Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

Robust Opponent Modeling via Adversarial Ensemble Reinforcement Learning

Macheng Shen and Jonathan P. How
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge
Massachusetts, 02139
{macshen, jhow}@mit.edu

Abstract

This paper studies decision-making in two-player scenarios where the type (e.g. adversary, neutral, or teammate) of the other agent (opponent) is uncertain to the decision-making agent (protagonist), which is an abstraction of security-domain applications. In these settings, the reward for the protagonist agent depends on the type of the opponent, but this is private information known only to the opponent itself, and thus hidden from the protagonist. In contrast, as is often the case, the type of the protagonist agent is assumed to be known to the opponent, and this information-asymmetry significantly complicates the protagonist's decision-making. In particular, to determine the best actions to take, the protagonist agent must infer the opponent type from the observations and agent modeling. To address this problem, this paper presents an opponent-type deduction module based on Bayes' rule. This inference module takes as input the imagined opponent's decision-making rule (opponent model) as well as the observed opponent's history of actions and states, and outputs a belief over the opponent's hidden type. A multiagent reinforcement learning approach is used to develop this game-theoretic opponent model through self-play, which avoids the expensive data collection step that requires interaction with a real opponent. Besides, this multiagent approach also captures the strategy interaction and reasoning between agents. In addition, we apply ensemble training to avoid over-fitting to a single opponent model during the training. As a result, the learned protagonist policy is also effective against unseen opponents. Experimental results show that the proposed game-theoretic modeling, explicit opponent type inference and the ensemble training significantly improves the decision-making performance over baseline approaches, and generalizes well against adversaries that have not been seen during the training.

Introduction

Recent advances in deep reinforcement learning (DRL) have achieved breakthroughs in solving challenging decision-making problems in both single-agent environments (Mnih et al. 2013; Hausknecht and Stone 2015; Andrychowicz et al. 2018) and multiagent games (Silver et al. 2017; Moravčík et al. 2017; Jaderberg et al. 2018; OpenAI 2018; Schrittwieser et al. 2019). Among these multiagent games, some have fully observable states, and others includes hidden states that are partially-observable (or unobservable) to some agents. Nevertheless, the agent types in these games are known to all the agents. For example, Go is a zero-sum game where the two players compete with each other, where player 1 knows that its opponent player 2 is playing an adversarial role that tries to minimize player 1's winning probability. However, there also exist many important multiagent scenarios in which some of the agent types are uncertain or not well known to all. For example, in a cyber-security scenario, the network administrator (protagonist) observes signals that could have been sent by normal users (neutral) or by malicious attacker agents (adversary), without knowing the type of each individual (Hereafter, we refer to the second agent of uncertain types as opponent). Given this type uncertainty, the network administrator needs to infer the identity of the signal sender before making the decision of blocking the signal or not, which significantly increases the complexity of the network administrator's decision-making process.

An opponent model is typically required, either explicitly (Lockett, Chen, and Miikkulainen 2007; He et al. 2016; Carmel and Markovitch 1998; Chakraborty and Stone 2014), or implicitly (Bard et al. 2013; Bjarnason and Peterson 2002; Fagan and Cunningham 2003; Sohrabi, Riabov, and Udrea 2016), to make such inference feasible. An explicit modeling approach tries to model the opponent's policy directly, while an implicit model instead estimates intermediate statistics such as the anticipated value of the protagonist agent's policy against the opponent (Rubin and Watson 2011). In the scenarios with uncertain opponent types, the implicit opponent modeling approach could be predicting opponent's hidden type from the observation of opponent's states and actions, while an explicit modeling approach also permit this hidden type inference by using the Bayes' rule. We will show evidence that explicit opponent modeling leads to superior performance compared with implicit opponent modeling.

One simple explicit opponent modelling approach is to treat the opponent as a goal-directed Markov Decision Process MDP) agent by specifying a reward function and then learn/use the optimal goal-achieving policy as the opponent model. This approach has two limitations. First, an adversarial opponent has the incentive of concealing its identity through disguise behavior. For example, an attacker might mimic a normal user's behavior to avoid being detected immediately, while a simple goal-directed reward cannot capture

this strategic behavior. We argue that a game-theoretic opponent model which captures the full interaction and strategic reasoning between the protagonist agent and the opponent agent is superior than a goal-directed opponent model of single-agent perspective. The second limitation of using the opponent's optimal policy as the opponent model is the high sensitivity with respect to modeling error, which could result in significant performance degradation against a previously unseen opponent. To mitigate sensitivity and improve the robustness of the opponent model, we propose to learn an ensemble of diverse opponent policies through multiagent reinforcement learning (MARL) and distill these policies to form an 'average' opponent model for inferring the hidden type of an opponent from its observed state-action history.

This paper presents an algorithmic framework for learning robust policies in multiagent scenarios with uncertain opponent types [1]. We focus on the scenarios where the opponent type is unknown to the protagonist agent, but the protagonist agent's identity is certain to the opponent. We are interested in learning an opponent model for the protagonist to update its belief on the opponent type and defend robustly against previously unseen opponent. This setting is an abstraction of security-domain scenarios, but has seldom been well-studied in the context of MARL.

In summary, the paper has two main contributions: We derived a recursive rule for explicitly updating the belief on opponent's hidden types, which significantly improves the decision-making quality in the security-domain scenario compared against the standard Recurrent Neural Network (RNN) based approach (implicitly infer the hidden types) in partially-observable domain. We developed a game-theoretic opponent modeling approach which captures more sophisticated adversarial behaviors than what a goal-directed MDP opponent model can capture. As a result, the protagonist agent can learn a stronger policy against sophisticated adversaries. We also showed that learning an ensemble of diverse adversarial policies is an effective way to improve the robustness of the protagonist policy against previously unseen adversaries.

Preliminary

This section reviews the preliminary of the decision-making framework and solution techniques.

Decision-making Framework

The multiagent scenarios with uncertain opponent types as described in the Introduction Section is a special case of Bayesian Games. A Bayesian game (BG) (Cheon and Iqbal 2008) is given by $G = \langle \mathcal{I}, \langle \mathcal{S}, \mathcal{H} \rangle, \{b^0\}, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, \mathcal{P}, \{R_i\} \rangle$, where,

- $\mathcal{I}$ is a finite set of agents indexed by $1, \ldots, n$,
- $\Omega = \langle \mathcal{S}, \mathcal{H} \rangle$ is the set of state of nature, which includes the physical states and the agent hidden states corresponding to agent types in our problem,
- $b^0 \in \Delta(\mathcal{S} \times \mathcal{H})$ is the common prior probability distribution over Ω, where Δ is the probabilistic simplex,
- $\mathcal{A}_i$ is the action space of each agent, and we use $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle$ to denote the joint action, with $a_i \in \mathcal{A}_i$,
- $\mathcal{O}_i$ is the observation space for each agent, and we use $\boldsymbol{o} = \langle o_1, \ldots, o_n \rangle$ to denote the joint observation, with $o_i \in \mathcal{O}_i$,
- $\mathcal{P}$ is the Markovian state transition and observation probability, denoted as $\mathcal{P}^T(s'|s, \boldsymbol{a})$ and $\mathcal{P}^O(\boldsymbol{o}|s, \boldsymbol{a})$,
- $\mathcal{R}_i : \Omega \times \mathcal{A} \to \mathbb{R}$ is the reward function of each agent.

Note that the reward function depends on both the state, action and the hidden agent types. The same action against different types of opponents could result in rewards of opposite signs (positive v.s. negative). As a result, correctly inferring the opponent's type is crucial for the protagonist agent to maximize its reward.

Belief Space Reward

In partially observable domains, the belief-space value function is used instead of the state value function for decision-making, which is defined as the expected cumulative reward with respect to the state-action distribution under the belief space policy π,

$$V^{\pi}(b_0) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s^t \sim p(s^t), a^t \sim \pi(b^t)}[R(s^t, a^t)], \quad (1)$$

where $p(s^t)$ is the state distribution and b^t is the belief over the state. If the belief is unbiased, then $p(s^t) = b(s^t)$ and Eq. 1 reduces to

$$V^{\pi}(b^0) = \sum_{t=0}^{\infty} \gamma^t r(b^t, a^t), \quad (2)$$

where $r(b, a) = \mathbb{E}_{s \sim b(s)}[r(s, a)]$ is the belief-space reward. In model-free reinforcement learning (RL), reward is sampled from the environment at each step. The belief-space reward sample $r(b^t, a^t)$ has lower variance than the actual reward sample $r(s^t, a^t)$ because the uncertainty associated with the state distribution has been analytically marginalized out. This low variance is beneficial for RL algorithms. In general, however, the state distribution p^t and the belief b^t could be different, for example, when the environment model $\mathcal{P}$ used for belief update is biased. In this case, the policy maximizing the belief space cumulative reward Eq. 2 does not necessarily maximize the actual cumulative reward Eq. 1. This mismatch is inevitable in multiagent scenarios with uncertain opponent types, because it is impossible to perfectly model an opponent, which makes these type of problems challenging. Therefore, developing an accurate opponent type inference scheme is crucial for learning a good belief-space policy.

Approach

We first give an overview of our approach. We use MARL to derive a game-theoretic model of the opponent, where the protagonist agent and the opponent agent are trained against each other. Note that this learned opponent model is an 'imagined opponent', which is different than the unseen opponent during the testing. This opponent model is only used to update the protagonist agent's belief about the type of the opponent during the testing. We use neural network to

[1] code: github.com/MachengShen/robust_opponent_modeling

represent a belief space policy for the protagonist agent. The belief state is updated via Bayes' rule using the learned opponent model. The opponent model learning process consists of an ensemble policy training step and a policy distillation step. We apply a neuro-evolutionary method to improve the diversity of the ensemble population to avoid over-fitting to a single opponent model, which improves the robustness of the opponent modeling. The above steps are illustrated in Fig. 1. We present the detail of each step in the following sections.

MARL with Ensemble Training

To improve the policy robustness of the protagonist agent, we formulate its RL objective as the average cumulative reward against an ensemble of opponent policies of size K, as in (Lowe et al. 2017),

$$J(\pi_i) = \mathbb{E}_{\substack{k\sim \mathrm{unif}(1,K),\\ a_i\sim\pi_i,\\ a_{-i}\sim\pi_{-i}^{(k)}}} \left[\sum_{t=0}^{\infty} \gamma^t r_i(s, \boldsymbol{a})\right], \tag{3}$$

where $\mathrm{unif}(1, K)$ denotes the uniform distribution. The policy ensemble $\{\pi_{-i}^{(k)}, k = 1, 2, ..., K\}$ is also learned from training the opponent agent against the protagonist policy. Via this concurrent learning, both the protagonist agent and its opponent improve their policies. Nonetheless, there is no explicit mechanism to enforce distinction among the policies within the ensemble. As a result, there could be policies that are very similar to the others. To address this redundancy issue, we apply the cooperative evolutionary reinforcement learning (CERL) approach (Khadka et al. 2019). The key idea is to use different hyper-parameter settings for each opponent policy, use an off-policy learning algorithm and a shared experience replay buffer to improve the sample efficiency for efficient training.

Belief Space Policy and Belief Update

We use the belief space approach for agent policy learning. Agents explicitly maintain a belief over the hidden states (including the uncertain opponent types), and learns a belief-space policy that maps belief to action. We parameterize this mapping using a multi-layer perceptron (MLP). Instead of Eq. 3, the learning objective becomes,

$$J(\pi_i) = \mathbb{E}_{\substack{k\sim \mathrm{unif}(1,K),\\ a_i\sim\pi_i(b_i),\\ a_{-i}\sim\pi_{-i}^{(k)}}} \left[\sum_{t=0}^{\infty} \gamma^t r_i(b_i, \boldsymbol{a})\right]. \tag{4}$$

A belief update mechanism is required to fully specify the agent policy. The belief is the posterior distribution over the hidden states given action and observation history, $b_i^t = p(s^t, h^t|o_i^{0:t})$. The intuition behind this hidden state inference is: the observation is affected by the joint action, which depends on the joint policy as well as the hidden type; the joint policy also depends on some hidden state such as agent type. Therefore, reasoning about the hidden agent type via modeling the agent policy is possible.

We present the belief update rule for hidden type inference, starting from introducing our key assumptions.

Assumption 1 (Objective observation)**.** *Agents' observations are conditionally independent of their internal type states, given the physical state and joint action.*

Assumption 2 (Independent decision-maker)**.** *Each agent i makes its own decision conditioned on its own type variable h_i^t, and its immediate observation o_i^t.*

Assumption 3 (Time-invariant agent type)**.** *Within each episode, the agent types are sampled at the beginning of this episode and do not change over time.*

Based on these assumptions, we derive the belief update scheme beginning from the Bayes' rule,

$$b_i^t \propto p(o_i^t|s^t, h^t, o_i^{0:t-1}) p(s^t, h^t|o_i^{0:t-1}) \tag{5}$$

the first term of which can be written as

$$\begin{aligned} p(o_i^t|s^t, h^t, o_i^{0:t-1}) &= p(o_i^t|s^t, h^t) \\ &= \int p(o_i^t|\boldsymbol{a}^t, s^t, h^t) p(\boldsymbol{a}^t|s^t, h^t) d\boldsymbol{a}^t \end{aligned} \tag{6}$$

where the first term, $p(o_i^t|\boldsymbol{a}^t, s^t, h^t)$, is the observation probability. It is reasonable to assume $p(o_i^t|\boldsymbol{a}^t, s^t, h^t) = p(o_i^t|\boldsymbol{a}^t, s^t) = \mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$, i.e., agents' observations are independent from their internal type states (see Assumption 1). The second term in Eq. (6) $p(\boldsymbol{a}^t|s^t, h^t)$ is the key connection between opponent type inference and opponent policy modeling. Intuitively, this term is closely related to agent policy, we introduce the joint observation immediately before all the agents taking actions, denoted as $\boldsymbol{o}^{t^-}$, and rewrite $p(\boldsymbol{a}^t|s^t, h^t)$ as follows,

$$p(\boldsymbol{a}^t|s^t, h^t) = \int p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, s^t, h^t) p(\boldsymbol{o}^{t^-}|s^t, h^t) d\boldsymbol{o}^{t^-}. \tag{7}$$

The second term $p(\boldsymbol{o}^{t^-}|s^t, h^t)$ is the observation probability $\mathcal{P}^O(\boldsymbol{o}^{t^-}|s^t)$. This probability is not conditioned on the immediate joint actions, because the joint actions have not been taken yet. The first term $p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, s^t, h^t)$ is related to the joint policies. In order to reveal this connection, we invoke Assumption 2. Based on this assumption, we have the following factorization,

$$p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, s^t, h^t) = p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, h^t) \approx \prod_j^N \pi_j(o_j^t|h_j). \tag{8}$$

To summarize, Eq. (6) can be represented as:

$$p(o_i^t|s^t, h^t, o_i^{0:t-1}) = \mathbb{E}_{\boldsymbol{a}^t\sim\boldsymbol{\pi}(\bar{\boldsymbol{o}}|\boldsymbol{h})}[\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)], \tag{9}$$

where $\bar{\boldsymbol{o}} = \int \mathcal{P}^O(\boldsymbol{o}^{t^-}|s^t) d\boldsymbol{o}^{t^-}$. The interpretation of Eq. (9) is that the probability of receiving an observation o_i^t is the expected observation by marginalizing out all the possible joint actions over the observation probability $\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$, where the probability of the joint actions $\boldsymbol{\pi}(\bar{\boldsymbol{o}}|\boldsymbol{h})$ is obtained from the joint policies using the expected joint observation of all the agents.

The second term in Eq. (5), $p(s^t, h^t|o_i^{0:t-1})$ can be expressed as

$$\begin{aligned} &\int p(s^t, h^t|s^{t-1}, h^{t-1}) p(s^{t-1}, h^{t-1}|o_i^{0:t-1}) ds^{t-1} dh^{t-1} \\ &= \int p(s^t, h^t|s^{t-1}, h^{t-1}) b_i^{t-1} ds^{t-1} dh^{t-1}. \end{aligned} \tag{10}$$

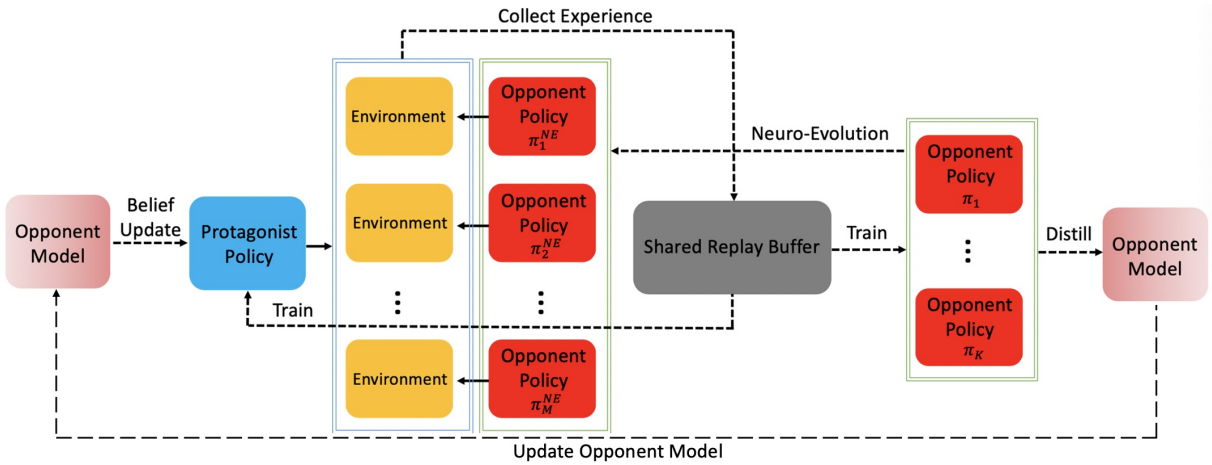


Figure 1: Illustration of the workflow: we train one protagonist policy that uses an internal opponent model for belief update. The opponent model is learned by distilling an ensemble of opponent policies trained against the protagonist policy. Both the protagonist and the opponent improve their policies concurrently in the training environment

To further simplify this expression, we invoke Assumption 3 so that $p(s^t, h^t|s^{t-1}, h^{t-1}) = p(s^t, h^t|s^{t-1}, h^{t-1})\delta(h^t|h^{t-1})$, where $\delta(h^t|h^{t-1})$ denotes the Dirac-delta measure. With this assumption, Eq. (10) simplifies to

$$p(s^t, h^t|o_i^{0:t-1}) = \int p(s^t|s^{t-1}, h^t) b_i^{t-1} ds^{t-1}. \quad (11)$$

Combining Eqs. (5), (9), and (11) yields the recursive belief update rule,

$$b_i^t \propto \mathbb{E}_{\boldsymbol{a}^t \sim \boldsymbol{\pi}(\bar{o}|\boldsymbol{h})}[\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)] \int p(s^t|s^{t-1}, h^t) b_i^{t-1} ds^{t-1}, \quad (12)$$

which has the following interpretation: To infer the state of current step, we can predict it based on the posterior belief of the last step, by propagating the physical state distribution and correcting the belief over the hidden type variable via comparing the actual observation with the anticipated observation according to agent policy modeling.

Remark 1. *In the belief update rule, the inference over the hidden type variable is implicit inside the expectation term. The observation probability is crucial to the discriminative power of this inference. To illustrate this point, consider an extreme case where the observation contains no information about agents' actions, i.e., $\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$ is not a function of $\boldsymbol{a}^t$. In this case, this expectation term will be independent of the joint policy (will be a constant due to normalization condition of expectation). As a result, no information about the hidden type variable can be extracted from this term. This makes sense, because if the observation tells us nothing about the actions taken by the other agents (dictated by their policies and hidden types), then it is impossible to update our belief over their hidden types. Conversely, if the observation contains full information about the joint action (e.g., the protagonist directly observes the joint action), this expectation term would be highly dependent on the joint policies (therefore, on the hidden type variable), and the discriminative power of this inference scheme is maximized.*

The observation probability $\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$, the state transition probability $p(s^t|s^{t-1}, h^{t-1})$, and agent policies $\boldsymbol{\pi}$, are required to implement the belief update, which is anticipated. This work focuses on a special case in which the physical states are fully observable to all the agents, so agents do not need to maintain a belief over s^t. This assumption simplifies the computational aspect of the problem, but it does not diminish the central difficulty of the problem, i.e., inferring the hidden type of opponent.

To approximate the policies of agent j of each possible type $\{h_j^{(m)}\}_{m=1}^M$, recall that, in the ensemble training step, we create K different policies $\{\pi_{j,m}^{(k)}\}_{k=1}^K$ for each agent of each type. Here we use shorthand $\pi_{j,m}$ to denote agent j with type $h_j^{(m)}$. Each policy within one ensemble can be interpreted as one of the likely strategies that could be adopted by agent j with type $h_j^{(m)}$. However, in the belief update equation, we need only one single policy for agent j with type $h_j^{(m)}$. As a result, we must synthesize the policy ensemble into one representative policy that best represents the average behavior of the policy ensemble. We learn this representative policy by minimizing the information theoretic distance (Kullback−Leibler (KL) divergence) between this policy and the policy ensemble. The resulting optimization problem to learn the representative policy $\pi_{j,m}^0$ is then

$$J(\pi_{j,m}^0) = \sum_{k=1}^K \mathbb{KL}(\pi_{j,m}^{(k)}, \pi_{j,m}^0), \quad (13)$$

which is policy distillation (Teh et al. 2017). The solution to this minimization is,

$$\pi_{j,m}^0 = \frac{1}{K} \sum_{k=1}^K \pi_{j,m}^{(k)}, \quad (14)$$

Procedure 1 Ensemble evaluation

1: Fix the protagonist policy
2: Train a single opponent policy against the fixed protagonist policy
3: Obtain the average protagonist agent reward r^p and opponent reward r^o after training

Procedure 2 Ensemble Optimization

1: Randomly select an operation ξ from {**pop**, **append**, **exchange**} to apply on the policy ensemble
2: Obtain a new metrics ρ_{new} via Procedure 2
3: Accept the operation ξ with probability p, where $p = \exp(\min\{0, \rho_{\textbf{old}} - \rho_{\textbf{new}}\}/T)$

which happens to be the average over the policies within one ensemble. As a result, we can approximate $\pi^0_{j,m}$ by training a policy distillation network to minimize the residue of Eq. (14) using data sample, which is computationally much more efficient than calculating all the K policies during testing.

Policy Ensemble Optimization

The ensemble training step typically improves the robustness of the protagonist agent's policy. However, two problems need to be addressed to make this approach more effective and efficient. First, we want a metric for measuring policy robustness and we want to explicitly optimize this robustness metrics. Second, we want to reduce the additional computation overhead introduced by ensemble training.

We address these two problems by training a hold-out adversary agent to exploit the protagonist agent's policy, and use the resulted protagonist agent's reward as a robustness score. Instead of using a fixed-size ensemble, we dynamically resize the ensemble through three operations: **pop**, **append**, and **exchange**. **pop** randomly removes one policy from the ensemble and pushes it into a deactivation-set. **append** randomly selects one policy from the deactivation-set and adds it to the ensemble. **exchange** randomly selects one policy from both the ensemble and the deactivation-set and exchanges them.

The objective of modifying the ensemble is to obtain a good trade-off between robustness and computational complexity, which is dominated by the ensemble size. We propose to measure the robustness via Procedure 1, and we define the following metric (with weights λ_i)

$$\rho = -r^p + \lambda_1 r^o + \lambda_2 K, \tag{15}$$

where K is the varying size of the policy ensemble. The combined reward term $-r^p + \lambda_1 r^o$ is a measure of the robustness of the protagonist policy, which is noisy due to the intrinsic stochasticity of RL, while K is a surrogate measure for computation complexity. Minimizing ρ leads to a trade-off between policy robustness and computation complexity. We interpret this minimization problem as a stochastic optimization over the powerset of the initial policy ensemble. We solve this stochastic optimization via simulated annealing (Procedure 2).

Hyper-parameter	Value
Opponent loss weight λ_1	0.1
Ensemble size weight λ_2	1.0
Initial temperature T_0	30.0
Minimum temperature T_{min}	0.2
Temperature decay rate	0.975

Table 1: Hyper-parameter of ensemble optimization

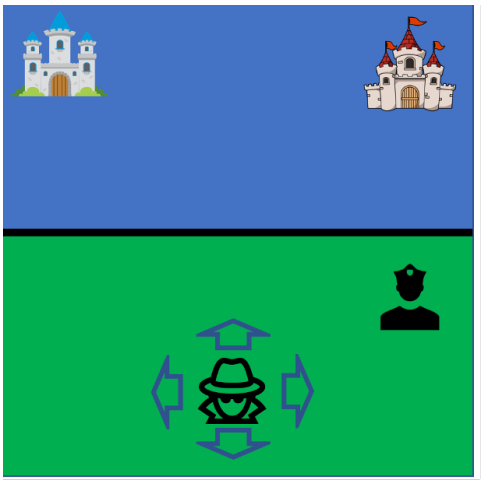


Figure 2: Sketch of the scenario, the opponent (bottom middle) could be an adversary or a neutral agent. The protagonist agent (upper right corner in the green region) must infer the identity of the opponent and tag the adversary and let the neural agent pass.

Evaluation

This section addresses the following questions:

1. Is it necessary to use ensemble training, considering its additional computation overhead?
2. Is it beneficial to explicitly model opponent policy and maintain a belief?
3. How much improvement do we get from ensemble training?

Scenario: Urban-security Game

We design a two-player urban-security game with uncertain opponent types to evaluate our algorithm, as illustrated in Fig. 2. There are two agents: the protagonist agent (the officer) and the opponent agent with two possible types (either a neutral agent or an adversary). The opponent's objective is to reach its home base (a neutral opponent goes to the ally's base, the blue castle, while an adversarial opponent goes to the enemy's base, the red castle). The protagonist's objective is to identify the type of its opponent and obtain reward by tagging the adversarial opponent and let pass the neutral opponent. Mistakenly tagging a neutral would incur a large penalty to the protagonist. The protagonist cannot enter the blue region of the map, so once the opponent has passed the green region and entered the blue region, the protagonist cannot tag it anymore. If the opponent is an adversary,

it receives a large penalty if tagged. The opponent always receives penalty if it has not reached its base, and the penalty increases with its distance from its base. Based on the game rules, an adversarial opponent could try multiple strategies. For example, one strategy is to rush towards its home base to minimize the distance penalty. However, if it takes this greedy strategy, the protagonist can quickly identify this adversary and try to tag it (large penalty for the adversary). Another strategy is to initially head towards the ally base, to trick the protagonist agent into believing that the opponent is a neutral opponent. Once the adversary is close enough to the blue region, it can safely head to its base. This strategy incurs a larger distance penalty but might get a higher reward by avoiding being tagged.

State and Action Space The state of each agent is its 2-d position, i.e., $\mathcal{S}_i = [0, 8] \times [0, 8]$. The protagonist agent has a discrete action space $\mathcal{A}^p = [\textbf{move left}, \textbf{move right}, \textbf{move up}, \textbf{move down}, \textbf{tag}, \textbf{probe}]$, and the opponent agent's action space is $\mathcal{A}^o = [\textbf{move left}, \textbf{move right}, \textbf{move up}, \textbf{move down}]$.

Each of the 'move' action changes the agent position by one unit distance. The tag action succeeds if and only if the distance between the two agents is less than 2.5. The probe action is equivalent to query a noisy measurement of the opponent's true type, where there is 0.8 probability getting the correct type and 0.2 probability getting the wrong type. The protagonist agent could take this probe action to help with its inference besides simply observing the opponent. Each probe action incurs cost, so the protagonist agent has to wisely decide when and how many times to probe.

Reward The reward of the opponent agent consists of two parts: (1) $r_d = -0.25d^{2/5}$, where d is its distance from its home base; (2) $r_{\text{tagged}} = -10$ if being tagged.

The state-action reward of the protagonist agent consists of several parts: (1) $r_{\text{tag adversary}} = 10$ if tagging an adversarial agent; (2) $r_{\text{tag neutral}} = -20$ if tagging a neutral agent; (3) $r_{\text{d2o}} = -0.25d_{\text{o}}^{2/5}$, where d_{o} is the distance between the protagonist agent and the opponent. This is a heuristic reward to help the protagonist agent learning sensible behaviors; (4) Tag cost $r_{\text{tag cost}} = -0.2$, no matter tagging is successful or not; (5) Probe cost $r_{\text{probe cost}} = -0.25C$, where C is the cumulative counts of the probing action so far, i.e., the probe cost per time increases as the total number of probing increases. This effectively prevents the agent from abusing the probe action.

Ensemble Training vs. Single Model

To answer the first question, we compared the protagonist policy learned from training against an ensemble of opponent policies and that from training against a single opponent policy. We used a similar ensemble as used in (Khadka et al. 2019), which consists of four opponent policies, each policy is learned from training against the protagonist policy. We used four different discount factors for the opponent learning objectives: $\gamma_1 = 0.9, \gamma_2 = 0.99, \gamma_3 = 0.997, \gamma_4 = 0.9995$. An interpretation of this setting is a variety of opponent playing styles ranging from myopic to far-sighted strategies.

Algorithm*	Protagonist	Adversary
belief space, with EO & CE	**-14.4±1.49**	**-83.0±17.0**
LSTM, with EO & CE	-17.7±1.9	-66.2±13.8
belief space, w/o EO & CE	-16.5±1.1	-58.6±24.9
LSTM, w/o EO & CE	-16.8±3.1	-49.4±6.6

* EO: ensemble optimization, CE: cooperative evolution.

Table 2: Mean evaluation reward: vs. LSTM

For comparison, we also trained the protagonist policy individually against each opponent model, so we obtained five protagonist policies in total. For evaluation, we trained five separate opponent evaluation policies, each corresponding to one of the protagonist policies. The evaluation policies all used the same discount factor $\gamma = 0.99$.

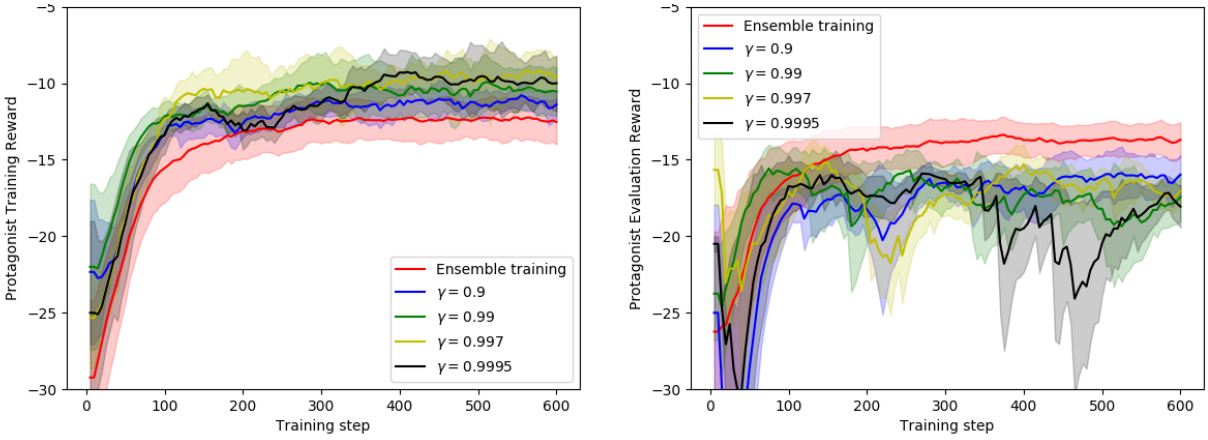
Fig. 3(a) and Fig. 3(b) (each curve is averaged over three runs) show the training and evaluation rewards. During training, the single model policies generally lead to higher protagonist reward, while the ensemble training results in the lowest protagonist reward. This suggests that the protagonist policy overfits to one of the single opponent models, thus achieving high training reward but low evaluation reward. In contrast, the protagonist policy trained against the ensemble achieves the best evaluation reward. It is worth pointing out that, in the second single model setting, although the hyper-parameter $\gamma_2 = 0.99$ is the same as that of the evaluation opponent, the evaluation reward is still significantly worse than the training reward. This is not surprising, as the agent could learn different policies even with the same hyper-parameter setting. Therefore, overfitting is almost inevitable when training against a single model.

Belief Space Policy vs. Implicit Approach via RNN

To answer the second question, we replaced belief space policy with a recurrent policy parameterized by a Long short term memory network (LSTM). Fig. 4 (histogram of rewards from 10 runs) and Table 2 (mean rewards) show the comparison between these two settings, where the belief space policy consistently outperforms the recurrent policy. This result agrees with our conjecture that learning a recurrent policy is difficult due to the lack of prior knowledge on the information structure and the high-variance of the state-space reward.

Ablation Study

To answer the third question, we compared our algorithm with its ablated versions: (I) without neuro-evolution, (II) without both neuro-evolution and ensemble optimization. For the ablated version II, we randomly sampled subsets of the ensemble from its powerset and used the fixed subset for training. We ran 10 independent simulations for each of the ablated version. Fig. 5 (histogram of rewards from 10 runs) and Table 3 (mean rewards) shows the evaluation rewards of the full and ablated versions of our algorithm. This result suggests that both neuro-evolution and ensemble optimization have important contributions to the performance improvement.



(a) Protagonist training reward

(b) Protagonist evaluation reward

Figure 3: Training and evaluation rewards of the protagonist agent: Single opponent models ($\gamma = 0.9$, $\gamma = 0.99$, $\gamma = 0.997$, $\gamma = 0.9995$) performs better than ensemble training in the training phase due to overfitting to simple opponent models, while ensemble training outperforms single opponent models in evaluation

Ablated version*	Protagonist	Adversary
with EO & CE	**-14.4±1.49**	**-83.0±17.0**
w/o EO	-15.5±2.2	-65.8±28.5
w/o EO & CE	-16.5±1.1	-58.6±24.9

* EO: ensemble optimization, CE: cooperative evolution.

Table 3: Mean evaluation reward: ablation study

Accuracy of Hidden Type Inference

To gain insight into the reason behind the results shown above, we present the accuracy of the opponent's hidden type inference under the different opponent modeling settings:

1. MDP-S: The baseline approach, where the opponent agent is modeled as a goal-directed MDP agent, and the optimal goal-achieving policy is used by the protagonist agent for opponent type inference.

2. GT-S: A game theoretic opponent modeling approach with a single opponent policy where the protagonist agent and the opponent concurrently update their policies during the training, and the protagonist agent uses this learned opponent model for type inference during the testing.

3. GT-E: A game theoretic opponent modeling approach with an ensemble of opponent policies, and the protagonist agent uses the distilled policy from this ensemble for type inference during the testing.

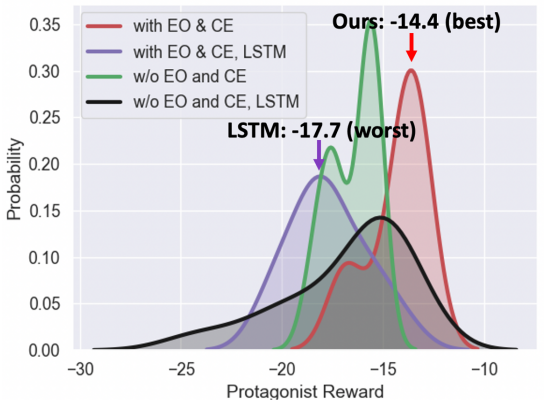
We randomly sampled the opponent type with an equal probability (50%) of neutral and adversary. Table 4 shows the average true positive rate (TPR, successfully identify an adversary), true negative rate (TNR, successfully identify a neutral), precision, recall, and mean rewards for both agents.

Metrics*/Approach	MDP-S	GT-S	GT-E
TPR10	0.12	0.34	**0.68**
TPR20	0.06	0.26	**0.74**
TNR10	0.91	0.73	0.85
TNR20	0.96	0.82	0.91
Precision10	0.57	0.56	**0.82**
Precision20	0.60	0.59	**0.89**
Recall10	0.12	0.34	**0.68**
Recall20	0.06	0.26	**0.74**
Protagonist reward	-19.44	-17.55	**-14.4**
Adversary reward	-50.48	-58.19	**-83.0**

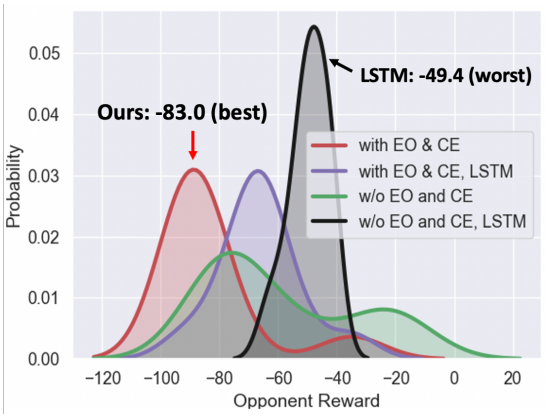
* TRPn: TPR after n time steps, similarly for the other metrics.

Table 4: Opponent type inference accuracy / Rewards

The TPR corresponding to MDP-S is quite low, indicating that the protagonist agent completely mis-classifies the adversary agent as a neutral agent. As the time step increases (comparison between TPR10 and TPR20), this mis-classification becomes worse (from 0.12 to 0.06). This result indicates that there is a significant mismatch between the protagonist's model of the adversary policy and the actual adversary policy during the testing. Besides, the true negative rate is quite high. This result suggests that the adversary indeed learned a policy that confuses the protagonist agent by mimicking the behavior of a neutral agent, and therefore, the protagonist agent always classifies the opponent as a neutral agent. Similarly, the TPR corresponds to GT-S is also significantly lower than the prior probability (50%), which indicates a mismatch between the opponent model and the actual opponent, while GT-S outperforms MDP-S by a large margin since the game-theoretic aspect accounts for the strategic reasoning between the adversary and the protagonist agent which is missing in the MDP-S approach. In contrast, the TPR corresponding to GT-E significantly outperforms the other two approaches. Moreover, as the time step increases, the TPR also increases

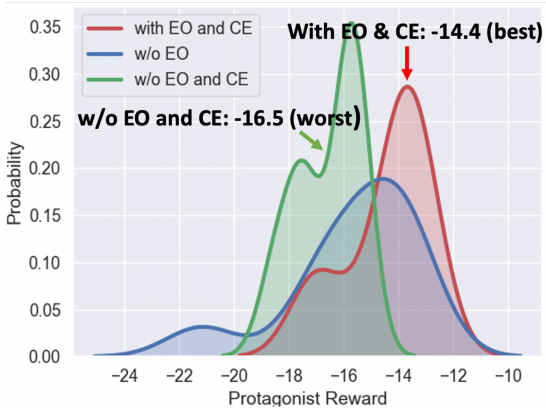


(a) Protagonist reward: higher is better

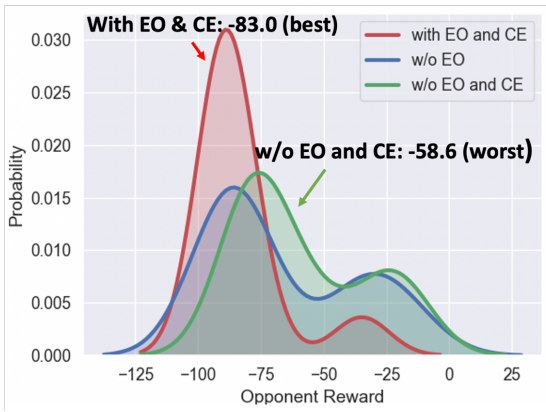


(b) Opponent (adversary) reward: lower is better

Figure 4: Evaluation rewards distribution of the protagonist agent (top) and the opponent agent (bottom): (1) belief space policy, with ensemble optimization (EO) and cooperative-evolution (CE); (2) LSTM, with EO and CE; (3) belief space policy, without EO and CE (single opponent model); (4) LSTM, without EO and CE (single opponent model); Belief space policy (1) and (3) outperforms LSTM (2) and (4)



(a) Protagonist reward: higher is better



(b) Opponent (adversary) reward: lower is better

Figure 5: Evaluation rewards of the protagonist agent (top) and the opponent agent (bottom): (1) with both ensemble optimization (EO) and cooperative-evolution (CE); (2) with CE but without EO; (3) without EO and CE (single opponent model); EO + CE outperforms CE only, which outperforms single opponent model

from 0.68 to 0.74, which indicates that this opponent model successfully captures an adversary's general behavior pattern and is able to generalize to previously unseen adversaries.

The TNR statistics of these three modeling approaches are all significantly higher than 50%, which indicates a good success rate of identifying a neutral agent. This result is also anticipated due to the fact that the neutral agent's policy is simple (heading towards its goal), so it can be accurately learned by the protagonist agent. Although the TNR of MDP-S is highest among these three modeling approaches, this high TNR does not necessarily indicate MDP-S is good at identifying the neutral agent, but rather always 'guessing' the opponent as being neutral. As a result, the recall scores of the MDP-S and the GT-S approaches are much lower than that of the GT-E approach, and consequently lead to a poorer protagonist agent performance as indicated by the rewards.

Related Works

Our work is at the intersection of hidden-information/(hidden-role) games, robust MARL, and adversarial attack. The DeepRole algorithm (Serrino et al. 2019) is the first deep MARL approach for hidden role games, to the best of the authors' knowledge. It combines counterfactual regret minimization (CFR) with deep value networks trained through self-play and integrates deductive reasoning into the RL module to reason about joint beliefs and deduce partially observable actions. Our work is similar to (Serrino et al. 2019) in terms of opponent modeling: both works learn deep policy through self-play and explicitly infer opponent type using the learned policy. However, (Serrino et al. 2019) did not explicitly consider the robustness of the learned policy, while we demonstrated in our example that robustness is a critical issue in hidden role games, and proposed solutions to effectively improve the robustness through ensemble training.

Ensemble training techniques for robust MARL have been

developed/applied in (Lowe et al. 2017), (Bansal et al. 2017), (Jaderberg et al. 2017), (Jaderberg et al. 2018), (Schrittwieser et al. 2019). Among these works, (Schrittwieser et al. 2019) is the only one that actively optimizes the ensemble within the population-based-training (PBT) (Jaderberg et al. 2017) framework. In (Schrittwieser et al. 2019), the main exploiters trained against their main agents and the league exploiters trained against all past players play a similar role as the ensemble evaluation procedure does in our work. That architecture has achieved considerable improvements in the robustness, leading to superhuman level performance in StarCraft II, but the associated significant increase in complexity makes this technique impractical for most implementations. In contrast, our ensemble optimization scheme requires much less computation.

Adversarial attacks against deep neural network and countermeasures have also been widely studied, e.g., in (Akhtar and Mian 2018), (Yuan et al. 2019), (Gleave et al. 2019a), (Ilahi et al. 2020). The works on adversarial attacks in RL mostly focus on different types of problems where the adversary can manipulate the reward (Han et al. 2018), policy (Huang et al. 2017), observation (Behzadan and Munir 2017) or environment (Chen et al. 2018) of the protagonist agent. Our work is different from these bodies of work, in that the adversary is unable to directly manipulate the reward, policy or environment. Our work is based on a similar assumption as in (Gleave et al. 2019b): the adversary cannot directly manipulate the protagonist's observation but can carefully choose an adversarial policy to act in the multi-agent environment to create natural adversarial observations. In (Gleave et al. 2019b), the authors showed that in a humanoid robot domain, the adversary can learn policies that reliably win against the protagonist agent but generate seemingly random and uncoordinated behavior (feature-level attack), which induce substantially different activations in the protagonist policy network than when the protagonist agent plays against a normal opponent. Also, these adversarial policies are more successful in high-dimensional environments. In contrast, our work focuses on scenarios where the adversarial attack is on the strategic level (adversary type hidden, carefully chooses action from low-dimensional discrete action space that maneuvers the protagonist agent's belief). In addition, the main focus of our work is on developing an inference scheme on the hidden type of the opponent and a robust policy learning algorithm in MARL, while (Gleave et al. 2019b) focuses on investigating the possibility of learning an adversarial policy against a fixed victim, which is essentially a single-agent problem.

The scope of this work is also closely related with multiagent reasoning and goal recognition, where the standard assumption is the availability of a library of action models (one action model is analogous to one single policy of a certain opponent type in our work), as in (Fagan and Cunningham 2003), (Sohrabi, Riabov, and Udrea 2016). Therefore, our MDP-S baseline corresponds to goal recognition with a library of MDP action models, and the GT-S baseline corresponds to goal recognition with a library of game-theoretic action models, which requires a game-solver for bayesian games. This requirement is non-trivial with planning-based approaches. As a result, planning-based multiagent reasoning was mostly studied in very restricted domains (e.g., matrix games as in (Huang and Zhu 2018); two-stage games as (Nguyen et al. 2019)), while our approach is more scalable. Besides, we demonstrated that game-theoretic modeling alone is insufficient for learning robust policy, and our results show that the ensemble training is critical for improving robustness of policy against adversarial exploitation, which has rarely been explored in planning-based multiagent reasoning works.

Summary

We summarize the key findings of this work as follows:

- We present algorithms based on MARL and ensemble training for robust opponent modeling and posterior inference over the opponent type from the observed action.
- We demonstrate that the explicit opponent modeling outperforms a black-box RNN approach, and the ensemble training approach outperforms a single agent model. We analyze the reason for this observation by inspecting the agent type inference, and show that the performance of the protagonist agent policy is highly correlated to the quality of the type inference accuracy.

Acknowledgements

This research is supported by Scientific Systems Company, Inc. The authors would like to thank Chuangchuang Sun and Kasra Khosoussi for their insightful discussions and Amazon Web Services for computation supports.

References

Akhtar, N.; and Mian, A. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6: 14410–14430.

Andrychowicz, M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. 2018. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177* .

Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; and Mordatch, I. 2017. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748* .

Bard, N.; Johanson, M.; Burch, N.; and Bowling, M. 2013. Online implicit agent modelling. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 255–262.

Behzadan, V.; and Munir, A. 2017. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, 262–275. Springer.

Bjarnason, R. V.; and Peterson, T. S. 2002. Multi-agent learning via implicit opponent modeling. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, 1534–1539. IEEE.

Carmel, D.; and Markovitch, S. 1998. Model-based learning of interaction strategies in multi-agent systems. *Journal*

of Experimental & Theoretical Artificial Intelligence 10(3): 309–332.

Chakraborty, D.; and Stone, P. 2014. Multiagent learning in the presence of memory-bounded agents. *Autonomous agents and multi-agent systems* 28(2): 182–213.

Chen, T.; Niu, W.; Xiang, Y.; Bai, X.; Liu, J.; Han, Z.; and Li, G. 2018. Gradient band-based adversarial training for generalized attack immunity of a3c path finding. *arXiv preprint arXiv:1807.06752* .

Cheon, T.; and Iqbal, A. 2008. Bayesian Nash equilibria and Bell inequalities. *Journal of the Physical Society of Japan* 77(2): 024801.

Fagan, M.; and Cunningham, P. 2003. Case-based plan recognition in computer games. In *International Conference on Case-Based Reasoning*, 161–170. Springer.

Gleave, A.; Dennis, M.; Kant, N.; Wild, C.; Levine, S.; and Russell, S. 2019a. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615* .

Gleave, A.; Dennis, M.; Kant, N.; Wild, C.; Levine, S.; and Russell, S. 2019b. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615* .

Han, Y.; Rubinstein, B. I.; Abraham, T.; Alpcan, T.; De Vel, O.; Erfani, S.; Hubczenko, D.; Leckie, C.; and Montague, P. 2018. Reinforcement learning for autonomous defence in software-defined networking. In *International Conference on Decision and Game Theory for Security*, 145–165. Springer.

Hausknecht, M.; and Stone, P. 2015. Deep recurrent Q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

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.

Huang, L.; and Zhu, Q. 2018. Dynamic Bayesian Games for Adversarial and Defensive Cyber Deception. *ArXiv* abs/1809.02013.

Huang, S.; Papernot, N.; Goodfellow, I.; Duan, Y.; and Abbeel, P. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* .

Ilahi, I.; Usama, M.; Qadir, J.; Janjua, M. U.; Al-Fuqaha, A.; Hoang, D. T.; and Niyato, D. 2020. Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning. *arXiv preprint arXiv:2001.09684* .

Jaderberg, M.; Czarnecki, W. M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A. G.; Beattie, C.; Rabinowitz, N. C.; Morcos, A. S.; Ruderman, A.; et al. 2018. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281* .

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* .

Khadka, S.; Majumdar, S.; Miret, S.; Tumer, E.; Nassar, T.; Dwiel, Z.; Liu, Y.; and Tumer, K. 2019. Collaborative evolutionary reinforcement learning. *arXiv preprint arXiv:1905.00976* .

Lockett, A. J.; Chen, C. L.; and Miikkulainen, R. 2007. Evolving explicit opponent models in game playing. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2106–2113.

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.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* .

Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356(6337): 508–513.

Nguyen, T. H.; Wang, Y.; Sinha, A.; and Wellman, M. P. 2019. Deception in Finitely Repeated Security Games 33: 2133–2140.

OpenAI. 2018. OpenAI Five. *OpenAI blog* URL https://blog.openai.com/openai-five/.

Rubin, J.; and Watson, I. 2011. Implicit opponent modelling via dynamic case-base selection. In *Workshop on case-based reasoning for computer games at the 19th international conference on case-based reasoning*, 63–71.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2019. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265* .

Serrino, J.; Kleiman-Weiner, M.; Parkes, D. C.; and Tenenbaum, J. 2019. Finding Friend and Foe in Multi-Agent Games. In *Advances in Neural Information Processing Systems*, 1249–1259.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan Recognition as Planning Revisited. In *IJCAI*, 3258–3264.

Teh, Y.; Bapst, V.; Czarnecki, W. M.; Quan, J.; Kirkpatrick, J.; Hadsell, R.; Heess, N.; and Pascanu, R. 2017. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, 4496–4506.

Yuan, X.; He, P.; Zhu, Q.; and Li, X. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems* 30(9): 2805–2824.