A DEEP REINFORCEMENT LEARNING APPROACH FOR FINDING NON-EXPLOITABLE STRATEGIES IN TWO-PLAYER ATARI GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper proposes novel, end-to-end deep reinforcement learning algorithms for learning two-player zero-sum *Markov games*. Different from prior efforts on training agents to beat a fixed set of opponents, our objective is to find the *Nash equilibrium* policies that are free from exploitation by even the adversarial opponents. We propose (1) Nash DQN algorithm, which integrates DQN (Mnih et al., 2013) with a Nash finding subroutine for the joint value functions; and (2) Nash DQN Exploiter algorithm, which additionally adopts an exploiter for guiding agent's exploration. Our algorithms are the practical variants of theoretical algorithms which are guaranteed to converge to Nash equilibria in the basic tabular setting. Experimental evaluation on both tabular examples and two-player Atari games demonstrates the robustness of the proposed algorithms against adversarial opponents, as well as their advantageous performance over existing methods.

1 INTRODUCTION

Reinforcement learning (RL) in multi-agent systems has succeeded in many challenging tasks, including Go (Silver et al., 2017), hide-and-seek (Baker et al., 2019), Starcraft (Vinyals et al., 2017), Dota (Berner et al., 2019), Poker (Heinrich & Silver, 2016; Brown & Sandholm, 2019; Zha et al., 2021), and board games (Lanctot et al., 2019; Serrino et al., 2019). Excluding the systems for Poker, a large number of these works measure their success in terms of performance against fixed agents, average human players or experts in a few shots. A distinguishing feature of games is that the opponents can further model the learner's behaviors, adapt their strategies, and exploit the learner's weakness. It is highly unclear whether the policies found by many of these multi-agent systems remain viable against the adversarial exploitation of the opponents.

In this paper, we consider two-player zero-sum *Markov games* (MGs), and our objective is to find the *Nash Equilibrium* (NE) (Nash et al., 1950). By definition, the NE strategy is a stationary point where no player has the incentive to deviate from its current strategy. Due to the minimax theorem, the NE strategy for one player is also the best solution when facing against the best response of the opponent. That is, NE is a natural solution that is free from the exploitation by adversarial opponents.

The concepts of Nash equilibrium and non-exploitability have been well studied in the community of learning *extensive-form games* (EFGs) such as Poker (Heinrich & Silver, 2016; Brown & Sandholm, 2019; Zha et al., 2021; McAleer et al., 2021). Distinct from EFGs which feature tree-structured transition dynamics and do not *efficiently* represent the games where multiple states in the past may lead to the same state in the future, this paper focuses on MGs with general transition structure, and leverages the Markov structures. Another line of prior works (Heinrich & Silver, 2016; Lanctot et al., 2017) directly combine the best-response-based algorithm for finding NE in normal-form games, such as fictitious play (Brown, 1951) and double oracle (McMahan et al., 2003), with single-agent deep RL algorithm such as DQN (Mnih et al., 2013) and PPO (Schulman et al., 2017) for finding the best-response. While these approaches can be applied to MGs, they do not utilize the fine structure of MGs beyond treating it as normal-form games, which leads to the significant inefficiency in learning (as shown in both tabular and Atari experiments of this paper).

This paper proposes two novel, end-to-end deep reinforcement learning algorithms for learning the Nash equilibrium of two-player zero-sum MGs—Nash Deep Q-Network (NASH_DQN), and its variant Nash Deep Q-Network with Exploiter (NASH_DQN_EXPLOITER). NASH_DQN combines the recent theoretical progress for learning Nash equilibria of tabular MGs (Hu & Wellman, 2003; Bai & Jin, 2020; Liu et al., 2021; Jin et al., 2021b), with well-known single-agent DRL algorithm DQN (Mnih et al., 2013) for addressing continuous state space and function approximation.

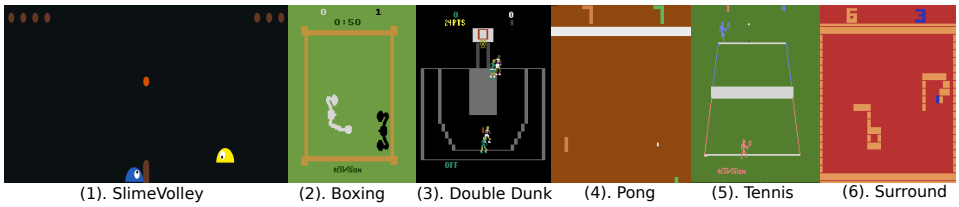

Figure 1: Screen shots of the six two-player video games.

NASH_DQN_EXPLOITER is a variant of NASH_DQN which explicitly train an exploiting opponent during its learning. The exploiting opponent stimulates the exploration for the main agent. Both algorithms are the practical variants of theoretical algorithms which are guaranteed to converge to Nash equilibria in the basic tabular setting.

Experimental evaluations are conducted on both tabular MGs and two-player video games, to show the effectiveness and robustness of the proposed algorithms. As shown in Fig. 1, the video games in our experiments include five two-player Atari games in PettingZoo library (Terry et al., 2021; Bellemare et al., 2013) and a benchmark environment Slime Volley-Ball (Ha, 2020). Due to the constraints of computational resource, we consider the RAM-based version of Atari games, and truncate the length of each game to 300 steps. We test the performance by training adversarial opponents using DQN that directly exploit the learner's policies. Our experiments in both settings show that our algorithms significantly outperform standard algorithms for MARL including Neural Fictitious Self-Play (NFSP) (Heinrich & Silver, 2016) and Policy Space Response Oracle (PSRO) (Lanctot et al., 2017), in terms of the robustness against adversarial exploitation. The implementation is also released[1]

1.1 RELATED WORK

MARL in IIEFGs. This line of work (see, e.g., Lanctot et al., 2009; Heinrich et al., 2015; Brown & Sandholm, 2019; Farina et al., 2020; McAleer et al., 2021; Kozuno et al., 2021; Bai et al., 2022) focuses on learning Imperfect Information Extensive-Form Games (IIEFGs). This line of results focuses on finding Nash equilibrium, which is non-exploitable. However, a majority of these results focus on the settings and applications (such as Poker) where state space is discrete. More importantly, comparing to MGs, the model of IIEFGs made a strong assumption of transition which must be tree-formed. The results of IIEFG typically scales super-linearly with respect to the number of information sets, which in general grows exponentially with the horizon length of the game.

MARL in zero-sum MGs. There has not been much prior empirical efforts to design algorithms specializing in solving zero-sum MGs. However, there is a rich class of empirical algorithms that can be directly applied to this setting. These algorithms involve combining single-agent RL algorithm, such as deep Q-network (DQN) (Mnih et al., 2013) or proximal policy optimisation (PPO) (Schulman et al., 2017), with best-response-based Nash equilibrium finding algorithm for *normal-form* games, including fictitious play (FP) (Brown, 1951), double oracle (DO) (McMahan et al., 2003), and many others. A few other examples such as neural fictitious self-play (NFSP) (Heinrich & Silver, 2016), policy space response oracles (PSRO) (Lanctot et al., 2017), online double oracle (Dinh et al., 2021) and prioritized fictitious self-play (Vinyals et al., 2019) also in general fall into this class of algorithms or their variants. These algorithms call single-agent RL algorithm to compute the best response of the current "meta-strategy", and then use the best-response-based algorithm for normal-form games to compute a new "meta-strategy". However, these algorithms inherently treat MGs as normal-form games, do not efficiently utilize the finer structure within MGs. Specifically, for normal-form game, FP has a convergence rate exponential in the number of actions, while DO is linear. However, a direct converting of Markov game to normal-form game will generate a new action space **exponential** in horizon, number of states and number of actions in original Markov game. This, as shown in our experiments, leads to significant inefficiency in scaling up with size of the MGs.

On the other hand, there has been rich studies on two-player zero-sum MGs from the theoretical perspectives. Many of these works (see, e.g., Hu & Wellman, 2003; Bai & Jin, 2020; Bai et al., 2020; Liu et al., 2021; Jin et al., 2021a) focused on the tabular setting, which requires the numbers of states and actions to be finite. These algorithms are proved to converge to the NE policies in a number of samples that is **polynomial** in the number of states, actions, horizon (or the discount coefficient), and the target accuracy. Among those, Nash Q-learning (Hu & Wellman, 2003) is one of the earliest works along this line of research, which provably converges to NE for general-sum games

[1]The code is available at here (click).

under the assumption that the NE is unique for each stage game during the learning process. On the other hand, GOLF_WITH_EXPLOITER (Jin et al., 2021b) is another theoretical work with provable polynomial convergence for two-player zero-sum MGs. Our NASH_DQN algorithm is designed based on the provable tabular algorithm Nash Value Iteration (Liu et al., 2021), which is a natural extension of value iteration algorithm from single-agent setting to the multi-agent setting. For better understanding, we provide a detailed comparison of similarities and differences of Nash Q-learning, Nash Value Iteration, GOLF_WITH_EXPLOITER and NASH_DQN in the Appendix B.11. Xie et al. (2020) considers MGs with linear function approximation. There are a few theoretical works on studying zero-sum MGs with general function approximation (Jin et al., 2021b; Huang et al., 2021), which include neural network function approximation as special cases. However, these algorithms are sample-efficient, but not computationally efficient. They require solving optimistic policies with complicated confidence sets as constraints, which cannot be run in practice.

2 PRELIMINARIES

In this paper, we consider Markov Games (MGs, Shapley, 1953; Littman, 1994), which generalizes standard Markov Decision Processes (MDPs) into multi-player settings. Each player has its own utility and optimize its policy to maximize the utility. We consider a special setting in MG called two-player zero-sum games, which has a competitive relationship between the two players.

More concretely, consider a infinite-horizon discounted version of two-player zero-sum MG, which is denoted as $\mathrm{MG}(\mathcal{S}, \mathcal{A}, \mathcal{B}, \mathbb{P}, r, \gamma)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ and $\mathcal{B}$ are the action spaces for the max-player and min-player respectively. $\mathbb{P}(\cdot|s,a,b)$ is the state transition distribution, $r\colon \mathcal{S}\times\mathcal{A}\times\mathcal{B}\to\mathbb{R}$ is the reward function. In the zero-sum setting, the reward is the gain for the max-player and the loss for the min-player due to the zero-sum payoff structure. $\gamma\in[0,1]$ is the discount factor.

Policy, value function. We define the policy and value functions for each player. For the max-player, the (Markov) policy is a map $\mu:\mathcal{S}\to\Delta_{\mathcal{A}}$. Here we only consider discrete action space, so $\Delta_{\mathcal{A}}$ is the probability simplex over action set $\mathcal{A}$. Similarly, the policy for the min-player is $\nu:\mathcal{S}\to\Delta_{\mathcal{B}}$.

$V^{\mu,\nu}\colon\mathcal{S}\to\mathbb{R}$ represents the value function evaluated with policies μ and ν, which can be expanded as the expected cumulative reward starting from the state s,

$$V^{\mu,\nu}(s) := \mathbb{E}_{\mu,\nu}\bigg[\sum_{h=1}^{\infty}\gamma^{h-1}r(s_h,a_h,b_h)\Big|s_1=s\bigg]. \tag{1}$$

Correspondingly, $Q^{\mu,\nu}:\mathcal{S}\times\mathcal{A}\times\mathcal{B}\to\mathbb{R}$ is the state-action value function evaluated with policies μ and ν, which can also be expanded as expected cumulative rewards as:

$$Q^{\mu,\nu}(s,a,b) := \mathbb{E}_{\mu,\nu}\bigg[\sum_{h=1}^{\infty}\gamma^{h-1}r(s_h,a_h,b_h)\big|s_1=s,a_1=a,b_1=b\bigg]. \tag{2}$$

In this paper we also use a simplified notation for convenience, $[\mathbb{P}V](s,a,b) := \mathbb{E}_{s'\sim\mathbb{P}(\cdot|s,a,b)}V(s')$, where $\mathbb{P}$ as the transition function can be viewed as an operator. Similarly, we denote $[\mathbb{D}_\pi Q](s) := \mathbb{E}_{(a,b)\sim\pi(\cdot,\cdot|s)}Q(s,a,b)$ for any state-action value function. In this way, the Bellman equation for two-player MG can be written as the following, for $\forall(s,a,b)\in\mathcal{S}\times\mathcal{A}\times\mathcal{B}$,

$$Q^{\mu,\nu}(s,a,b) = (r+\gamma\mathbb{P}V^{\mu,\nu})(s,a,b),\quad V^{\mu,\nu}(s) = (\mathbb{D}_{\mu\times\nu}Q^{\mu,\nu})(s). \tag{3}$$

Best response and Nash equilibrium. In two-player games, if the other player always play a fixed Markov policy, optimizing over learner's policy is the same as optimizing over the policy of single agent in MDP (with other players' polices as a part of the environment). For two-player cases, given the max-player's policy μ, there exists a *best response* of the min-player, which is a policy $\nu^\dagger(\mu)$ satisfying $V^{\mu,\nu^\dagger(\mu)}(s) = \inf_\nu V^{\mu,\nu}(s)$ for any $s\in\mathcal{S}$. We simplify the notation as: $V^{\mu,\dagger} := V^{\mu,\nu^\dagger(\mu)}$. Similar best response for a given min-player's policy ν also exists as $\mu^\dagger(\nu)$ satisfying $V^{\dagger,\nu} = \sup_\mu V^{\mu,\nu}$. By leveraging the Bellman equation as Eq. (3), the best response can be derived with dynamic programming,

$$Q^{\mu,\dagger}(s,a,b) = (r+\gamma\mathbb{P}V^{\mu,\dagger})(s,a,b),\quad V^{\mu,\dagger}(s) = \inf_\nu(\mathbb{D}_{\mu\times\nu}Q^{\mu,\dagger})(s) \tag{4}$$

The *Nash equilibrium* (NE) is defined as a pair of policies $(\mu^\star,\nu^\star)$ satisfying the following minimax equation:

$$\sup_\mu\inf_\nu V^{\mu,\nu}(s) = V^{\mu^\star,\nu^\star}(s) = \inf_\nu\sup_\mu V^{\mu,\nu}(s). \tag{5}$$

which is similar as the normal-form game but without the bilinear structure of the payoff matrix. NE strategies are the ones where no player has incentive to change its own strategy. The value functions of $(\mu^\star, \nu^\star)$ is denoted as $V^\star$ and $Q^\star$, which satisfy the following Bellman optimality equation:

$$\begin{aligned} Q^\star(s,a,b) &= (r + \gamma \mathbb{P} V^{\mu,\dagger})(s,a,b) \\ V^\star(s) &= \sup_{\mu\in\Delta_{\mathcal{A}}} \inf_{\nu\in\Delta_{\mathcal{B}}} (\mathbb{D}_{\mu\times\nu} Q^\star)(s) = \inf_{\nu\in\Delta_{\mathcal{B}}} \sup_{\mu\in\Delta_{\mathcal{A}}} (\mathbb{D}_{\mu\times\nu} Q^\star)(s). \end{aligned} \tag{6}$$

Learning Objective. The *exploitability* of policy $(\hat{\mu}, \hat{\nu})$ can be defined as the difference in values comparing to Nash strategies when playing against their best response. Formally, the exploitability of the max-player can be defined as $V^\star(s_1) - V^{\hat{\mu},\dagger}(s_1)$ while the exploitability of the min-player is defined as $V^{\dagger,\hat{\nu}}(s_1) - V^\star(s_1)$. We define the total suboptimality of $(\hat{\mu}, \hat{\nu})$ simply as the summation of the exploitability of both players

$$V^{\dagger,\hat{\nu}}(s_1) - V^{\hat{\mu},\dagger}(s_1) = \big[V^{\dagger,\hat{\nu}}(s_1) - V^\star(s_1)\big] + \big[V^\star(s_1) - V^{\hat{\mu},\dagger}(s_1)\big]. \tag{7}$$

which is the duality gap as a distance measure to Nash equilibria. We note that the duality gap of Nash equilibria is equal to zero. Furthermore, all video games we conduct experiments on are symmetric to two players, which implies that $V^\star(s_1) = 0$.

NASH_VI and Nash Q-Learning. As a model-based algorithm, NASH_VI (Liu et al., 2021) estimates the transition and reward functions (omitted here) in Eq. (6) with the collected samples, for $\forall(s,a,b,s') \in \mathcal{S}_h \times \mathcal{A}_h \times \mathcal{B}_h \times \mathcal{S}_{h+1}, h \in [H]$:

$$\tilde{\mathbb{P}}_h(s_{h+1} = s' | s_h = s, a_h = a, b_h = b) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(s_{h+1} = s'_i) \tag{8}$$

Nash Q-Learning (Hu & Wellman, 2003) is the model-free version of NASH_VI. Pseudo-codes and discussions about NASH_VI and Nash Q-Learning are provided in Appendix B.4, B.8 and B.11.

3 METHODOLOGY

To learn the Nash equilibria of two-player zero-sum MGs, this paper proposes two novel, end-to-end deep MARL algorithms—NASH_DQN and NASH_DQN_EXPLOITER. NASH_DQN combines single-agent DQN (Mnih et al., 2013) with NASH_VI—a provable algorithm for tabular MGs. NASH_DQN_EXPLOITER is a variant of NASH_DQN by explicitly training an adversarial opponent during the learning phase to encourage the exploration of the learning agent.

3.1 NASH_DQN

We describe NASH_DQN in Algorithm 1 which incorporates neural networks into the tabular NASH_VI algorithm for approximating the Q-value function. Similar to the single-agent DQN, NASH_DQN maintains two networks in the training process: the Q-network and its target network, which are parameterized by ϕ and ϕ^{target}, respectively. In each episode, NASH_DQN executes the following two main steps:

• **Data collection:** NASH_DQN adopts the ϵ-greedy strategy for exploration. At each state s_t, with probability ϵ, both players take random actions; otherwise, they will sample actions (a_t, b_t) from the Nash equilibrium of its Q-value matrix (i.e., $\text{NASH}(Q_\phi(s_t, \cdot, \cdot))$, see (9)). After that, we add the collected data into the experience replay buffer $\mathcal{D}$.

• **Model update**: We first randomly sample a batch of data $\mathcal{M}$ from replay buffer $\mathcal{D}$, and then perform m-steps gradient descent to update ϕ using the loss below

$$\textstyle\sum_{j\in\mathcal{M}} \left(Q_\phi(s_j, a_j, b_j) - y_j\right)^2,$$

where target y_j is computed according to line 14. We adopt the convention of setting $Q_{\phi^{\text{target}}}(s_{j+1}, \cdot, \cdot) = 0$ for all terminal state s_{j+1}.

Here, $\text{NASH}(\cdot)$ is the NE subroutine for normal-form games, which takes a payoff matrix $\mathbf{A} \in \mathbb{R}^{A\times B}$ as input and outputs one of its Nash equilibria $(\mu^\star, \nu^\star)$. In math, we have:

$$(\mu^\star, \nu^\star) = \text{NASH}(\mathbf{A}) \quad \text{if and only if} \quad \forall \mu, \nu, \;\; \mu^\top \mathbf{A} \nu^\star \le (\mu^\star)^\top \mathbf{A} \nu^\star \le (\mu^\star)^\top \mathbf{A} \nu. \tag{9}$$

Algorithm 1 Nash Deep Q-Network (NASH_DQN)

1: Initialize replay buffer $\mathcal{D} = \emptyset$, counter $i = 0$, Q-network Q_ϕ
2: Initialize target network parameters: $\phi^{\text{target}} \leftarrow \phi$.
3: **for** episode $k = 1, \ldots, K$ **do**
4: reset the environment and observe s_1.
5: **for** $t = 1, \ldots, H$ **do**
6: % collect data
7: sample actions (a_t, b_t) from $\begin{cases} \text{Uniform}(\mathcal{A} \times \mathcal{B}) & \text{with probability } \epsilon \\ (\mu_t, \nu_t) = \text{NASH}(Q_\phi(s_t, \cdot, \cdot)) & \text{otherwise.} \end{cases}$
8: execute actions (a_t, b_t), observe reward r_t, next state s_{t+1}.
9: store data sample $(s_t, a_t, b_t, r_t, s_{t+1})$ into $\mathcal{D}$
10: % update Q-network
11: randomly sample minibatch $\mathcal{M} \subset \{1, \ldots, |\mathcal{D}|\}$.
12: **for** all $j \in \mathcal{M}$ **do**
13: compute $(\hat{\mu}, \hat{\nu}) = \text{NASH}(Q_{\phi^{\text{target}}}(s_{j+1}, \cdot, \cdot))$
14: set $y_j = r_j + \gamma \hat{\mu}^\top Q_{\phi^{\text{target}}}(s_{j+1}, \cdot, \cdot)\hat{\nu}$.
15: Perform m steps of GD on loss $\sum_{j \in \mathcal{M}} (y_j - Q_\phi(s_j, a_j, b_j))^2$ to update ϕ.
16: % update target network
17: $i = i + 1$; if $i \% N = 0$: $\phi^{\text{target}} \leftarrow \phi$.

There are several off-the-shelf libraries to implement this NASH subroutine. After comparing the performance of several different implementations (see Appendix A for details), we found the ECOS library (Domahidi et al., 2013) works the best, which from now on is set as the default choice in our algorithms.

Regarding the choice of target value update (line 14), one can view it as a Monte Carlo estimate of

$$r(s_j, a_j, b_j) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s_j, a_j, b_j)} [\max_{\hat{\mu} \in \Delta_A} \min_{\hat{\nu} \in \Delta_B} \hat{\mu}^\intercal Q_{\phi^{\text{target}}}(s', \cdot, \cdot)\hat{\nu}]. \tag{10}$$

Intuitively, we aim to approximate the Q-value function of Nash equilibria $Q^\star$ by our Q-network Q_ϕ. Recall that $Q^\star$ is the *unique* solution of the Bellman optimality equations:

$$\forall (s, a, b), \quad Q^\star(s, a, b) = r(s, a, b) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a,b)} [\max_{\hat{\mu} \in \Delta_A} \min_{\hat{\nu} \in \Delta_B} \hat{\mu}^\intercal Q^\star(s', \cdot, \cdot)\hat{\nu}]. \tag{11}$$

As a result, by performing gradient descent on ϕ to minimize the square loss as in line 15, Q_ϕ will decrease its Bellman error, and eventually converge to $Q^\star$, as more samples are collected. Finally, we remark that the target Nash Q-network ($Q_{\phi^{\text{target}}}$) is updated in a delayed manner as DQN to stabilize the training process.

3.2 NASH_DQN_EXPLOITER

NASH_DQN relies on the ϵ-greedy strategy for exploration. To improve the exploration efficiency, we propose a variant of NASH_DQN— NASH_DQN_EXPLOITER, which additionally introduces an exploiter in the training procedure. By constantly exploiting the weakness of the main agent, the exploiter forces the main agent to play the part of the games she is still not good at, and thus helps the main agent improve and discover more effective strategies.

We describe NASH_DQN_EXPLOITER in AppendixB.10 Algorithm 11. We let the main agent maintain a Q-network Q_ϕ and let the exploiter maintain a separate value network $\tilde{Q}_\psi$, both are functions of (s, a, b). We make two key modifications from NASH_DQN. First, in the data collection phase, at state s_t, we no longer choose both μ_t, ν_t to be the Nash equilibrium computed from Q_ϕ. Instead, we only choose μ_t to be the Nash strategy of Q_ϕ but pick the policy of the exploiter ν_t to be the best response of μ_t under the exploiter's Q-network $\tilde{Q}_\psi$. Formally,

$$\begin{aligned} (\mu_t, \cdot) &= \text{NASH}(Q_\phi(s_t, \cdot, \cdot)) \\ \nu_t &= \operatorname*{argmin}_\nu \mu_t^\top \tilde{Q}_\psi(s_t, \cdot, \cdot)\nu. \end{aligned} \tag{12}$$

In the model update phase, NASH_DQN_EXPLOITER follows exactly the same rule as NASH_DQN to update Q_ϕ and $Q_{\phi^{\text{target}}}$, the Q-networks of the main agent. However, for the update of the exploiter networks, NASH_DQN_EXPLOITER utilizes a different regression target in the loss function as specified in line 15. One can view the target as a Monte Carlo estimate of

$$r(s_j, a_j, b_j) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s_j, a_j, b_j)} \left[\min_b \hat{\mu}(s')^\intercal \tilde{Q}_{\psi^{\text{target}}}(s', \cdot, b) \right] \tag{13}$$

We set the target in this way because we aim to approximate $Q^{\hat{\mu},\dagger}$, which is the value of the current policy of the main player $\hat{\mu}$ against its best response, using our exploiter network $\tilde{Q}_\psi$. Recall that $Q^{\hat{\mu},\dagger}$ satisfies the following Bellman equations for the best response:

$$\forall(s,a,b), \quad Q^{\hat{\mu},\dagger}(s,a,b) = r(s,a,b) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a,b)}[\min_b \hat{\mu}(s')^\intercal Q^{\hat{\mu},\dagger}(s',\cdot,b)]. \tag{14}$$

Therefore, by performing gradient descent on ψ to minimize the square loss as in line 17, $\tilde{Q}_\psi$ will decrease its (best response version of) Bellman error, and approximate $Q^{\hat{\mu},\dagger}$.

3.3 Theoretical Justification

With special choices of Q-network architecture Q_ϕ, minibatch size $|\mathcal{M}|$ and number of steps for GD m, both our algorithms NASH_DQN and NASH_DQN_EXPLOITER reduce to the ϵ-greedy version of standard algorithm NASH_VI (Liu et al., 2021) and NASH_VI_EXPLOITER (Jin et al., 2021b) for learning tabular Markov games, where the numbers of states and actions are finite and small. Please see Appendix B for a detailed discussion on the connections of these algorithms.

When replacing the ϵ-greedy exploration with optimistic exploration (typically in the form of constructing upper confidence bounds), both NASH_VI and NASH_VI_EXPLOITER are guaranteed to efficiently find the Nash equilibria of MGs in the tabular settings.

Theorem 1 ((Liu et al., 2021; Jin et al., 2021b)). *For tabular Markov games, the optimistic versions of both* NASH_VI *and* NASH_VI_EXPLOITER *can find ϵ-approximate Nash equilibria in poly$(S, A, B, (1-\gamma)^{-1}, \epsilon^{-1}, \log(1/\delta))$ steps with probability at least $1-\delta$. Here S is the size of states, A, B are the size of two players' actions respectively, and γ is the discount factor.*

We defer the proof of Theorem 1 to Appendix B. We highlight that, in contrast, existing deep MARL algorithms such as NFSP (Heinrich & Silver, 2016) or PSRO (Lanctot et al., 2017) are incapable of efficient learning of Nash equilibria with a polynomial convergence rate for tabular Markov games. Our simulation results reveal that they are indeed highly inefficient in finding Nash equilibria.

4 Experiments

The experimental evaluations are conducted on randomly generated tabular Markov games and two-player video games on Slime Volley-Ball (Ha, 2020) and PettingZoo Atari (Terry et al., 2021). We tested the performance of proposed methods as well as the baseline algorithms in both (a) the basic tabular form without function approximation (only in tabular environments); and (b) full versions with deep neural networks (in both tabular environments and video games). For (a), we measure the exploitability by computing the exact best response using the ground truth transition and reward function. This is only feasible in the tabular environment. For (b), we measure the exploitability by training single-agent DQN (exploiter) against the learned policy to directly exploit it.

4.1 Baselines

For benchmarking purpose, we have the following baselines with deep neural network function approximation for scalable tests:

- **Self-Play** (SP): each agent learns to play the best response strategy against the fixed opponent strategy alternatively, i.e., iterative best response.
- **Fictitious Self-Play** (FSP) (Heinrich et al., 2015): each agent learns a best-response strategy against the episodic average of its opponent's historical strategy set, and save it to its own strategy set.
- **Neural Fictitious Self-Play** (NSFP) (Heinrich & Silver, 2016): an neural network approximation of FSP, a policy network is explicitly maintained to imitate the historical behaviours by an agent, and the learner learns the best response against it.
- **Policy Space Response Oracles** (PSRO) (Lanctot et al., 2017): we adopt a version based on double oracle (DO), each agent learns the best response against a meta-Nash strategy of its opponent's strategy set, and add the learned strategy to its own strategy set.

For tabular case without function approximation, SP, FSP and DO are implemented with Q-learning as the base learning agents for finding best responses. For tabular case with function approximation and video games, all four baseline methods use DQN as the basic agent for learning the best-response strategies. The pseudo-codes for algorithms SP, FSP, DO are provided in Appendix B.

4.2 TABULAR MARKOV GAME

Tabular forms without function approximation. We first evaluate methods (1) SP, (2) FSP, (3) PSRO, (4) NASH_DQN and (5) NASH_DQN_EXPLOITER without function approximation (i.e., w/o neural network) on the tabular Markov games. They reduce to methods (1) SP, (2) FSP, (3) DO, (4) Nash value iteration (NASH_VI) and (5) Nash value iteration with exploiter (NASH_VI_EXPLOITER), correspondingly. For SP, FSP and DO, we adopt Q-learning as a subroutine for finding the best response policies. As tabular versions of our deep MARL algorithms, NASH_VI and NASH_VI_EXPLOITER also use NASH subroutine for calculating NE in normal-form games, with $\epsilon = 0.5$ for ϵ-greedy exploration. The pseudo-codes for NASH_VI and NASH_VI_EXPLOITER are provided in Appendix B.

We randomly generated the tabular Markov games, which has discrete state space $\mathcal{S}$, discrete action spaces $\mathcal{A}, \mathcal{B}$ for two players and the horizon H[2] The state transition probability function $\{\mathcal{T}_h : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1], h \in [H]\}$ and reward function $\{R_h : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [-1, 1], h \in [H]\}$ are both i.i.d sampled uniformly over their ranges. As shown in Fig. 2, we tested on two randomly generated

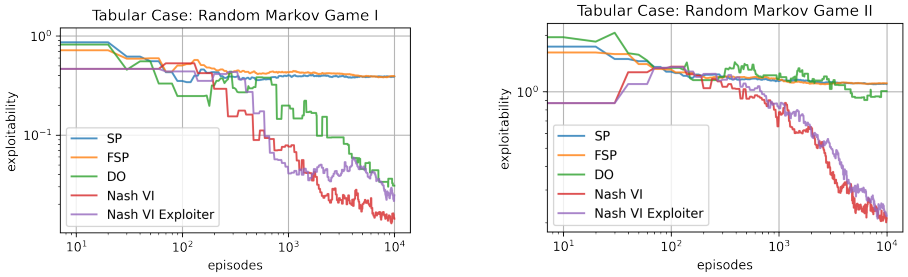

Figure 2: Tabular case experiments on two randomly generated Markov games.

Markov games of different sizes: I. $|\mathcal{S}| = |\mathcal{A}| = |\mathcal{B}| = H = 3$; II. $|\mathcal{S}| = |\mathcal{A}| = |\mathcal{B}| = H = 6$. The exploitability is calculated according to Eq. (7), which can be solved with dynamic programming in the tabular cases with known transition and reward functions. Our two proposed algorithms without function approximation show significant speedup for decreasing the exploitability compared against other baselines, especially for the larger environment (II). This aligns with our theoretical justification as in Section 3.3.

With neural networks as function approximation. In this set of experimentation, we add neural networks as function approximators. We evaluate NASH_DQN and NASH_DQN_EXPLOITER on the same tabular MG environments I and II. We setup the neural-network versions of baseline methods—FSP, NFSP and PSRO, using the same set of hyperparameters (Appendix C) and the same training configurations. During training, the model checkpoints are saved at different stages for each method and reloaded for exploitation test. Each method is trained for a total of 5×10^4 episodes to get the final model, against which the exploiter is trained for 3×10^4 episodes as the exploitation test. The exploiter is a DQN agent trained from scratch with the same hyperparameters as the base agents (for finding best responses) in FSP, NFSP, PSRO. We empirically measure the exploitability of the learner's policy as follows: we first compute smoothed version of cumulative utility achieved by the exploiter at each episode (the smoothing is conducted by averaging over a small set of neighboring episodes); we then report the highest smoothed cumulative utility of the exploiter as an approximation for the exploitability. We also test the effectiveness of using DQN exploiter to measure the exploitability, by training it against the oracle Nash strategies (i.e., the ground-truth Nash equilibria). Results for two tabular environments are displayed in Table 1. As shown in Table 1, both NASH_DQN and NASH_DQN_EXPLOITER outperform all other methods by a significant margin. The negative exploitability values of the Oracle Nash strategy indicates that the DQN-based exploiter is not able to find the exact theoretical best response. Nevertheless, it approximates the best response very well. The reported exploitability of Oracle Nash is very close to zero, which justifies the effectiveness of using DQN for exploitation tests. Complete results and details about exploitability calculation are provided in Appendix D.

4.3 TWO-PLAYER VIDEO GAME

To evaluate the scalability and robustness of the proposed method, we examine all algorithms in five two-player Atari environments in PettingZoo library (Terry et al., 2021) (*Boxing-v1*, *Double Dunk-v2*,

[2]We encode the horizon into the state space in order to use the algorithm designed for the discounted setting.

Table 1: Approximate exploitability (**lower is better**) in two tabular Markov games

Env \ Method	SP	FSP	NFSP	PSRO	**Nash DQN**	**Nash DQN Exploiter**	Oracle Nash
Tabular Env I	0.448	0.379	0.317	0.134	0.096	**0.020**	−0.027
Tabular Env II	1.239	0.694	0.379	0.569	**0.017**	0.071	−0.082

Pong-v2, *Tennis-v2*[3], *Surround-v1*) and in environment *SlimeVolley-v0* in a public available benchmark named Slime Volley-Ball (Ha, 2020), as shown in Fig. 1. The algorithms tested for this setting include: (1) SP, (2) FSP, (3) NFSP, (4) PSRO, (5) NASH_DQN and (6) NASH_DQN_EXPLOITER. To speed up the experiments, each environment is truncated to 300 steps per episode for both training and exploitation. A full length experiment is conducted on one environment in Appendix. G. For Atari games, the observation is based on RAM and normalized in range $[0, 1]$.

Similar to experiments in the tabular environment with function approximation, the exploitation test (using single-agent DQN) is conducted to evaluate the learned models. Ideally, if an agent learns the perfect Nash equilibrium strategy, then by definition, we shall expect the agent to be perfectly non-exploitable (i.e., with even the strongest exploiter only capable of achieving her cumulative utility at most zero in symmetric games). To carry out the experiment, we first trained all the methods for 5×10^4 episodes, with detailed hyperparameters provided in Appendix E. After the methods are fully trained, we take their final models or certain distributions of historical strategies (uniform for FSP and meta-Nash for PSRO), and the train separate exploiters playing against those learned strategies. We instantiate a DQN agent as exploiter using the same set of hyperparameters and network architectures to learn from scratch against the fixed trained checkpoint. The resulting learning curve in the exploitation test illustrates the degree of exploitation. An exploiter reward greater than zero indicates that the agent has been exploited since the games are symmetric. The model with lower exploiter reward means is more difficult to be exploited (is thus better).

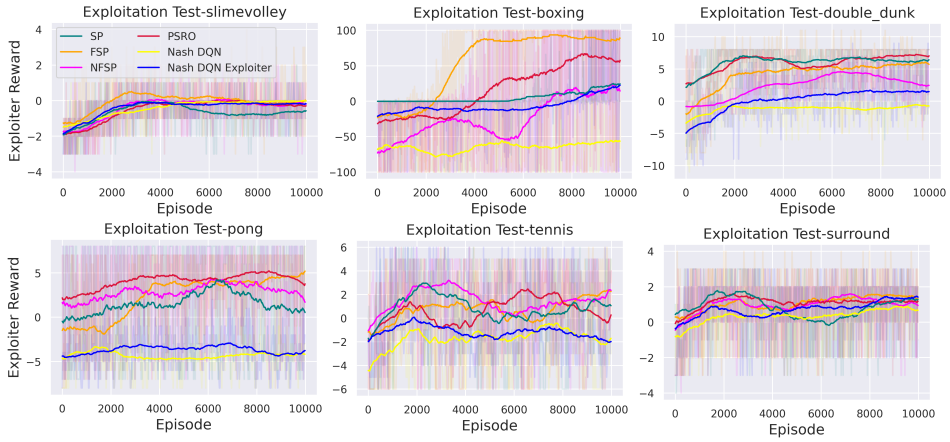

Figure 3: The exploiter learning curves for exploitation tests on six two-player zero-sum video games.

Table 2: Approximate exploitability (lower is better) for six two-player video games.

Env \ Method	SP	FSP	NFSP	PSRO	**Nash DQN**	**Nash DQN Exploiter**
SlimeVolley	−0.049	0.514	0.069	0.000	0.000	**−0.099**
Boxing	24.907	93.683	24.544	66.891	**−55.471**	22.490
Double Dunk	7.039	6.067	4.564	7.256	**−0.539**	1.702
Pong	4.207	5.196	4.396	5.217	**−3.336**	−2.920
Tennis	2.970	2.355	3.207	2.465	**−0.425**	0.069
Surround	1.782	1.574	1.594	1.603	**0.904**	1.462

Fig. 3 and Table 2 show the exploitation results of all algorithms and baselines. Each method is trained for three random seeds, and the model for each random seed is further exploited with DQN exploiter for 10^4 episodes under three random initializations. For each method and environment, Table 2 displays the best performing models with its corresponding exploitability. Complete results are provided in Appendix Sec. F. The values in the Table 2 is the maximum of smoothed exploiter reward in the exploitation test of Fig. 3. The baseline methods SP, FSP, NFSP, PSRO do not perform well in most games. This shows the challenge for finding approximate Nash equilibrium

[3]The original *Tennis-v2* environment in PettingZoo is not zero-sum, a reward wrapper is applied to make it.

strategies for these games. NASH_DQN demonstrates significant advantages over other methods across all six games. Except for *Surround* environment, NASH_DQN achieves non-positive exploiter rewards for five environments, which demonstrates the non-exploitability of the policies learned by NASH_DQN. NASH_DQN_EXPLOITER also shows unexploitable performance on *SlimeVolley* and *Pong* environments. Different environments show different levels of difficulties to find a non-exploitable model. *SlimeVolley* is relatively easy with almost all methods achieving exploitability close to zero. *Surround* is generally hard to resolve due to the inherent complexity of the game. We believe that solving Surround requires more advanced exploration technique to boost its performance.

Interestingly, we observe that in the exploitation test for *Boxing* environment, baseline methods such as SP sometimes produce a policy that keeps staying at the corner of the ground. As shown in Fig. 4, the black agent uses the learned suboptimal model by SP algorithm, which tries to avoid any touch with the white opponent (a-b). Such policy (always hide in a corner) is not bad when playing against average player or AI whose policies may not have considered this extreme cases and thus unable to even locate the black agent. However, this policy is very vulnerable to exploitation. Once the exploiter explores the way to touch the black agent, (c) our exploiter learns to heavily exploit such policy in a short time. On the contrary, our algorithm NASH_DQN and NASH_DQN_EXPLOITER will never learn such easy-to-exploit policies. The models learned with NASH_DQN and NASH_DQN_EXPLOITER are usually aggressively approaching the exploiter and directly fighting against it, which is found to be harder to exploit in this game. Moreover, our policies are robust to a turn-around strategy by the exploiter, as Fig. 4 (d).

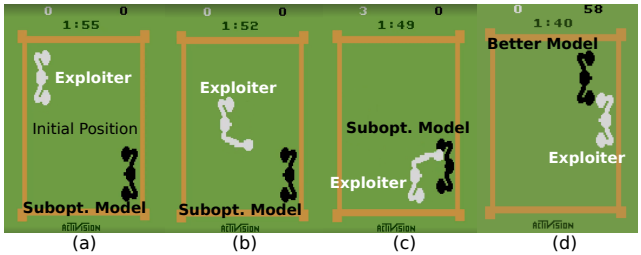

Figure 4: The key frames in *Boxing* exploitation test: (a-c) shows a sub-optimal model exploited by the exploiter. (d) shows our proposed algorithms learn hard-to-exploit policy robust against the turning-around strategy of the exploiter.

To address the possibility insufficient exploitation on our models, we exploit the models for longer time (5×10^4 episodes) for those methods within cells shaded in gray in Table 2 , and the results are shown in Fig. 5. Except for the *Double Dunk* environment, the NASH_DQN and NASH_DQN_EXPLOITER models are still hard to be exploited on four environments even for long enough exploitation. The difficulty of *Double Dunk* is that each agent needs to control a team of two players to compete through team collaboration, which might require a longer training time to further improve the learned policies.

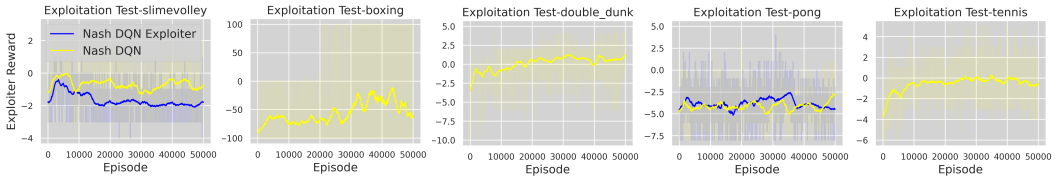

Figure 5: The exploiter learning curves for longer exploitation tests on five video games.

5 CONCLUSION AND DISCUSSION

In this paper, we propose two novel algorithms for deep reinforcement learning in the two-player zero-sum games. The idea is to incorporate Nash equilibria computation into the training process so that the agents can learn non-exploitable strategies. The first method that we propose is NASH_DQN, with the two players trained to learn the Nash value function. To guide its exploration, we also propose a variant of it named NASH_DQN_EXPLOITER, with the learning agent trained to play against an opponent that's designed to exploit learner's weakness. Experimental results demonstrate that the proposed methods converges significantly faster than existing methods in random generated tabular Markov games with or without function approximation. Our experiment further shows that our algorithms are scalable to learn the non-exploitable strategies in most of the six two-player video games, and significantly outperform all the baseline algorithms. This is to our knowledge the first work for learning non-exploitable strategies in two-player zero-sum video games like Atari. Limitations also exist for the present methods, the NE solving subroutine is repeatedly called in both inference and update procedures, which leads to large computational costs. Image-based solutions with explicit policies are to be explored in the future.

REFERENCES

Yu Bai and Chi Jin. Provable self-play algorithms for competitive reinforcement learning. In *International Conference on Machine Learning*, pp. 551–560. PMLR, 2020.

Yu Bai, Chi Jin, and Tiancheng Yu. Near-optimal reinforcement learning with self-play. *Advances in Neural Information Processing Systems*, 2020.

Yu Bai, Chi Jin, Song Mei, and Tiancheng Yu. Near-optimal learning of extensive-form games with imperfect information. *arXiv preprint arXiv:2202.01752*, 2022.

James P Bailey and Georgios Piliouras. Multiplicative weights update in zero-sum games. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 321–338, 2018.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2019.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL `http://dx.doi.org/10.1613/jair.3912`.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

Constantinos Daskalakis and Ioannis Panageas. Last-iterate convergence: Zero-sum games and constrained min-max optimization. *arXiv preprint arXiv:1807.04252*, 2018.

Le Cong Dinh, Yaodong Yang, Zheng Tian, Nicolas Perez Nieves, Oliver Slumbers, David Henry Mguni, Haitham Bou Ammar, and Jun Wang. Online double oracle. *arXiv preprint arXiv:2103.07780*, 2021.

Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pp. 3071–3076. IEEE, 2013.

Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Stochastic regret minimization in extensive-form games. In *International Conference on Machine Learning*, pp. 3018–3028. PMLR, 2020.

David Ha. Slime volleyball gym environment. `https://github.com/hardmaru/slimevolleygym`, 2020.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games, 2016.

Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *International conference on machine learning*, pp. 805–813. PMLR, 2015.

Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

Baihe Huang, Jason D Lee, Zhaoran Wang, and Zhuoran Yang. Towards general function approximation in zero-sum markov games. *arXiv preprint arXiv:2107.14702*, 2021.

Chi Jin, Qinghua Liu, Yuanhao Wang, and Tiancheng Yu. V-learning–a simple, efficient, decentralized algorithm for multiagent rl. *arXiv preprint arXiv:2110.14555*, 2021a.

Chi Jin, Qinghua Liu, and Tiancheng Yu. The power of exploiter: Provable multi-agent rl in large state spaces. *arXiv preprint arXiv:2106.03352*, 2021b.

Tadashi Kozuno, Pierre Ménard, Rémi Munos, and Michal Valko. Model-free learning for two-player zero-sum partially observable markov games with perfect recall. *arXiv preprint arXiv:2106.06279*, 2021.

Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. *Advances in neural information processing systems*, 22, 2009.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning, 2017.

Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Qinghua Liu, Tiancheng Yu, Yu Bai, and Chi Jin. A sharp analysis of model-based reinforcement learning with self-play. In *International Conference on Machine Learning*, pp. 7001–7010. PMLR, 2021.

Stephen McAleer, John Lanier, Pierre Baldi, and Roy Fox. Xdo: A double oracle algorithm for extensive-form games. *arXiv preprint arXiv:2103.06426*, 2021.

H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 536–543, 2003.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jack Serrino, Max Kleiman-Weiner, David C Parkes, and Joshua B Tenenbaum. Finding friend and foe in multi-agent games. *arXiv preprint arXiv:1906.02330*, 2019.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10): 1095–1100, 1953.

David Silver, Julian Schrittwieser, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 10 2017. doi: 10.1038/nature24270.

Justin K. Terry, Benjamin Black, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Santos, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, Caroline Horsch, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning, 2021.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Qiaomin Xie, Yudong Chen, Zhaoran Wang, and Zhuoran Yang. Learning zero-sum simultaneous-move markov games using function approximation and correlated equilibrium. In *Conference on Learning Theory*, pp. 3674–3682. PMLR, 2020.

Daochen Zha, Jingru Xie, Wenye Ma, Sheng Zhang, Xiangru Lian, Xia Hu, and Ji Liu. Douzero: Mastering doudizhu with self-play deep reinforcement learning. *arXiv preprint arXiv:2106.06135*, 2021.