Leader-based Pre-training Framework for Cooperative Multi-Agent Reinforcement Learning

Wenqi Chen [*1] **Xin Zeng** [*1] **Amber Li** [2]

Abstract

A leader in the team enables efficient learning for other novices in the social learning setting for both humans and animals. This paper constructs the leader-based pre-training framework for Multi-Agent Reinforcement Learning and investigates whether the leader enables the learning of novices as well. We compare three different approaches to distilling a leader's experiences from the pre-training model: Linear Layer Dimension Reduction, Attentive Graph Pooling, and Attention-based Graph Neural Network. We successfully show that a leader-based pre-training framework can 1) enable agents to learn faster, cooperate more effectively, and escape local optimum, and 2) promote the generalizability of agents in more challenging and unseen environments. The key to effective distillation is to maintain and aggregate important information.

1. Introduction

Social learning, learning by observing the behavior of other agents in the same environment, is a key component of human and animal intelligence (Ndousse et al., 2020), enabling the agents to learn behaviors difficult to learn by individual exploration. In particular, it is believed that the leader in the team encourages cooperation, such that agents tend to learn faster and cooperate more efficiently through obtaining experience from the leader about key changes in the environment. For example, there is always an experienced and elder lead dog on any team of Alaskan Malamute sled dogs, where the leader provides key guidance to other dogs.

We ask whether such behavior will also emerge in Multi-Agent Reinforcement Learning (MARL) and whether this paradigm can inspire us to construct a successful pre-training framework to improve the performance and generalizability of MARL agents? Our paper investigates leader-based social learning in the context of MARL: specifically, improving the performance of a group of novice agents (i.e. not trained in the environment) through gaining the experience of a leader agent (i.e. pre-trained in the environment).

Past research work of RL shows interest in related problems. Imitation learning of RL resembles the copying behaviors, such as learning from demonstrated trajectories, (Schaal, 1999; Argall et al., 2009), behavior cloning (BC) (Torabi et al., 2018) and BC combined with RL fine-tuning (Lerer & Peysakhovich, 2019). However, BC has limitations on generalizability because the knowledge of pre-training is distilled directly to all the agents. Furthermore, Ndousse et al. 2020 explored emergent social learning in RL agents and proposed that agents trained alone taking cues from the behavior of experts could learn to perform better and generalize to more complex environments. Our work is inspired by their work, but we differ in that their experts are not assumed to be a leader in multi-agent settings, and their expert does not train and adapt to the environment alongside the other agents. We also do not use a model-based auxiliary loss and instead focus on pre-trained knowledge transfer from leader to the novice.

We are also influenced by the work of Baker et al. 2019, which worked with MARL and showed that agents may create a self-supervised autocurricula through rounds of interactions. Literature on agents learning to teach in cooperative multi-agent environments (Omidshafiei et al., 2018) and agents learning by watching another agent learn (Jacq et al., 2019) also shed light on the teacher-student pre-trained framework but lacked generalizability.

Contributions. Our paper focuses on two hypotheses: Leader-based learning enables the agents to 1) out-perform agents trained alone from the start due to enhanced cooperation; 2) generalize faster to novel and more complicated environments. These hypotheses are representative of real-world problems such as autonomous driving and robot cooperation. Our contribution is proposing a successful leader-based pre-training framework for MARL which validates

[*]Equal contribution [1]Institute for Applied Computational Science, Harvard University, Cambridge, US [2]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, US. Correspondence to: Xin Zeng <xinzeng@fas.harvard.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

our hypotheses. Through the comparison between three different methods to distill a leader's experience including Linear Layer Dimension Reduction, Attentive Graph Pooling, and Attention-based Graph Neural Network, we provide evidence that our pre-training framework scales better and leads to behavior that centers around more human-relevant skills than other MARL methods such as Multi-Agent Deep Deterministic Policy Gradient (MADDPG).

2. Problem Background

In this work, we consider the setup of a partially observable Markov game. The game has N agents, a set of states S describing all configurations of all the agents, and a set of actions $A_1; : : : ; A_N$ and a set of observations $O_1; : : : ; O_N$ for each agent. Each agent i receives an observation that consists of partial information from the state $o_i : S \; ! \; O_i$ and uses its policy $\;_i : O_i \; ! \; A_i$ to choose its action, where the policy is parameterized by $\;_i$: For convenience, we may abbreviate $\;_i$ as $\;_i$: The next state comes from the state transition function $T : S \quad A_1 \qquad A_N \; ! \; S$: Each agent i receives reward $r_i : S \quad A_1 \qquad A_N \; ! \; R$ as a function of the state and the agents' actions. The objective for each agent (in our work, each adversarial agent) is to maximize its total expected return $R_i = \sum_{t=0}^{T} \; ^t r_i^t$; where is the discount factor and T is the time horizon.

Specifically, we use the multi-agent particle environment called *Simple Tag* (Lowe et al., 2017). This is a "predator-prey" environment, where the good agents move faster and want to avoid being hit by the adversaries, which are slower and must cooperate to effectively chase and tag the good agents. A "tag" happens when the agents collide. The environment also contains obstacles. The good agents move randomly, while the adversaries are controlled by us. The goal is for the adversaries to accumulate as many points as possible by tagging the good agents. Each episode terminates after a maximum number of steps, which we set to be 100.

Figure 1 visualizes the default environment, with the good agent shown in green, three adversarial agents shown in blue, and two obstacles depicted by large black circles. The agents' observations include its own position and velocity and the relative positions of others, obstacles. The adversarial agents select their actions in a decentralized manner but may learn to cooperate based on each their observations of each other in the world.

3. Methods

To have social learning with a leader agent and novice agents, we implement a MARL algorithm on the novice agents and transfer experience from the leader (an agent that has been pre-trained) to them. We describe our baseline algorithm and then the exact training procedures we used to transfer experience from leaders to novices in this section.

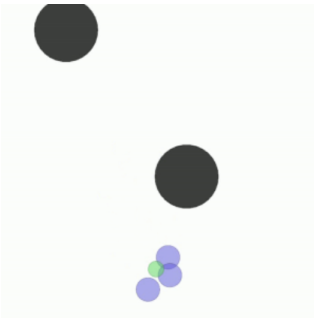


Figure 1: Simple Tag with 1 Good Agent and 3 Adversarial Agents

3.1. Baseline Model: MADDPG

MADDPG is a method that deploys a centralized action-value function within a Deep Deterministic Policy Gradient (DDPG) algorithm (Lowe et al., 2017). MADDPG is developed as a framework of centralized training and decentralized execution enabled by actor-critic policy gradient methods, which is also flexible enough for us to implement our leader-novice-style training. We choose MADDPG as our baseline for the following reasons: 1) The centralized critic and decentralized actor could potentially overcome the most drawbacks of MARL; 2) The critic and actor networks in MADDPG are flexible, which enables us to conduct the experience distillation from the leader; 3) MADDPG is widely used as a baseline in MARL research.

3.2. Leader Experience Distillation

Based on the current MARL model, we propose a leader-based framework to distill leader experience to the team of agents. It involves two steps: 1) train a leader in the simplest version of an environment with only one good, one adversarial agent; 2) in the new environment, initialize one controlled agent with the parameters from the trained leader. In this way, one of the agents in the new environment will become the leader of the team and be trained with other agents. We only initialize one of the agents with the parameters of the leader for the following reasons: 1) initializing only one agent makes other agents learn to cooperate instead mimic the leader, and gives other agents more flexibility to explore what the leader could not do; 2) we want to evaluate how our model works in the most limited condition where we only have the resources to get one leader.

The main challenge to overcome in the process of distilling experience from the leader is that going from an environment with two agents to an environment with $N > 2$ total agents increases the size of the inputs to the actor and critic

models. For the actor, when increasing the number of agents, the dimension in observation space increases in the degree of addition because we observe more agents. For the critic, since the centralized critic takes as input the concatenated actions and observation spaces of all agents, this increases the input dimension in the degree of multiplication. Figure 2 illustrates our model structure. We propose three methods to compress the information from multiple agents into the dimensions of a single agent so that we can distill the leader's experiences.

Linear Layer Dimension Reduction (LLDR) We firstly propose to add a linear layer in the actor network as the embedding layer for dimension reduction. Considering the drastic change in the dimension of the input to the critic, we do not conduct dimension reduction or parameter transfer in the critic network to prevent the loss of information.

Attentive Graph Pooling (AGP) We then propose a method to solve the dimension increase problem in the critic network. By representing the input of each agent as a node with fixed-sized embedding in a fully connected graph, we could pool several vertices to a single vertex embedding through AGP. In detail, this is done by creating attention heads, using leaky ReLU activation, and using linear layers to reduce dimension. In this way, we can obtain a representation with consistent feature dimensions for different numbers of agents as the input to the critic network.

Attention-Based Graph Neural Network (AGNN) We further propose another method to help effectively conduct dimension reduction in critic network. Instead of directly feeding the input of each agent into the layers for AGP, a more sophisticated method is to first use the AGNN to learn graph embeddings and then apply AGP to process the learned graph embeddings. In this way, the graph is represented as a fixed-sized embedding by pooling features across all vertices, so we can obtain a representation with consistent feature dimensions for different numbers of agents.

Motivated by Li et al. 2019, which applied AGNN to transfer knowledge between tasks with a different number of objects, we apply a similar AGNN structure to transfer knowledge from the leader to the novice agents. The AGNN architecture is able to capture which neighbors are more relevant to classifying a target node, which is critical in our case where not all edges imply the same types or strengths of relations. Through the weighted sum operation, we could get a well-represented embedding capturing the agents' relations, so that they can learn to cooperate more effectively.

The structure of our attention-based GNN model follows similar structure as the one proposed by Li et al. 2019. Mathematically, assuming $\vec{v}_i^t$ is the graph feature representation of the i-th agent at timestep t. The representation at timestep $t+1$ could be computed through sum operation, $\vec{v}_i^{t+1} = \sum_j w_{ij} \vec{m}_j^t$. In this expression, $\vec{m}_j^t$ is generated from a parameterized function $\vec{m}_j^t = f_m^t(\vec{v}_j^t)$, and w_{ij} is computed through $w_{ij} = (V^T \tanh(\vec{q}_j^t + \vec{k}_j^t))$, where $\vec{q}_j^t$ and $\vec{k}_j^t$ are generated from independent parameterized functions $\vec{q}_j^t = f_q^t(\vec{v}_i^t)$ and $\vec{k}_j^t = f_k^t(\vec{v}_i^t)$.

4. Experimental Results

We conduct three sets of experiments to compare the performance and generalizability of the proposed approaches against a baseline and to validate our hypotheses.

4.1. Leader-based Learning with Different Approaches

We begin with the environment using the default settings, where we control 3 adversarial agents to catch 1 good agent, which moves 1.3 faster. By default, there are 2 obstacles. We train the baseline and proposed approaches for 1M episodes and evaluate performance every 1K episodes.

Figure 3 shows the performance of the baseline and our three approaches. We see that LLDR and AGNN methods outperform the baseline, while the AGP method performs worse. Although the baseline converges relatively quickly, it gets stuck at a local optimum after 400K episodes. We verified by visualizing the agents that this is caused by ineffective cooperation. The LLDR method converges quicker and reaches a higher reward. Although the AGNN method initially converges slowly due to the relatively large amount of parameters being learned, it ultimately reaches a higher reward and outperforms the baseline. Through visualizing the agents' trajectories, we observe that generally, the three adversarial agents trained using the baseline struggle to cooperate, with adversarial agents making little effort to work together to corral the good agent. However, adversarial agents trained through LLDR and AGNN methods cooperate better, catching the good agent by surrounding it from different directions.

This experiment validates our hypothesis that leader-based MARL can enable agents to learn more quickly the skills that are difficult to learn through individual exploration. The presence of the leader also appears to encourage cooperation. However, the distillation methods for compressing experience matter significantly. LLDR and AGNN approaches worked well, indicating that the embedding layer should not lose too much information. This makes sense because AGNN can support "communication" among agents through message passing on the graph. We also experimented with applying the linear layer in both actor and critic networks, instead of just the actor network, and the resulting performance was similarly poor to results using AGP. All of this suggests that the embedding layer must be utilized in a way

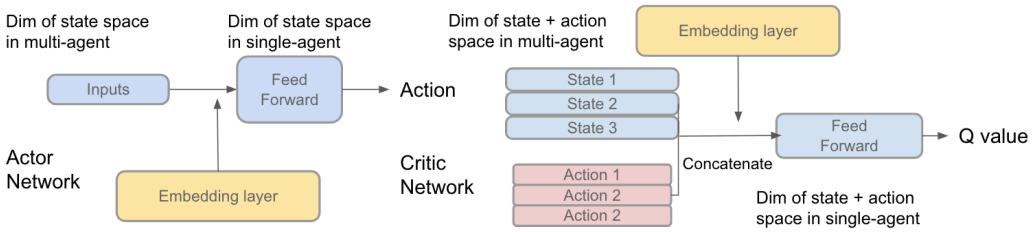


Figure 2: Model Structure of Leader Experience Distillation. It is based on the structure of Multi-agent Actor-Critic with an additional embedding layer for dimension compression.

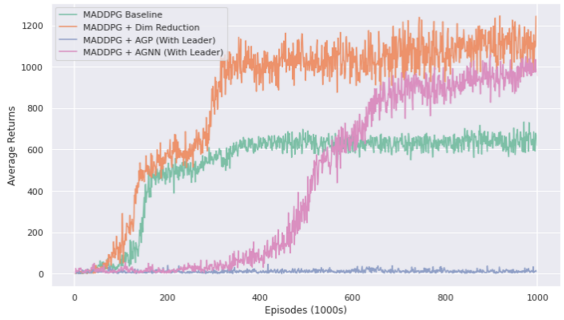


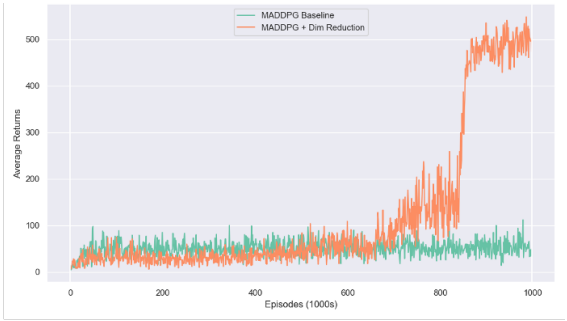
Figure 3: Experiments with different approaches in the default settings. Effective leader-based learning enables agents to learn faster, cooperate more effectively, and escape the local optimum compared with baseline.

to maintain information between agents.

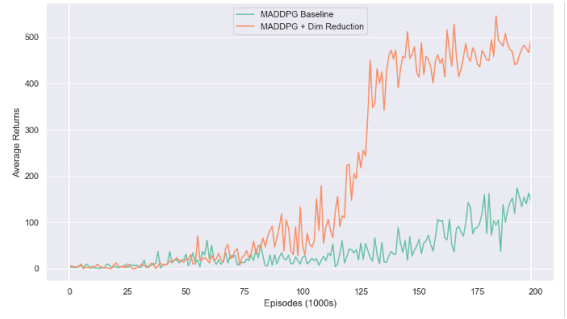
4.2. Leader-based Learning with Linear Layer in New Environment

Because of the satisfactory performance of the LLDR method in the default environment, we experiment in two environments that are far different from and more complicated than the environment in which the leader is trained. The first has 3 good and 3 adversarial agents, which makes cooperation more challenging since more goals tend to distract the focus of adversarial agents. We train the baseline and LLDR approaches for 1M episodes and evaluate their performance every 1K episodes. The second more challenging environment has 1 good, 3 adversarial agents, and 3 obstacles. Additionally, the good agents move 1.8 faster than the default 1.3 . We train the baseline and LLDR approaches for 200K episodes and evaluate their performance every 1K episodes.

Figure 4 shows the performance. We see that the LLDR method strongly outperforms the baseline method, which struggles to learn a good cooperation strategy. This again validates our hypothesis that leader-based learning can enable the agents to learn faster in challenging environments.



(a) 3 Good & 3 Adversarial Agents, 2 Obstacles, and Normal Speed of Good Agents



(b) 1 Good & 3 Adv Agents, 3 Obstacles, and Quicker Speed of Good Agents

Figure 4: Experiments with leader teaching in new environment. Leader-based learning with LLDR enables agents to learn faster than the agents trained individually.

4.3. Leader-based Learning in Few-Shot Learning

Finally, we evaluate the baseline and three proposed methods in a new environment with 1 good and 3 adversarial agents, where the good agent moves 1.8 the speed of the adversarial agents. Instead of initializing a single adversarial agent with the parameters of a leader trained in an environment with 1 good and 1 adversarial agent as in the previous experiments, we utilize few-shot learning: we initialize three adversarial agents using the parameters of adversaries trained in the environment with 1 good and 3 adversarial agents as in the experiments using the corresponding methods (see Section 4.1). Figure 5 shows the performance of

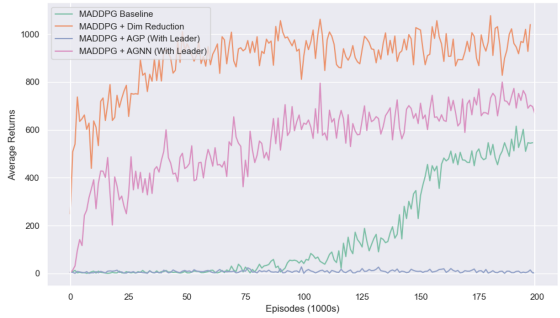


Figure 5: Experiments with Few-Shot Learning to validate generalization of agents from Leader-based Learning. Effective leader-based learning enables better generalization in more challenging unseen environments.

the baseline and our three approaches. We see that LLDR and AGNN methods outperform the baseline once again. LLDR performs the best, although AGNN may be slower to converge due to the larger model size. This result also validates our second hypothesis that leader-based learning would lead to more effective generalization. The agents trained with a leader not only learn the abilities within the trained environment but also possesses better cooperation and learning capabilities to generalize in new challenging environments.

5. Conclusion

In this work, we propose a new leader-based pre-training framework for MARL inspired by social learning. We design three approaches, LLDR, AGP, and AGNN, to distill a leader's experience into novice agents. We successfully show that leader-based learning enables agents to 1) learn faster, cooperate more effectively, and escape local optimum compared to agents trained individually, and 2) generalize easily in more challenging, unseen environments. Additionally, we notice that the embedding layer plays an important role in experience distillation, such that only the methods that perform compression without the loss of important information allow for effective leader-based learning.

Acknowledgements

The authors would like to thank Professor Pulkit Agrawal for valuable suggestions and helpful conversations during the course of this project.

References

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.10.024. URL https://www.sciencedirect.com/science/article/pii/S0921889008001772.

Baker, B., Kanitscheider, I., Markov, T. M., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. Emergent tool use from multi-agent autocurricula. *CoRR*, abs/1909.07528, 2019. URL http://arxiv.org/abs/1909.07528.

Jacq, A., Geist, M., Paiva, A., and Pietquin, O. Learning from a learner. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2990–2999. PMLR, 09–15 Jun 2019.

Lerer, A. and Peysakhovich, A. Learning existing social conventions via observationally augmented self-play. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, pp. 107–114, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450363242. doi: 10.1145/3306618.3314268. URL https://doi.org/10.1145/3306618.3314268.

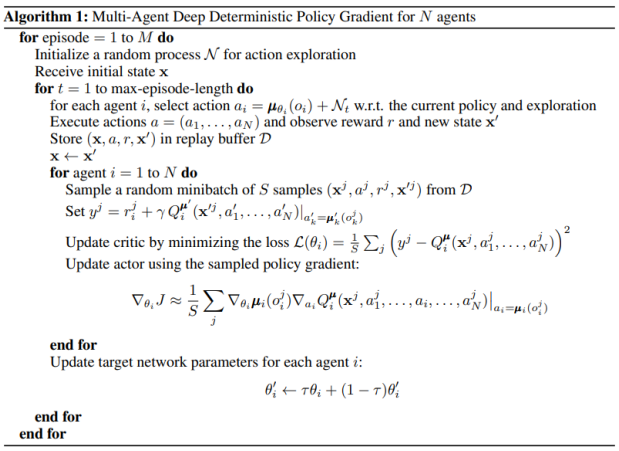
Li, R., Jabri, A., Darrell, T., and Agrawal, P. Towards practical multi-object manipulation using relational reinforcement learning. *CoRR*, abs/1912.11032, 2019. URL http://arxiv.org/abs/1912.11032.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Ndousse, K., Eck, D., Levine, S., and Jaques, N. Emergent social learning via multi-agent reinforcement learning, 2020. URL https://arxiv.org/abs/2010.00581.

Omidshafiei, S., Kim, D., Liu, M., Tesauro, G., Riemer, M., Amato, C., Campbell, M., and How, J. P. Learning to teach in cooperative multiagent reinforcement learning. *CoRR*, abs/1805.07830, 2018. URL http://arxiv.org/abs/1805.07830.

Schaal, S. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(99)01327-3.



Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i} \boldsymbol{\mu}_i(o_i^j) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

  end for
end for

Figure 6: Full MADDPG Algorithm

URL https://www.sciencedirect.com/science/article/pii/S1364661399013273.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pp. 4950–4957. AAAI Press, 2018. ISBN 9780999241127.

A. Appendix

A.1. Details of MADDPG

For completeness, the full MADDPG algorithm is included in Figure 6 (Lowe et al., 2017).

In particular, we use the extension of MADDPG that works with deterministic policies. Given N agents with continuous policies $= f \ _1;:::; \ _N g$ parameterized by $= f \ 1;:::; \ N g;$ let $J(\ i)$ be the expected return for adversarial agent i. We abbreviate each $\ i$ as $\ i$: Then the gradient of the expected return is

$$r \ _i J(\ i) = \mathsf{E}_{x;a \ D}[r \ _i \log \ _i(a_i j o_i) \quad r \ a_i Q_i \ (o_i; a_1; :::; a_N) j_{a_i = \ i(o_i)}]; \tag{1}$$

where x is the state, and experience replay buffer D holds transition data experienced by all agents in the form $(o; a; r; o^0)$. The $Q_i \ (o_i; a_1; :::; a_N)$ is the centralized action-value function (critic) that takes in an observation from agent i and the actions of all agents in order to compute the Q-value for agent i. We update Q_i by minimizing the loss given by

$$L(\ i) = \mathsf{E}_{o;a;r;o^0} \ \left(Q_i \ (o_i; a_1; :::; a_N) \quad y\right)^2 \ ;$$

where the target is

$$y = r_i + \ Q_i \ ^0(o^0; a_1^0; :::; a_N^0) j_{a_j^0 = \ j(o_j)};$$

and we use a set of target policies $^0 = f \ _1^0;:::; \ _N^0 g$ parameterized by ^0_i.