?**) is an off policy, model free actor critic based reinforcement learning method. The critic is used to estimate the action value function $Q(s_t, a_t)$, while the actor refers to the deterministic policy of the agent. The critic is learned by minimizing the standard TD error

$$\delta_{TD} = r_t + \gamma Q'(s_{t+1}, \pi(s_{t+1})) \; - \; Q(s_t, a_t) \tag{4}$$

,where Q' refers to a target network (Mnih et al., 2015) which is updated after a fixed number of time steps. The actor is optimized by following the gradient of the critic's estimate of the Q value.

3.3. Multi-Agent Deep Deterministic Policy Gradients (MADDPG)

MADDPG (Lowe et al., 2017) is an extension to the standard DDPG setup for multi-agent systems. The basic idea is to have multiple actors $\pi_i(a_i|o_i)$ and critics $Q_i(o_i, a_1, a_2, ..., a_N)$ for each agent while giving actions of all other agents as input to the critic of a given agent. The insight behind such a scheme is that given the actions of all other agents, the environment is stationary for a single agent. All critics are trained independently, allowing for different rewarding schemes for each agent. Each actor only takes the observation of the current agent as input and gives its action as output. This is done so that while evaluation, the actors can be run independently without requiring any mixing of inputs.

4. Methodology

In this work, we propose learning to predict states or actions using Graph Networks and using the learnt latent embedding from such a GN module as input to the critic network of each agent in the DDPG setup (See Figure 1). The rationale is that once action prediction has succeeded, the information provided by giving all actions as input in the MADDPG case can be substituted by a fixed size representation learnt which inherently provides the required relational structure to learn optimal critic values. Moreover, such a scheme scales easily with an increased number of agents as well since the latent representation is of a fixed size. On the other hand, MADDPG's performance will hurt as the number of agents increase since the number of inputs to each critic network multiplies fast. We further analyze the effect of using GNs as compared to MLP based networks in terms of prediction and control performance. This is to understand what benefit GNs provide in the case all-to-all relations are maintained.

4.1. Graph Network Module

For our Graph Network prediction, we borrow the architecture (Figure 2) from () and use a MLP based encoder block which is concatenated with a hidden graph representation from time $t-1$. This concatenated input is fed into a GN block which outputs a hidden graph representation for time t and is fed as input to another MLP based decoder block. Figure 2 describes this in detail. We use this hidden representation as the embedding described above for learning using the DDPG setup.

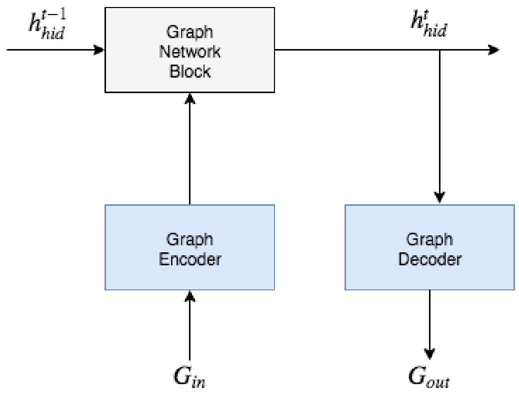

Figure 2. Graph Network Module architecture

4.2. State Prediction

We use the current state s_t and action a_t to predict the next state s_{t+1} of a set of agents interacting between each other. The action used here is either from a random policy or an expert policy (more on this below). We report one-step

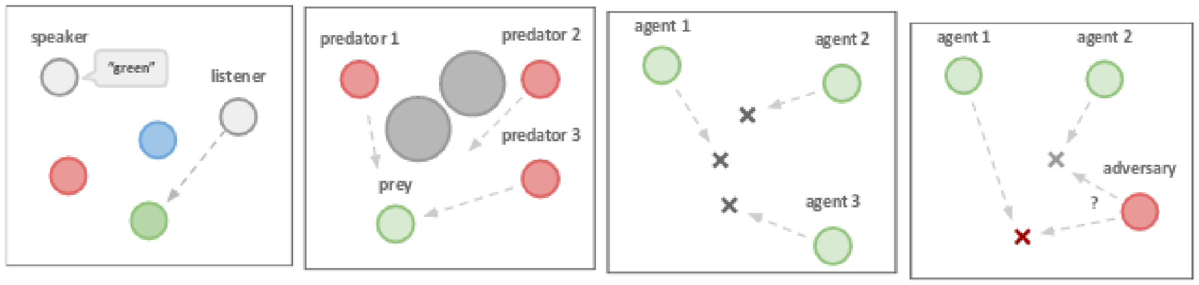

Figure 3. Few of the Multi-Agent Environment Tasks : (left-to-right) Cooperative Communication, Predator-Prey, Cooperative Navigation, Physical Deception

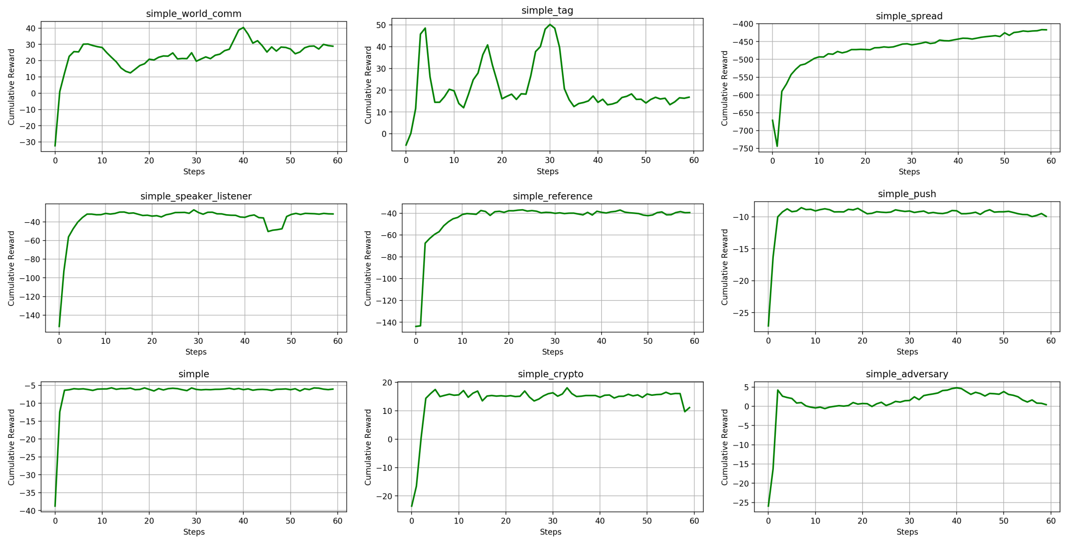

Figure 4. MADDPG Performance on Multi-Agent Environments

performance as well as recursive prediction performance. The latter involves using the state prediction at each step to make successive predictions.

4.3. Action Prediction

We use the current state s_t to predict the action a_t taken for a set of agents interacting between each other. The target action labels used here are from an expert policy. Using a random policy in this case does not rationalize well as learning to predict random behavior does not involve exploiting the agent states or the interaction between them.

5. Experiments

5.1. Environments

We consider recently released continuous state-action multi-agent environments by OpenAI (Figure 3). The tasks here exhibit the need for cooperative as well as competitive behavior. For instance, the Cooperative Communication task requires a speaker agent to learn to output the right color among its observations while the other listener agent is required to learn to go to the output action color of the speaker. Similarly, one of the competitive tasks involves a single agent, an adversary and a single landmark location. The adversary is rewarded based on how close it is to the landmark and how far the agent is to the landmark, whereas the agent is solely rewarded based on its distance from the landmark. Such a task induces an optimal behavior where the adversary learns to push the agent away from the landmark.

We first analyze the performance of MADDPG on all such environments. Figure 4 reports the learning curves generated respectively. Overall, we observe two cases where these tasks suite the MADDPG setup specifically and do not provide a fair comparison when using baselines like DDPG on them :

- **Action Sharing** Tasks such as Cooperative Communication explicitly require some information to be made available other agents to learn optimal behavior (without the action output of color given to the other agent, it does not have sufficient data to learn optimal behavior). A DDPG agent is inherently deprived of this information, leaving the comparison in performance unsatisfactory. We believe the actual divide can only be appreciated when all agents are provided with all required observations, and the difference arises due to the algorithmic design instead.

- **Observation Sharing** Tasks such as Physical Deception in which agents have to cover all landmarks so as to prevent an adversary to go to the one target landmark highlight the same issue above. The difference here is in terms of not providing all observations to all agents instead of actions.

Keeping the above two points in mind, we focus on tasks which do not explicitly hide useful information. Thus we consider the task of Cooperative Navigation which requires multiple agents to go to an equal number of landmarks without colliding with each other (negative reward). Such a task does not require explicit observation and action sharing, and thus we observe that DDPG does perform relatively well as compared to MADDPG. This begs the question as to how and if MADDPG actually improves performance on tasks that are solvable by single agent algorithms as well. The other major issue is in terms of scaling this to a decently large number of tasks. All the above introduced tasks involve three agents, with most working only with two. With the number of agents increased (to about 10), we observe that the training time scales almost linearly with the number of agents. This is because each added observation and action is added to the policy and state-action value networks for all agents.

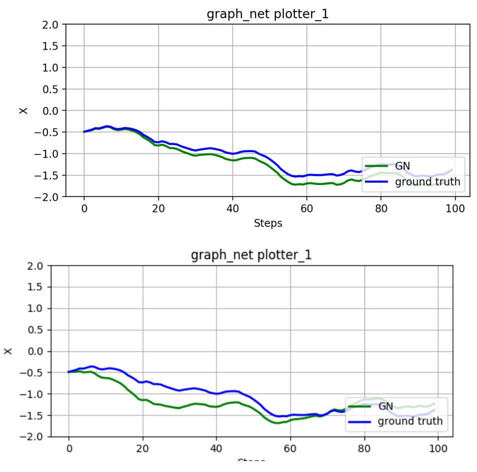

Figure 5. Recursive state (x coordinate value) prediction performance using Graph Network Module for two random runs

5.2. Training Details

For the GN prediction module, we use a 9 length vector for state prediction and a 4 length vector for action prediction. Each agent's action is a continuous valued vector of length 5, whereas for GN prediction we use only 4 observation values out of the available 18, namely the velocity in x direction, velocity in y direction, position in x coordinate, position in y coordinate. Edges in the graph connect to all agents in both directions. Therefore, a graph with three agents will have 6 edges, 2 between any two agents. In this case, we do not consider the global attributes of the graph and keep them as zero. In the recursive prediction, we provide the agent with the true velocities at each time step while the x and y position are recursively computed. We run for 100,000 iterations with a batch size of 200, using Adam optimizer with a learning rate of 0.001. After every 100 iterations new trajectories are collected and learning is performed over those. This is to ensure generalization in terms of experiencing varied state-action pairs. For the action prediction case, the expert policy is generated using MADDPG. The hidden layer size used for all cases is 8. This was finalized after a course grid search over the set of layer size values.

For the control module, we give all observations, the agent's action and the extracted embedding E as input to each agent's Q value and policy network. The rest of the details are as available at https://github.com/openai/maddpg.

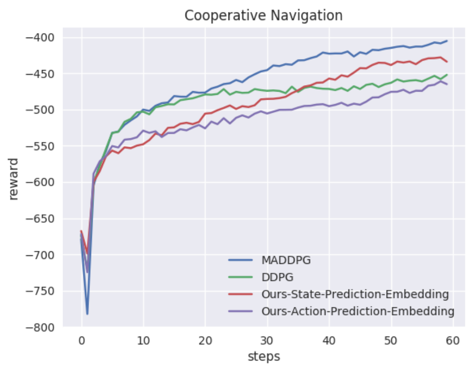

Figure 6. Reward plots for Cooperative Navigation task

5.3. Results

We begin by reporting state prediction performance on the Cooperative Navigation task with 3 agents and 10 agents. We see that using a simpler version of GNs (Interaction Networks) allows accurate recursive prediction when the number of agents are increased as compared to using standard MLPs (Figure 7). We further report prediction results using the Graph Network Module introduced in Section 4.1 (Figure 5). We see that the network is able to follow the true

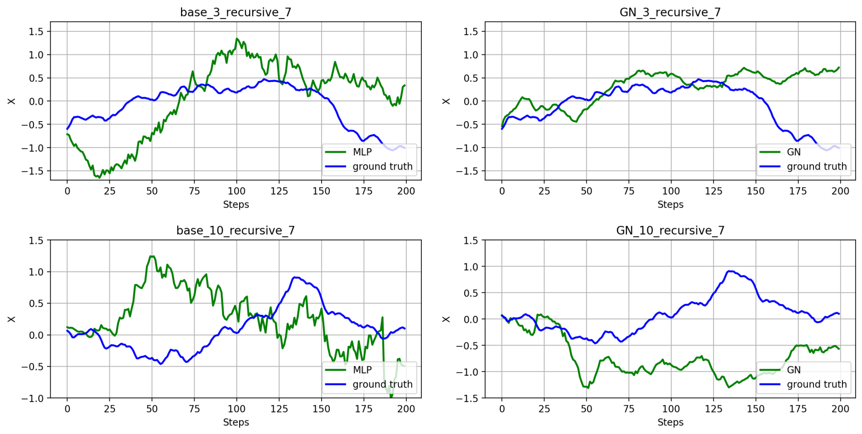

Figure 7. Recursive state prediction comparison for Interaction Network (right column) and a baseline MLP network (left column) with 3 (top row) and 10 agents (bottom row).

trajectory for about 5-20 steps while following the overall trend consistently across all 100 steps.

Further in our experiments, we try to find answers to the two questions raised in Section 5.1 by validating our approach for control using GNs against MADDPG. We observe that using a fixed size embedding for Q values of all agents produces similar performance without the burden of having individual observations and actions added (Figure 6). Moreover, since we use Graph Networks to learn the embedding using prediction tasks, it is able to capture the relational inductive bias present and exhibit invariance to input ordering. This is not the case when a standard MLP based embedding is used.

6. Discussion

Overall, we observe that the tasks considered might not be general enough to judge the performance of multi-agent algorithms accurately. Although enough good for initial sanity checks, such methods need to be tested on even more challenging environments which do not skew towards working well for a particular class of algorithms. Moreover, although we do prediction using Graph Networks, we never use it for any explicit model based learning procedure and only as a means for providing a compact representation. Perhaps this can be explored further, especially since the multi-step prediction for Graph Networks is essentially the most crucial step-up to standard MLP based predictions.

7. Conclusion and Future Work

Through our work, we have been able to answer the following question to a greater extent; Is it possible to design an algorithm that leverages the relational information between the agents and give it as a prior instead of the complete information?. The results have shown that the agents can achieve comparable or better performance than baseline single agent algorithm (DDPG) and multi-agent algorithm (MADDPG). We can conclude that agents therefore do not need complete information about their surroundings but some prior knowledge is sufficient. For example, a swarm of robots will find it difficult to process the complete information about every robot in the swarm. However, some prior knowledge that encapsulates this information can prove to be useful and computationally easy for the individual agents to process. We have been able to successfully demonstrate the performance of our algorithm on the multi-agent particle environment for a single task
Future work will see us experimenting with recurrent neural networks and sequential learning models that help alleviate the non-stationarity of the environment. Although, physical systems are theoretically Markov in nature, they can only be modelled accurately by treating them as partially observed environments and where past information plays a significant role in the forward prediction of the system. We also wish to experiment across different tasks and study in detail the effect of the prior knowledge in communication between agents.

References

Bansal, Trapit, Pachocki, Jakub, Sidor, Szymon, Sutskever, Ilya, and Mordatch, Igor. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

Battaglia, Peter, Pascanu, Razvan, Lai, Matthew, Rezende, Danilo Jimenez, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Duvenaud, David K, Maclaurin, Dougal, Iparraguirre, Jorge, Bombarell, Rafael, Hirzel, Timothy, Aspuru-Guzik, Alán, and Adams, Ryan P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

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

Gaiotto, Davide, Moore, Gregory W, and Neitzke, Andrew. Spectral networks. In *Annales Henri Poincaré*, volume 14, pp. 1643–1731. Springer, 2013.

Hoshen, Yedid. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*, pp. 2701–2711, 2017.

Hughes, Edward, Leibo, Joel Z, Philips, Matthew G, Tuyls, Karl, Duéñez-Guzmán, Edgar A, Castañeda, Antonio García, Dunning, Iain, Zhu, Tina, McKee, Kevin R, Koster, Raphael, et al. Inequity aversion resolves intertemporal social dilemmas. *arXiv preprint arXiv:1803.08884*, 2018.

Jaderberg, Max, Czarnecki, Wojciech M, Dunning, Iain, Marris, Luke, Lever, Guy, Castaneda, Antonio Garcia, Beattie, Charles, Rabinowitz, Neil C, Morcos, Ari S, Ruderman, Avraham, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

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

Jiang, Jiechuan, Dun, Chen, and Lu, Zongqing. Graph convolutional reinforcement learning for multi-agent cooperation, 2018.

Kaiser, Łukasz and Sutskever, Ilya. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

Lanctot, Marc, Zambaldi, Vinicius, Gruslys, Audrunas, Lazaridou, Angeliki, Perolat, Julien, Silver, David, Graepel, Thore, et al. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4190–4203, 2017.

Leibo, Joel Z, Zambaldi, Vinicius, Lanctot, Marc, Marecki, Janusz, and Graepel, Thore. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

Levine, Sergey, Pastor, Peter, Krizhevsky, Alex, Ibarz, Julian, and Quillen, Deirdre. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

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

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

Mnih, Volodymyr, Heess, Nicolas, Graves, Alex, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

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

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

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

Peysakhovich, Alexander and Lerer, Adam. Consequentialist conditional cooperation in social dilemmas with imperfect information. *arXiv preprint arXiv:1710.06975*, 2017.

Sanchez-Gonzalez, Alvaro, Heess, Nicolas, Springenberg, Jost Tobias, Merel, Josh, Riedmiller, Martin, Hadsell, Raia, and Battaglia, Peter. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018.

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

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

Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, Bolton, Adrian, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

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

Wang, Tingwu, Liao, Renjie, Ba, Jimmy, and Fidler, Sanja. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1sqHMZCb.

Yang, Yaodong, Luo, Rui, Li, Minne, Zhou, Ming, Zhang, Weinan, and Wang, Jun. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*, 2018.