Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient

Shihui Li[†] **Yi Wu**[‡] **Xinyue Cui**[§] **Honghua Dong**[§] **Fei Fang**[†] **Stuart Russell**[‡]

[†] Carnegie Mellon University {shihuil@cs.,feifang@}cmu.edu

[‡] University of California, Berkeley {jxwuyi,russell}@eecs.berkeley.edu

[§] Tsinghua University {cuixy14,dhh14}@mails.tsinghua.edu.cn

Abstract

Despite the recent advances of deep reinforcement learning (DRL), agents trained by DRL tend to be brittle and sensitive to the training environment, especially in the multi-agent scenarios. In the multi-agent setting, a DRL agent's policy can easily get stuck in a poor local optima w.r.t. its training partners – the learned policy may be only locally optimal to other agents' current policies. In this paper, we focus on the problem of training robust DRL agents with continuous actions in the multi-agent learning setting so that the trained agents can still generalize when its opponents' policies alter. To tackle this problem, we proposed a new algorithm, *MiniMax Multi-agent Deep Deterministic Policy Gradient (M3DDPG)* with the following contributions: (1) we introduce a minimax extension of the popular multi-agent deep deterministic policy gradient algorithm (MADDPG), for robust policy learning; (2) since the continuous action space leads to computational intractability in our minimax learning objective, we propose *Multi-Agent Adversarial Learning (MAAL)* to efficiently solve our proposed formulation. We empirically evaluate our M3DDPG algorithm in four mixed cooperative and competitive multi-agent environments and the agents trained by our method significantly outperforms existing baselines.

1. Introduction

Most real-world problems involve interactions between multiple agents and the complexity of problem increases significantly when the agents co-evolve together. Thanks to the recent advances of deep reinforcement learing (DRL) on single agent scenarios, which led to successes in playing Atari game (Mnih et al. 2015), playing go (Silver et al. 2016) and robotics control (Levine et al. 2016), it has been a rising trend to adapt single agent DRL algorithms to multi-agent learning scenarios and many works have shown great successes on a variety of problems, including automatic discovery of communication and language (Sukhbaatar, Fergus, and others 2016; Mordatch and Abbeel 2017), multiplayer games (Peng et al. 2017a; OpenAI 2018), traffic control (Wu et al. 2017) and the analysis of social dilemmas (Leibo et al. 2017).

The critical challenge when adapting classical single agent DRL algorithms to multi-agent setting is the training instability issue: as training progresses, each agent's policy is changing and therefore the environment becomes non-stationary from the perspective of any individual agent (in a way that is not explainable by changes in the agent's own policy). This non-stationary problem can cause significant problems when directly applying the single agent DRL algorithms, for example, the variance of the policy gradient can be exponentially large when the number of agents increases (Lowe et al. 2017). To handle this instability issue, recent works, such as the counterfactual multi-agent policy gradients (Foerster et al. 2017) and the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al. 2017), proposed to utilized a *centralized critic* within the actor-critic learning framework to reduce the variance of policy gradient.

Despite the fact that using a centralized critic stabilizes training, the learned policies can still be brittle and sensitive to its training partners and converge to a poor local mode. This is particularly severe for competitive environments: when the opponents alter their policies during testing, the performance of the learned policies can be drastically worse (Lazaridou, Peysakhovich, and Baroni 2016). Hence, a *robust* policy becomes desirable in multi-agent setting: a well-trained agent should be able behave well in testing when competing against opponents even with strategies different from its training partners.

In this work, we focus on robust multi-agent reinforcement learning with continuous action spaces and propose a novel algorithm, *MiniMax Multi-agent Deep Deterministic Policy Gradient (M3DDPG)*. M3DDPG is a minimax extension[1] of the classical MADDPG algorithm (Lowe et al. 2017). Its core idea is that during training, we force each agent to behave well even when its training opponents response in the worst way.

Our major contributions are summarized as follow:

- We introduce the minimax approach to robust multi-agent DRL and propose a novel minimax learning objective based on the MADDPG algorithm;
- In order to efficiently optimize the minimax learning objective, we propose an end-to-end learning approach, *Multi-agent Adversarial Learning (MAAL)*, which is inspired by

[1] In fact, we are dealing with gains, i.e., maximizing each agent's accumulative reward, so the "minimax" here is essentially "maximin". We keep the term "minimax" to be consistent with literature.

the *adversarial training* (Goodfellow, Shlens, and Szegedy 2014) technique[2].

- We empirically evaluate our proposed M3DDPG algorithm on four mixed cooperative and competitive environments and the agents trained by M3DDPG outperform baseline policies on all these environments.

In the rest of the paper, we will firstly present related works in section 2. Notations and standard algorithms are described in section 3. Our main algorithm, M3DDPG, is introduced in section 4. Experimental results are in section 5.

2. Related Work

Multi-agent reinforcement learning (Littman 1994) has been a long-standing field in AI (Hu, Wellman, and others 1998; Busoniu, Babuska, and De Schutter 2008). Recent works in DRL use deep neural networks to approximately represent policy and value functions. Inspired by the success of DRL in single-agent settings, many DRL-based multi-agent learning algorithms have been proposed. Forester et al. (2016b) and He et al. (2016) extended the deep Q-learning to multi-agent setting; Peng et al. (2017a) proposed a centralized policy learning algorithm based on actor-critic policy gradient; Forester et al. (2016a) developed a decentralized multi-agent policy gradient algorithm with centralized baseline; Lowe et al. (2017) extended DDPG to multi-agent setting with a centralized Q function; Wei et al. (2018) and Grau-Moya (2018) proposed multi-agent variants of the soft-Q-learning algorithm (Haarnoja et al. 2017); Yang et al. (2018) focused on multi-agent reinforcement learning on a very large population of agents. Our M3DDPG algorithm is built on top of MADDPG and inherits the decentralized policy and centralized critic framework.

Minimax is a fundamental concept in game theory and can be applied to general decision-making under uncertainty, prescribing a strategy that minimizes the possible loss for a worst case scenario (Osborne and others 2004). Minimax was firstly introduced to multi-agent reinforcement learning as minimax Q-learning by Littman (1994). More recently, some works combine the minimax framework and the DRL techniques to find Nash equilibrium in two player zero-sum games (Foerster et al. 2018; Pérolat et al. 2016; Grau-Moya, Leibfried, and Bou-Ammar 2018). In our work, we utilize the minimax idea for the purpose of robust policy learning.

Robust reinforcement learning was originally introduced by Morimoto et al. (2005) considering the generalization ability of the learned policy in the single-agent setting. This problem is also studied recently with deep neural networks, such as adding random noise to input (Tobin et al. 2017) or dynamics (Peng et al. 2017b) during training. Besides adding random noise, some other works implicitly adopt the minimax idea by utilizing the "worst noise" (Pinto et al. 2017; Mandlekar et al. 2017). These works force the learned policy to work well even under the worst case perturbations and are typically under the name of "adversarial reinforcement learning", despite the fact that the original adversarial reinforcement learning problem was introduced in the setting of multi-agent learning (Uther and Veloso 1997). In our M3DDPG algorithm, we focus on the problem of learning polices that is robust to opponents with different strategies.

Within the minimax framework, finding the worst case scenario is a critical component. Lanctot et al. (2017) proposed an iterative approach that alternatively computes the best response policy while fixes the other. Gao et al. (Gao, Mueller, and Hayward 2018) replace "mean" in the temporal difference learning rule with "minimum". In our work, we proposed MAAL, which is a general, efficient and fully end-to-end learning approach. MAAL is motivated by adversarial training (Goodfellow, Shlens, and Szegedy 2014) and suitable for arbitrary number of agents. The core idea of MAAL is approximating the minimization in our minmax objective by a single gradient descent step. The idea of one-step-gradient approximation was also explored in meta-learning (Finn, Abbeel, and Levine 2017).

3. Background and Preliminary

In this section, we describe our problem setting and the standard algorithms. Most of the definitions and notations follow the original MADDPG paper (Lowe et al. 2017).

Markov Games

We consider a multi-agent extension of Markov decision processes (MDPs) called partially observable Markov games (Littman 1994). A Markov game for N agents is defined by a set of states $\mathcal{S}$ describing the possible configurations of all agents, a set of actions $\mathcal{A}_1, ..., \mathcal{A}_N$ and a set of observations $\mathcal{O}_1, ..., \mathcal{O}_N$ for each agent. To choose actions, each agent i uses a stochastic policy $\boldsymbol{\pi}_{\theta_i} : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$ parameterized by θ_i, which produces the next state according to the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_N \mapsto \mathcal{S}$. Each agent i obtains rewards as a function of the state and agent's action $r_i : \mathcal{S} \times \mathcal{A}_i \mapsto \mathbb{R}$, and receives a private observation correlated with the state $\mathbf{o}_i : \mathcal{S} \mapsto \mathcal{O}_i$. The initial states are determined by a distribution $\rho : \mathcal{S} \mapsto [0, 1]$. Each agent i aims to maximize its own total expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ where γ is a discount factor and T is the time horizon.

To minimize notation, in the following discussion we will often omit θ from the subscript of $\boldsymbol{\pi}$.

Q-Learning and Deep Q-Networks (DQN)

Q-Learning and DQN (Mnih et al. 2015) are popular methods in reinforcement learning and have been previously applied to multi-agent settings (Foerster et al. 2016a; Tesauro 2004). Q-Learning makes use of an action-value function for policy $\boldsymbol{\pi}$ as $Q^{\boldsymbol{\pi}}(s, a) = \mathbb{E}[R|s^t = s, a^t = a]$. This Q function can be recursively rewritten as $Q^{\boldsymbol{\pi}}(s, a) = \mathbb{E}_{s'}[r(s, a) + \gamma \mathbb{E}_{a' \sim \boldsymbol{\pi}}[Q^{\boldsymbol{\pi}}(s', a')]]$. DQN learns the action-value function Q^* corresponding to the optimal policy by minimizing the loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(Q^*(s, a|\theta) - y)^2], \quad (1)$$
$$\text{where} \quad y = r + \gamma \max_{a'} \bar{Q}^*(s', a').$$

[2]The connection between MAAL and adversarial training will be discussed in details at the end of section 4.

$\bar{Q}$ is a target Q function whose parameters are periodically updated with the most recent θ, which helps stabilize learning. Another crucial component of stabilizing DQN is the use of an experience replay buffer $\mathcal{D}$ containing tuples (s, a, r, s'). Q-learning algorithm is most suitable for DRL agents with discrete action spaces.

Policy Gradient (PG) Algorithms

Policy gradient methods is another popular choice for a variety of RL tasks. Let ρ^π denote discounted state visitation distribution for a policy π. The main idea of PG is to directly adjust the parameters θ of the policy in order to maximize the objective $J(\theta) = \mathbb{E}_{s\sim\rho^{\boldsymbol{\pi}}, a\sim\boldsymbol{\pi}_\theta}[R]$ by taking steps in the direction of $\nabla_\theta J(\theta)$. Using the Q function defined previously, the gradient of the policy can be written as (Sutton et al. 2000):

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^{\boldsymbol{\pi}}, a\sim\boldsymbol{\pi}_\theta}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|s) Q^{\boldsymbol{\pi}}(s, a)], \quad (2)$$

where $p^{\boldsymbol{\pi}}$ is the state distribution. The policy gradient theorem has given rise to several practical algorithms, which often differ in how they estimate $Q^{\boldsymbol{\pi}}$. For example, one can simply use a sample return $R^t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, which leads to the REINFORCE algorithm (Williams 1992). Alternatively, one could learn an approximation of the true action-value function $Q^{\boldsymbol{\pi}}(s, a)$ called the *critic* and leads to a variety of *actor-critic* algorithms (Sutton and Barto 1998).

Deterministic Policy Gradient (DPG) Algorithms

DPG algorithms extends the policy gradient algorithm to deterministic policies $\boldsymbol{\mu}_\theta : \mathcal{S} \mapsto \mathcal{A}$ (Silver et al. 2014). In particular, under certain conditions we can write the gradient of the objective $J(\theta) = \mathbb{E}_{s\sim\rho^{\boldsymbol{\mu}}}[R(s, a)]$ as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\mathcal{D}}[\nabla_\theta \boldsymbol{\mu}_\theta(s) \nabla_a Q^{\boldsymbol{\mu}}(s, a)|_{a=\boldsymbol{\mu}_\theta(s)}], \quad (3)$$

where $\mathcal{D}$ is the replay buffer. Since this theorem relies on $\nabla_a Q^{\boldsymbol{\mu}}(s, a)$, it requires the action space $\mathcal{A}$ (and thus the policy $\boldsymbol{\mu}$) be continuous.

Deep deterministic policy gradient (DDPG) (Lillicrap et al. 2015) is a variant of DPG where the policy $\boldsymbol{\mu}$ and critic $Q^{\boldsymbol{\mu}}$ are approximated with deep neural networks. DDPG is an off-policy algorithm, and samples trajectories from a replay buffer of experiences that are stored throughout training. DDPG also makes use of a target network, as in DQN (Mnih et al. 2015).

Multi-Agent Deep Deterministic Policy Gradient

Directly applying single-agent RL algorithms to the multi-agent setting by treating other agents as part of the environment is problematic as the environment appears non-stationary from the view of any one agent, violating Markov assumptions required for convergence. Particularly, this non-stationary issue is more severe in the case of DRL with neural networks as function approximators. The core idea of the MADDPG algorithm (Lowe et al. 2017) is learning a centralized Q function for each agent which conditions on global information to alleviate the non-stationary problem and stabilize training.

More concretely, consider a game with N agents with policies parameterized by $\boldsymbol{\theta} = \{\theta_1, ..., \theta_N\}$, and let $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N\}$ be the set of all agents' policies. Then we can write the gradient of the expected return for agent i with policy $\boldsymbol{\mu}_i$, $J(\theta_i) = \mathbb{E}[R_i]$ as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathbf{x}, a\sim\mathcal{D}}[\nabla_{\theta_i} \boldsymbol{\mu}_i(o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, ..., a_N)|_{a_i=\boldsymbol{\mu}_i(o_i)}] \ , \quad (4)$$

Here $Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)$ is a *centralized action-value function* that takes as input the actions of all agents, $a_1, \ldots, a_N$, in addition to some state information $\mathbf{x}$ (i.e., $\mathbf{x} = (o_1, ..., o_N)$) , and outputs the Q-value for agent i. Let $\mathbf{x}'$ denote the next state from $\mathbf{x}$ after taking actions $a_1, \ldots, a_N$. The experience replay buffer $\mathcal{D}$ contains the tuples $(\mathbf{x}, \mathbf{x}', a_1, \ldots, a_N, r_1, \ldots, r_N)$, recording experiences of all agents. The centralized action-value function $Q_i^{\boldsymbol{\mu}}$ is updated as:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}'}[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2], \quad (5)$$
$$y = r_i + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a_1', \ldots, a_N')\big|_{a_j'=\boldsymbol{\mu}_j'(o_j)},$$

where $\boldsymbol{\mu}' = \{\boldsymbol{\mu}_{\theta_1'}, ..., \boldsymbol{\mu}_{\theta_N'}\}$ is the set of target policies with delayed parameters θ_i'.

Note that the centralized Q function is only used during training. During decentralized execution, each policy $\boldsymbol{\mu}_{\theta_i}$ only takes local information o_i to produce an action.

4. Minimax Multi-Agent Deep Deterministic Policy Gradient (M3DDPG)

In this section, we introduce our proposed new algorithm, *Minimax Multi-agent Deep Deterministic Policy Gradient (M3DDPG)*, which is built on top of the MADDPG algorithm and particularly designed to improve the *robustness* of learned policies. Our M3DDPG algorithm contains two major novel components:

Minimax Optimization Motivated by the minimax concept in game theory, we introduce minimax optimization into the learning objective;

Multi-Agent Adversarial Learning The continuous action space results in computational intractability issue when optimizing our proposed minimax objective. Hence, we propose *Multi-Agent Adversarial Learning (MAAL)* to solve this optimization problem.

Minimax Optimization

In multi-agent RL, the agents' policies can be very sensitive to their learning partner's policy. Particularly in competitive environments, the learned policies can be brittle when the opponents alter their strategies. For the purpose of learning robust policies, we propose to update policies considering the *worst situation*: during training, we optimize the accumulative reward for each agent i under the assumption that all other agents acts adversarially. This yields the minimax

learning objective $\max_{\theta_i} J_{\mathrm{M}}(\theta_i)$ where

$$J_{\mathrm{M}}(\theta_i) = \mathbb{E}_{s\sim\rho^{\boldsymbol{\mu}}}\left[R_i\right]$$
$$= \min_{a^t_{j\neq i}} \mathbb{E}_{s\sim\rho^{\boldsymbol{\mu}}}\left[\sum_{t=0}^{T}\gamma^t\, r_i(s^t, a^t_1, \ldots, a^t_N)\big|_{a^t_i=\boldsymbol{\mu}(o^t_i)}\right] \quad (6)$$
$$= \mathbb{E}_{s^0\sim\rho}\left[\min_{a^0_{j\neq i}} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(s^0, a^0_1, \ldots, a^0_N)\Big|_{a^0_i=\boldsymbol{\mu}(o^0_i)}\right]. \quad (7)$$

Critically, in Eq. 6, state s^{t+1} at time $t+1$ depends not only on the dynamics $\rho^{\boldsymbol{\mu}}$ and the action $\boldsymbol{\mu}_i(o^t_i)$ but also on all the previous adversarial actions $a^{t'}_{j\neq i}$ with $t' \leq t$. In Eq. 7, we derive the modified Q function $Q^{\boldsymbol{\mu}}_{\mathrm{M}}(s, a_1, \ldots, a_N)$, which is naturally centralized and can be rewritten in a recursive form

$$Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(s, a_1, \ldots, a_N) = r_i(s, a_1, \ldots, a_N) +$$
$$\gamma\mathbb{E}_{s'}\left[\min_{a'_{j\neq i}} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(s', a'_1, \ldots, a'_N)\Big|_{a'_i=\boldsymbol{\mu}_i(s')}\right]. \quad (8)$$

Importantly, $Q^{\boldsymbol{\mu}}_{\mathrm{M}}(s, a_1, \ldots, a_N)$ conditions on the current state s as well as the *current actions* $a_1, \ldots, a_N$ and represents the current reward plus the discounted worst case future return starting from the *next state*, s'. This definition brings the benefits that we can naturally apply off-policy temporal difference learning later to derive the update rule for $Q^{\boldsymbol{\mu}}_{\mathrm{M}}$.

Note that for each agent i, none of the adversarial actions depend on its parameter θ_i, so we can directly apply the deterministic policy gradient theorem to compute $\nabla_{\theta_i} J_{\mathrm{M}}(\theta_i)$ and use off-policy temporal difference to update the Q function. Thanks to the *centralized Q function* in MADDPG (Eq. 4), which takes in the actions from all the agents, our derivation naturally applies and is perfectly aligned with the MADDPG formulation (Eq. 4) by injecting a *minimization* over other agents' actions as follows:

$$\nabla_{\theta_i} J_{\mathrm{M}}(\theta_i) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\begin{array}{l}\nabla_{\theta_i}\boldsymbol{\mu}_i(o_i)\nabla_{a_i} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a^\star_1, \ldots, a_i, \ldots a^\star_N)\big| \\ \quad a_i = \boldsymbol{\mu}_i(o_i) \\ \quad a^\star_{j\neq i} = \arg\min_{a_{j\neq i}} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a_1, \ldots, a_N)\end{array}\right], \quad (9)$$

where $\mathcal{D}$ denotes the replay buffer and $\mathbf{x}$ denotes the state information.

Correspondingly, we obtain the new Q function update rule by adding another minimization to Eq. 5 when computing the target Q value:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'\sim\mathcal{D}}[(Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a_1, \ldots, a_N) - y)^2], \quad (10)$$
$$y = r_i + \gamma\, Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}', a'^\star{}_1, \ldots, a'_i, \ldots, a'^\star{}_N)$$
$$a'_i = \boldsymbol{\mu}'_i(o_i),$$
$$a'^\star{}_{j\neq i} = \arg\min_{a'_{j\neq i}} Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}', a'_1, \ldots, a'_N),$$

where $\boldsymbol{\mu}'_i$ denotes the target policy of agent i with delayed parameters θ'_i, and $Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}$ denotes the target Q network for agent i. Combining Eq. 9 and Eq. 10 yields our proposed minimax learning framework.

Multi-Agent Adversarial Learning

The critical challenge in our proposed minimax learning framework is how to handle the embedded minimization in Eq. 9 and Eq. 10. Due to the continuous action space as well as the non-linearity of Q function, directly optimizing the minimization problem is computationally intractable. A naive approximate solution can be performing an inner-loop gradient descent whenever performing an update step of Eq. 9 or Eq. 10, but this is too computationally expensive for practical use. Here we introduce an efficient and end-to-end solution, multi-agent adversarial learning (MAAL). The main ideas of MAAL can be summarized in two steps: (1) approximate the non-linear Q function by a locally linear function; (2) replace the inner-loop minimization with a 1-step gradient descent. Note the core idea of MAAL, locally linearizing the Q function, is adapted from the recent *adversarial training* technique originally developed for supervised learning. We will discuss the connection between adversarial training and MAAL in the end of this section.

For conciseness, we first consider Eq. 10 and rewrite it into the following form with auxiliary variables ϵ:

$$y = r_i + \gamma\, Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}', a'^\star{}_1, \ldots, a'_i, \ldots, a'^\star{}_N) \quad (11)$$
$$a'_k = \boldsymbol{\mu}'_k(o_k), \;\; \forall 1 \leq k \leq N$$
$$a'^\star{}_j = a'_j + \epsilon_j, \;\; \forall j \neq i$$
$$\epsilon_{j\neq i} = \arg\min_{\epsilon_{j\neq i}} Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}', a'_1 + \epsilon_1, \ldots, a'_i, \ldots, a'_N + \epsilon_N).$$

Eq. 11 can be interpreted as we are now seeking for a set of *perturbations* ϵ such that the perturbed actions $a'^\star$ decrease Q value the most. By linearizing the Q function at $Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a'_1, \ldots, a'_N)$, the desired perturbation ϵ_j can be locally approximated by the gradient direction at $Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a'_1, \ldots, a'_N)$ w.r.t. a'_j. Then we use this local to derive an approximation $\hat{\epsilon}_j$ to the worst case perturbation by taking a small gradient step:

$$\forall j \neq i, \;\; \hat{\epsilon}_j = -\alpha\nabla_{a_j} Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}', a'_1, \ldots, a_j, \ldots, a'_N), \quad (12)$$

where α is a tunable coefficient representing the perturbation rate. It can be also interpreted as the step size of the gradient descent step: when α is too small, the local approximation error will be small but due to the small perturbation, the learned policy can be far from the optimal solution of the minimax objective we proposed; when α is too large, the approximation error may incur too much trouble for the overall learning process and the agents may fail to learn good policies.

We can apply this technique to Eq. 9 as well and eventually derive the following formulation:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}\left[\begin{array}{l}\nabla_{\theta_i}\boldsymbol{\mu}_i(o_i)\nabla_{a_i} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a^\star_1, \ldots, a_i, \ldots a^\star_N)\big| \\ \quad a_i = \boldsymbol{\mu}_i(o_i) \\ \quad a^\star_j = a_j + \hat{\epsilon}_j, \;\; \forall j \neq i \\ \quad \hat{\epsilon}_j = -\alpha_j\nabla_{a_j} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a_1, \ldots, a_N)\end{array}\right], \quad (13)$$

Algorithm 1: Minimax Multi-Agent Deep Deterministic Policy Gradient (M3DDPG) for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration, and receive initial state information $\mathbf{x}$
  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 information $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$, and set $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^k, a^k, r^k, \mathbf{x}'^k)$ from $\mathcal{D}$
      Set $y^k = r_i^k + \gamma\, Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}'^k, a'_1, \ldots, a'_N)\big|_{a'_i = \boldsymbol{\mu}'_i(o_i^k), a'_{j\neq i} = \boldsymbol{\mu}'_j(o_j^k) + \hat{\epsilon}'_j}$ with $\hat{\epsilon}'_j$ defined in Eq. 14
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_k \left(y^k - Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}^k, a_1^k, \ldots, a_N^k)\right)^2$
      Update actor using the sampled policy gradient with $\hat{\epsilon}_j$ defined in Eq. 13:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_k \nabla_{\theta_i}\boldsymbol{\mu}_i(o_i^k)\nabla_{a_i} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}^k, a_1^\star, \ldots, a_i, \ldots, a_N^\star)\big|_{a_i = \boldsymbol{\mu}_i(o_i^k), a_{j\neq i}^\star = a_j^k + \hat{\epsilon}_j}$$

    end for
    Update target network parameters for each agent i: $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
  end for
end for

and

$$\begin{aligned}
\mathcal{L}(\theta_i) &= \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}[(Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a_1, \ldots, a_N) - y)^2], \qquad (14)\\
y &= r_i + \gamma\, Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}', a'^\star{}_1, \ldots, a'_i, \ldots, a'^\star{}_N)\\
a'_k &= \boldsymbol{\mu}'_k(o_k), \quad \forall 1 \le k \le N\\
a'^\star{}_j &= a'_j + \hat{\epsilon}'_j, \quad \forall j \neq i\\
\hat{\epsilon}'_j &= -\alpha_j \nabla_{a'_j} Q^{\boldsymbol{\mu}'}_{\mathrm{M},i}(\mathbf{x}, a'_1, \ldots, a'_N),
\end{aligned}$$

where $\alpha_1, \ldots, \alpha_N$ are additional parameters. MAAL only requires one additional gradient computation, and can be executed in a fully end-to-end fashion. Finally, combining Eq. 13 and Eq. 14 completes MAAL. The overall algorithm, M3DDPG, is summarized as Algo. 1.

Discussion

Connection to Adversarial Training *Adversarial training* is a robust training approach for deep neural networks on supervised learning (Goodfellow, Shlens, and Szegedy 2014; Miyato, Dai, and Goodfellow 2017; Wu, Bamman, and Russell 2017). The core idea is to force the classifier to predict correctly even when given *adversarial examples*, which are obtained by adding a small adversarial perturbation to the original input data such that the classification loss can be decreased the most. Formally, suppose the classification loss function is $\mathcal{L}(\theta) = \mathbb{E}_{x,y}\left[f_\theta(x; y)\right]$ with input data x and label y. Adversarial training aims to optimize the following adversarial loss instead

$$\begin{aligned}
\mathcal{L}_{\text{adv}}(\theta) &= \mathbb{E}_{x,y}\left[f_\theta(x + \epsilon^\star; y)\right] \qquad (15)\\
\epsilon^\star &= \arg\max_{\|\epsilon\|\le\alpha} f_\theta(x + \epsilon; y).
\end{aligned}$$

The core technique to efficiently optimize $\mathcal{L}_{\text{adv}}(\theta)$ is to locally linearize the loss function at $f_\theta(x; y)$ and approximate $\epsilon^\star$ by the scaled gradient.

Thanks to the centralized Q function, which takes the actions from all the agents as part of the input, we are able to easily inject the minimax optimization (Eq. 11) and represent it in a similar way to adversarial training (Eq. 15) so that we can adopt the similar technique to effectively solve our minimax optimization in a fully end-to-end fashion.

Connection to Single Agent Robust RL M3DDPG with MAAL can be also viewed as the special case of robust reinforcement learning (RRL) (Morimoto and Doya 2005) in the single agent setting, which aims to bridge the gap between training in simulation and testing in the real world by adding adversarial perturbations to the transition dynamics during training. Here, we consider the multi-agent setting and add worst case perturbations to *actions* of opponent agents during training. Note that in the perspective of a single agent, perturbations on opponents' actions can be also considered as a special adversarial noise on the dynamics.

Choice of α In the extreme case of $\alpha = 0$, M3DDPG degenerates to the original MADDPG algorithm while as α increases, the policy learning tends to be more robust but the optimization becomes harder. In practice, using a fixed α throughout training can lead to very unstable learning behavior due to the changing scale of the gradients. The original adversarial training paper (Goodfellow, Shlens, and Szegedy 2014) suggests to compute ϵ with a fixed norm, namely $g = \nabla_x f_\theta(x; y)$, $\hat{\epsilon} = \alpha\frac{g}{\|g\|}$, where x denotes the input data to the classifier and y denotes the label. Accordingly, in our M3DDPG algorithm, we can adaptively compute the perturbation $\hat{\epsilon}_j$ by

$$g = \nabla_{a_j} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a_1, \ldots, a_N), \quad \hat{\epsilon}_j = -\alpha_j \frac{g}{\|g\|}. \qquad (16)$$

Eq. 16 generally works fine in practice but in some hard multi-agent learning environments, unstable training behav-

ior can be still observed. We suspect that it is because the changing norm of actions in these situations. Different from the supervised learning setting where the norm of the input data x is typically stable, in reinforcement learning the norm of actions can drastically change even in a single episode. Therefore, it is possible to see cases that even a perturbation with a small fixed norm overwhelms the action a_j, which may potentially lead to computational stability issue. Therefore, we also introduce the following alternative for adaptive perturbation computation:

$$g = \nabla_{a_j} Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x}, a_1, \ldots, a_N), \quad \hat{\epsilon}_j = -\alpha_j \|a_j\| \frac{g}{\|g\|}. \quad (17)$$

Lastly, note that in a mixed cooperative and competitive environment, ideally we only need to add adversarial perturbations to competitors. But empirically we observe that also adding (smaller) perturbations to collaborators can further improve the quality of learned policies.

5. Experiments

We adopt the same *particle-world* environments as the MADDPG paper (Lowe et al. 2017) as well as the training configurations. α is selected from a grid search over 0.1, 0.01 and 0.001. For testing, we generate a fixed set of 2500 environment configurations (i.e., landmarks and birthplaces) and evaluate on this fixed set for a fair comparison.

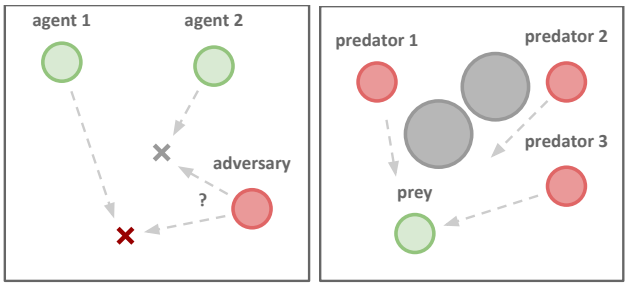

Figure 1: Illustrations of some environments we consider, including *Physical Deception* (left) and *Predator-Prey* (right).

Environments

The particle world environment consists of N cooperative agents, M adversarial agents and L landmarks in a two-dimensional world with continuous space. We focus on the four mixed cooperative and competitive scenarios to best examine the effectiveness of our minimax formulation.

Covert communication This is an adversarial communication environment, where a speaker agent ('Alice') must communicate a message to a listener agent ('Bob') ($N = 2$), who must reconstruct the message at the other end. However, an adversarial agent ('Eve') ($M = 1$) is also observing the channel, and wants to reconstruct the message — Alice and Bob are penalized based on Eve's reconstruction, and thus Alice must encode her message using a randomly generated *key*, known only to Alice and Bob.

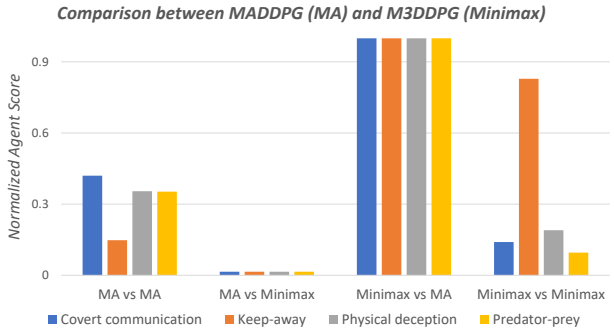

Figure 2: Comparison between M3DDPG (Minimax) and classical MADDPG (MA) on the four mixed competitive environments. Each bar cluster shows the 0-1 normalized score for a set of competing policies in different roles (agent vs adversary), where a higher score is better for the agent. In all cases, M3DDPG outperforms MADDPG when directly pitted against it.

Keep-away This scenario consists of $L = 1$ target landmark, $N = 2$ cooperative agents and $M = 1$ adversarial agent. Cooperating agents need to reach the landmark and keep the adversarial agent away from the landmark by pushing it while the adversarial agent must stay at the landmark to occupy it.

Physical deception Here, $N = 2$ agents cooperate to reach a single target landmark from a total of $L = 2$ landmarks. They are rewarded based on the minimum distance of any agent to the target (so only one agent needs to reach the target landmark). However, a lone adversary ($M = 1$) also desires to reach the target landmark; the catch is that the adversary does not know which of the landmarks is the correct one. Thus the cooperating agents, who are penalized based on the adversary distance to the target, learn to spread out and cover all landmarks so as to deceive the adversary.

Predator-prey In this variant of the classic predator-prey game, $N = 3$ slower cooperating agents must chase the faster adversary ($M = 1$) around a randomly generated environment with $L = 2$ large landmarks impeding the way. Each time the cooperative agents collide with an adversary, the agents are rewarded while the adversary is penalized.

Comparison to MADDPG

To evaluate the quality of learned policies trained by different algorithms in competitive scenarios, we measure the performance of agents trained by our M3DDPG and agents by classical MADDPG in the roles of both normal agent and adversary in each environment.

The results are demonstrated in Figure 2, where we measure the rewards of the normal agents in different scenarios and normalize them to 0-1. We notice that in all the environments, the highest score is achieved when the M3DDPG agents play as the normal agents against the MADDPG adversary (Minimax vs MA); while the lowest score is when the MADDPG agents act as normal agents against the M3DDPG adversary (MA vs Minimax). This indicates that policies

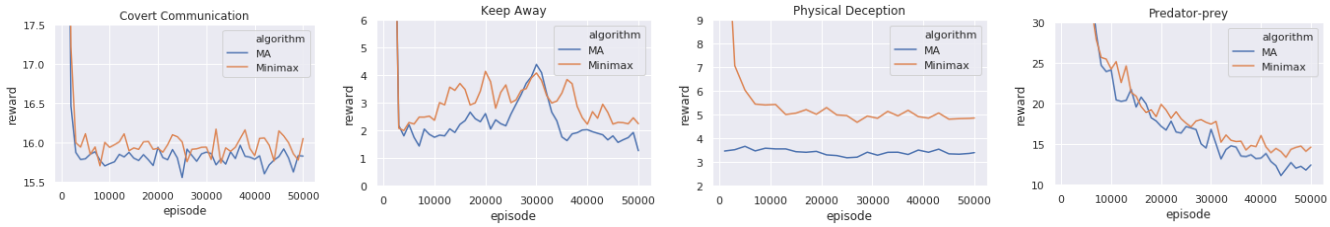

Figure 3: Performances of M3DDPG (Minimax, red) and MADDPG (MA, blue) under the *worst situation*, i.e., against the *disruptive adversaries*, on convert communication, keep-away, physical deception and predator-pray from left to right. The y-axis denotes the reward of normal agents (fixed) and x-axis denotes the training episodes performed of the disruptive adversaries. Higher reward implies a more robust policy. Agents trained by M3DDPG (Minimax) perform better on all the scenarios.

trained by M3DDPG have much higher quality than original MADDPG.

Evaluation with Disruptive Adversaries

Instead of M3DDPG and MADDPG directly competing against each other, now we consider their performances in the *worst* possible situations through their individual competitions against *disruptive adversaries*. We construct disruptive adversaries by (1) fixing the trained normal agents (M3DDPG or MADDPG); (2) setting the reward of the disruptive adversary as the negative value of normal agents' reward (so that the problem becomes zero-sum); and then (3) using DDPG to train the disruptive adversary from scratch under the zero-sum reward setting.

In the particle world environment, the competitive scenarios are generally not zero-sum, which implies that according to the default reward function, the adversaries may have different purposes rather than directly disrupting the normal agents. So, in order to evaluate the effectiveness our minimax optimization in the worst situation, we convert every problems into a zero-sum form and compare the performances of our M3DDPG agents as well as the MADDPG agents against this artificially constructed adversaries. Moreover, since each of our four environments has only 1 adversary, after fixing the normal agents, the learning problem degenerates to the single agent setting and classical DDPG is sufficient to stably train a disruptive adversary.

The results are shown in Figure 3, where we plot the reward of the fixed normal agents of different algorithms as the training of the disruptive adversaries progresses until convergence. Note that due to the different environment designs, the difficulty for the disruptive agents to break the strategy of normal agents varies: for example, in *convert communication*, since the private key is not accessible to the adversary agent, breaking the encrypted communication will be very hard; while in *physical deception*, since we do not allow communication and fix the normal agents, a smart enough adversary may easily infer the target landmark by observing the initial behavior of the two cooperative agents. Nevertheless, despite these intrinsic properties, the M3DDPG agents (Minimax) achieves higher reward in all the scenarios, which implies better robustness even in the worst situation.

6. Conclusion

In this paper, we propose a novel algorithm, minimax mulit-agent deep deterministic policy gradient (M3DDPG), for robust multi-agent reinforcement learning, which leverages the minimax concept and introduces a minimax learning objective. To efficiently optimize the minimax objective, we propose MAAL, which approximates the inner-loop minimization by a single gradient descent step. Empirically, M3DDPG outperforms the benchmark methods on four mixed cooperative and competitive scenarios.

Nevertheless, due to the single step gradient approximation in MAAL, which is efficient in computation, an M3DDPG agent can only explore locally worst situation during the evolving process at training, which can still lead to unsatisfying behavior when testing opponents have drastically different strategies. It will be an interesting direction to re-examine the robustness-efficiency trade-off in MAAL and further improve policy learning by placing more computations on the minimax optimization. We leave this as our future work.

References

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews* 38(2):156.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016a. Learning to communicate with deep multi-agent reinforcement learning. *CoRR* abs/1605.06676.

Foerster, J. N.; Assael1, Y. M.; de Freitas, N.; and Whiteson, S. 2016b. Learning to Communicate to Solve Riddles with Deep Distributed Recurrent Q-Networks.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2017. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*.

Foerster, J.; Chen, R. Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2018. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 122–130.

Gao, C.; Mueller, M.; and Hayward, R. 2018. Adversarial policy gradient for alternating markov games.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Grau-Moya, J.; Leibfried, F.; and Bou-Ammar, H. 2018. Balancing two-player stochastic games with soft q-learning. *arXiv preprint arXiv:1802.03216*.

Haarnoja, T.; Tang, H.; Abbeel, P.; and Levine, S. 2017. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*.

He, H.; Boyd-Graber, J.; Kwok, K.; and Daumé III, H. 2016. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, 1804–1813.

Hu, J.; Wellman, M. P.; et al. 1998. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, 242–250. Citeseer.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Perolat, J.; Silver, D.; Graepel, T.; et al. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, 4190–4203.

Lazaridou, A.; Peysakhovich, A.; and Baroni, M. 2016. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*.

Leibo, J. Z.; Zambaldi, V. F.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *CoRR* abs/1702.03037.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1):1334–1373.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, 157–163.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O. P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 6379–6390.

Mandlekar, A.; Zhu, Y.; Garg, A.; Fei-Fei, L.; and Savarese, S. 2017. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, 3932–3939. IEEE.

Miyato, T.; Dai, A. M.; and Goodfellow, I. 2017. Adversarial training methods for semi-supervised text classification. *5th International Conference on Learning Representations*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

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

Morimoto, J., and Doya, K. 2005. Robust reinforcement learning. *Neural computation* 17(2):335–359.

OpenAI. 2018. OpenAI Five. https://blog.openai.com/openai-five/. Accessed: 2018-09-03.

Osborne, M. J., et al. 2004. *An introduction to game theory*, volume 3. Oxford university press New York.

Peng, P.; Yuan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; and Wang, J. 2017a. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *CoRR* abs/1703.10069.

Peng, X. B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2017b. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*.

Pérolat, J.; Strub, F.; Piot, B.; and Pietquin, O. 2016. Learning nash equilibrium for general-sum markov games from batch data. *arXiv preprint arXiv:1606.08718*.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, 387–395.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sukhbaatar, S.; Fergus, R.; et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2244–2252.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Tesauro, G. 2004. Extending q-learning to general adaptive multi-agent systems. In *Advances in neural information processing systems*, 871–878.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, 23–30. IEEE.

Uther, W. T., and Veloso, M. M. 1997. Generalizing adversarial reinforcement learning. In *Proceedings of the AAAI Fall Symposium on Model Directed Autonomous Systems*, 206. Citeseer.

Wei, E.; Wicke, D.; Freelan, D.; and Luke, S. 2018. Multiagent soft q-learning. *arXiv preprint arXiv:1804.09817*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wu, Y.; Bamman, D.; and Russell, S. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1778–1783.

Wu, C.; Kreidieh, A.; Parvate, K.; Vinitsky, E.; and Bayen, A. M. 2017. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*.

Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; and Wang, J. 2018. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*.