DouZero: Mastering DouDizhu with Self-Play Deep Reinforcement Learning

Daochen Zha [1] Jingru Xie [2] Wenye Ma [2] Sheng Zhang [3] Xiangru Lian [2] Xia Hu [1] Ji Liu [2]

Abstract

Games are abstractions of the real world, where artificial agents learn to compete and cooperate with other agents. While significant achievements have been made in various perfect- and imperfect-information games, DouDizhu (a.k.a. Fighting the Landlord), a three-player card game, is still unsolved. DouDizhu is a very challenging domain with competition, collaboration, imperfect information, large state space, and particularly a massive set of possible actions where the legal actions vary significantly from turn to turn. Unfortunately, modern reinforcement learning algorithms mainly focus on simple and small action spaces, and not surprisingly, are shown not to make satisfactory progress in DouDizhu. In this work, we propose a conceptually simple yet effective DouDizhu AI system, namely DouZero, which enhances traditional Monte-Carlo methods with deep neural networks, action encoding, and parallel actors. Starting from scratch in a single server with four GPUs, DouZero outperformed all the existing DouDizhu AI programs in days of training and was ranked the first in the Botzone leaderboard among 344 AI agents. Through building DouZero, we show that classic Monte-Carlo methods can be made to deliver strong results in a hard domain with a complex action space. The code and an online demo are released[1] with the hope that this insight could motivate future work.

1. Introduction

Games often serve as benchmarks of AI since they are abstractions of many real-world problems. Significant achievements have been made in perfect-information games. For example, AlphaGo (Silver et al., 2016), AlphaZero (Silver et al., 2018) and MuZero (Schrittwieser et al., 2020) have established state-of-the-art performance on Go game. Recent research has evolved to more challenging imperfect-information games, where the agents compete or cooperate with others in a partially observable environment. Encouraging progress has been made from two-player games, such as simple Leduc Hold'em and limit/no-limit Texas Hold'em (Zinkevich et al., 2008; Heinrich & Silver, 2016; Moravčík et al., 2017; Brown & Sandholm, 2018), to multi-player games, such as multi-player Texas hold'em (Brown & Sandholm, 2019b), Starcraft (Vinyals et al., 2019), DOTA (Berner et al., 2019), Hanabi (Lerer et al., 2020), Mahjong (Li et al., 2020a), Honor of Kings (Ye et al., 2020b;a), and No-Press Diplomacy (Gray et al., 2020).

This work aims at building AI programs for DouDizhu[2] (a.k.a. Fighting the Landlord), the most popular card game in China with hundreds of millions of daily active players. DouDizhu has two interesting properties that pose great challenges for AI systems. First, the players in DouDizhu need to both compete and cooperate with others in a partially observable environment with limited communication. Specifically, two Peasants players will play as a team to fight against the Landlord player. Popular algorithms for poker games, such as Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2008)) and its variants, are often not sound in this complex three-player setting. Second, DouDizhu has a large number of information sets with a very large average size and has a very complex and large action space of up to 10^4 possible actions due to combinations of cards (Zha et al., 2019a). Unlike Texas Hold'em, the actions in DouDizhu can not be easily abstracted, which makes search computationally expensive and commonly used reinforcement learning algorithms less effective. Deep Q-Learning (DQN) (Mnih et al., 2015) is problematic in very large action space due to overestimating issue (Zahavy et al., 2018); policy gradient methods, such as A3C (Mnih et al., 2016), cannot leverage the action features in DouDizhu, and thus cannot generalize over unseen actions as naturally as DQN (Dulac-Arnold et al., 2015). Not surprisingly, previous work shows that DQN and A3C can not make satisfactory progress in DouDizhu. In (You et al., 2019), DQN and A3C are shown to have less than 20% winning percentage against

[1]Department of Computer Science and Engineering, Texas A&M University [2]AI Platform, Kwai Inc. [3]Georgia Institute of Technology. Correspondence to: Daochen Zha <daochen.zha@tamu.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

[1]https://github.com/kwai/DouZero

[2]https://en.wikipedia.org/wiki/Dou_dizhu

simple rule-based agents even with twenty days of training; the DQN in (Zha et al., 2019a) is only slightly better than random agents that sample legal moves uniformly.

Some previous efforts have been made to build DouDizhu AI by combining human heuristics with learning and search. Combination Q-Network (CQN) (You et al., 2019) proposes to reduce the action space by decoupling the actions into decomposition selection and final move selection. However, decomposition relies on human heuristics and is extremely slow. In practice, CQN can not even beat simple heuristic rules after twenty days of training. DeltaDou (Jiang et al., 2019) is the first AI program that reaches human-level performance compared with top human players. It enables an AlphaZero-like algorithm by using Bayesian methods to infer hidden information and sampling the other players' actions based on their own policy networks. To abstract the action space, DeltaDou pre-trains a kicker network based on heuristic rules. However, the kicker plays an important role in DouDizhu and can not be easily abstracted. A bad selection of the kicker may directly result in losing a game since it may break some other card categories, e.g., a Chain of Solo. Moreover, the Bayesian inference and the search are computationally expensive. It takes more than two months to train DeltaDou even when initializing the networks with supervised regression to heuristics (Jiang et al., 2019). Therefore, the existing DouDizhu AI programs are computationally expensive and could be sub-optimal since they highly rely on abstractions with human knowledge.

In this work, we present DouZero, a conceptually simple yet effective AI system for DouDizhu without the abstraction of the state/action space or any human knowledge. DouZero enhances traditional Monte-Carlo methods (Sutton & Barto, 2018) with deep neural networks, action encoding, and parallel actors. DouZero has two desirable properties. First, unlike DQN, it is not susceptible to overestimation bias. Second, by encoding the actions into card matrices, it can naturally generalize over the actions that are not frequently seen throughout the training process. Both of these two properties are crucial in dealing with the huge and complex action space of DouDizhu. Unlike many tree search algorithms, DouZero is based on sampling, which allows us to use complex neural architectures and generate much more data per second, given the same computational resources. Unlike many prior poker AI studies that rely on domain-specific abstractions, DouZero does not require any domain knowledge or knowledge of the underlying dynamics. Trained from scratch in a single server with only 48 cores and four 1080Ti GPUs, DouZero outperforms CQN and the heuristic rules in half a day, beats our internal supervised agents in two days, and surpasses DeltaDou in ten days. Extensive evaluations suggest that DouZero is the strongest DouDizhu AI system up to date.

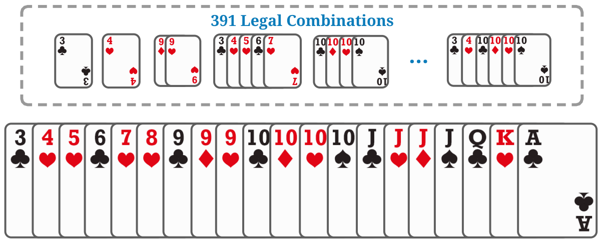


Figure 1. A hand and its corresponding legal moves.

Through building DouZero system, we demonstrate that classical Monte-Carlo methods can be made to deliver strong results in large-scale and complex card games that need to reason about both competing and cooperation over huge state and action spaces. We note that some work also discovers that Monte-Carlo methods can achieve competitive performance (Mania et al., 2018; Zha et al., 2021a) and help in sparse rewards settings (Guo et al., 2018; Zha et al., 2021b). Unlike these studies that focus on simple and small environments, we demonstrate the strong performance of Monte-Carlo methods on a large-scale card game. With the hope that this insight could facilitate future research on tackling multi-agent learning, sparse reward, complex action spaces, and imperfect information, we have released our environment and the training code. Unlike many Poker AI systems that require thousands of CPUs in training, e.g., DeepStack (Moravčík et al., 2017) and Libratus (Brown & Sandholm, 2018), DouZero enables a reasonable experimental pipeline, which only requires days of training on a single GPU server that is affordable for most research labs. We hope that it could motivate future research in this domain and serve as a strong baseline.

2. Background of DouDizhu

DouDizhu is a popular three-player card game that is easy to learn but difficult to master. It has attracted hundreds of millions of players in China, with many tournaments held every year. It is a shedding-type game where the player's objective is to empty one's hand of all cards before other players. Two of the Peasants players play as a team to fight against the other Landlord player. The Peasants win if either of the Peasants players is the first to have no cards left. Each game has a bidding phase, where the players bid for the Landlord based on the strengths of the hand cards, and a card-playing phase, where the players play cards in turn. We provide a detailed introduction in Appendix A.

DouDizhu is still an unsolved benchmark for multi-agent reinforcement learning (Zha et al., 2019a; Terry et al., 2020). Two interesting properties make DouDizhu particularly challenging to solve. First, the Peasants need to cooperate in fighting against the Landlord. For example, Figure 10 shows a typical situation where the bottom Peasant can choose to play a small Solo to help the Peasant on the right-hand side to win. Second, DouDizhu has a complex and large action

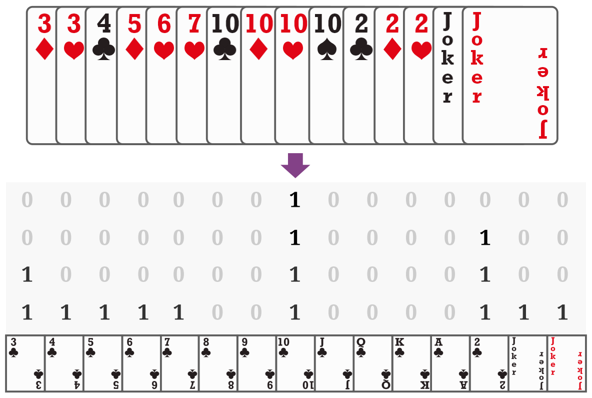


Figure 2. Cards for both states and actions are encoded into a 4×15 one-hot matrix, where columns correspond to the 13 ranks and the jokers, and each row corresponds to the number of cards of a specific rank or joker. More examples are provided in Appendix B.

space due to the combination of cards. There are 27,472 possible combinations, where different subsets of these combinations will be legal for different hands. Figure 1 shows an example of the hand, which has 391 legal combinations, including Solo, Pair, Trio, Bomb, Plane, Quad, etc. The action space can not be easily abstracted since improperly playing a card may break other categories and directly result in losing a game. Thus, building DouDizhu AI is challenging since the players in DouDizhu need to reason about both competing and cooperation over a huge action space.

3. Deep Monte-Carlo

In this section, we revisit Monte-Carlo (MC) methods and introduce Deep Monte-Carlo (DMC), which generalizes MC with deep neural networks for function approximation. Then we discuss and compare DMC with policy gradient methods (e.g., A3C) and DQN, which are shown to fail in DouDizhu (You et al., 2019; Zha et al., 2019a).

3.1. Monte-Carlo Methods with Deep Neural Networks

Monte-Carlo (MC) methods are traditional reinforcement learning algorithms based on averaging sample returns (Sutton & Barto, 2018). MC methods are designed for episodic tasks, where experiences can be divided into episodes and all the episodes eventually terminate. To optimize a policy π, every-visit MC can be used to estimate Q-table $Q(s, a)$ by iteratively executing the following procedure:

1. Generate an episode using π.
2. For each s, a appeared in the episode, calculate and update $Q(s, a)$ with the return averaged over all the samples concerning s, a.
3. For each s in the episode, $\pi(s) \leftarrow \arg\max_a Q(s, a)$.

The average return in Step 2 is usually obtained by the discounted cumulative reward. Different from Q-learning that relies on bootstrapping, MC methods directly approximate the target Q-value. In step 1, we can use epsilon-greedy to balance exploration and exploitation. The above procedure can be naturally combined with deep neural networks, which leads to Deep Monte-Carlo (DMC). Specifically, we can replace the Q-table with a neural network and use mean-square-error (MSE) to update the Q-network in Step 2.

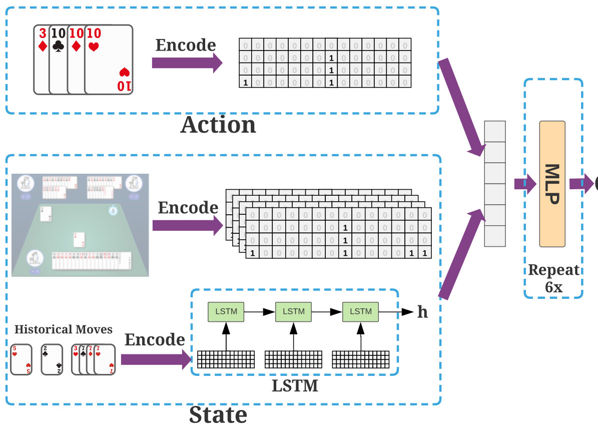


Figure 3. The Q-network of DouZero consists of an LSTM to encode historical moves and six layers of MLP with hidden dimension of 512. The network predicts a value for a given state-action pair based on the concatenated representation of action and state. More details are provided in Appendix C.1.

While MC methods are criticized not to be able to deal with incomplete episodes and believed to be inefficient due to the high variance (Sutton & Barto, 2018), DMC is very suitable for DouDizhu. First, DouDizhu is an episodic task so that we do not need to handle incomplete episodes. Second, DMC can be easily parallelized to efficiently generate many samples per second to alleviate the high variance issue.

3.2. Comparison with Policy Gradient Methods

Policy gradients methods, such as REINFORCE (Williams, 1992), A3C (Mnih et al., 2016), PPO (Schulman et al., 2017), and IMPALA (Espeholt et al., 2018), are very popular for reinforcement learning. They target modeling and optimizing the policy directly with gradient descent. In policy gradient methods, we often use a classifier-like function approximator, where the output scales linearly with the number of actions. While policy gradients methods work well in large action space, they cannot use the action features to reason about previously unseen actions (Dulac-Arnold et al., 2015). In practice, the actions in DouDizhu can be naturally encoded into card matrices, which are crucial for reasoning. For example, if the agent is rewarded by the action 3KKK because it chooses a nice kicker, it could also generalize this knowledge to unseen actions in the future, such as 3JJJ. This property is crucial in dealing with very large action spaces and accelerating the learning since many of the actions are not frequently seen in the simulated data.

DMC can naturally leverage the action features to generalize over unseen actions by taking as input the action features. While it might have high execution complexity if the action size is large, in most states of DouDizhu, only a subset of the actions is legal, so that we do not need to iterate over all the actions. Thus, DMC is overall an efficient algorithm for DouDizhu. While it is possible to introduce action features into an actor-critic framework (e.g., by using a Q-network as the critic), the classifier-like actor will still suffer from the large action space. Our preliminary experiments confirm that this strategy is not very effective (see Figure 7).

3.3. Comparison with Deep Q-Learning

The most popular value-based algorithm is Deep Q-Learning (DQN) (Mnih et al., 2015), which is a bootstrapping method that updates the Q-value based on the Q-values in the next step. While both DMC and DQN approximate the Q-values, DMC has several advantages in DouDizhu.

First, the overestimation bias caused by approximating the maximum action value in DQN is difficult to control when using function approximation (Thrun & Schwartz, 1993; Hasselt, 2010) and becomes more pronounced with very large action space (Zahavy et al., 2018). While some techniques, such as double Q-learning (van Hasselt et al., 2016) and experience replay (Lin, 1992), might alleviate this issue, we find in practice that DQN is very unstable and often diverges in DouDizhu. Whereas, Monte-Carlo estimation is not susceptible to bias since it directly approximates the true values without bootstrapping (Sutton & Barto, 2018).

Second, DouDizhu is a task with long horizons and sparse reward, i.e., the agent will need to go though a long chain of states without feedback, and the only time a nonzero reward is incurred is at the end of a game. This may slow down the convergence of Q-learning because estimating the Q-value in the current state needs to wait until the value in the next state gets close to its true value (Szepesvári, 2009; Beleznay et al., 1999). Unlike DQN, the convergence of Monte-Carlo estimation is not impacted by the episode length since it directly approximates the true target values.

Third, it is inconvenient to efficiently implement DQN in DouDizhu due to the large and variable action space. Specifically, the max operation of DQN in every update step will cause high computation cost since it requires iterating across all the legal actions on a very costly deep Q-network. Moreover, the legal moves differ in different states, which makes it inconvenient to do batch learning. As a result, we find DQN is too slow in terms of wall-clock time. While Monte-Carlo methods might suffer from high variance (Sutton & Barto, 2018), which means it might require more samples to converge, it can be easily parallelized to generate thousands of samples per second to alleviate the high variance issue and accelerate training. We find that the high variance of DMC is greatly outweighed by the scalability it provides, and DMC is very efficient in wall-clock time.

4. DouZero System

In this section, we introduce DouZero system by first describing the state/action representations and neural architecture and then elaborating on how we parallelize DMC with multiple processes to stabilize and accelerate training.

4.1. Card Representation and Neural Architecture

We encode each card combination with a one-hot 4×15 matrix (Figure 2). Since suits are irrelevant in DouDizhu, we use each row to represent the number of cards of a specific rank or joker. Figure 3 shows the architecture of the Q-network. For the state, we extract several card matrices to represent the hand cards, the union of the other players' hand cards and the most recent moves, and some one-hot vectors to represent the number of cards of the other players and the number of bombs played so far. Similarly, we use one card matrix to encode the action. For the neural architecture, LSTM is used to encode historical moves, and the output is concatenated with the other state/action features. Finally, we use six layers of MLP with a hidden size of 512 to produce Q-values. We provide more details in Appendix C.1.

4.2. Parallel Actors

We denote Landlord as L, the player that moves before the Landlord as U, and the player that moves after the Landlord as D. We parallelize DMC with multiple actor processes and one learner process, summarized in Algorithm 1 and Algorithm 2, respectively. The learner maintains three global Q-networks for the three positions and updates the networks with MSE loss to approximate the target values based on the data provided by the actor processes. Each actor maintains three local Q-networks, which are synchronized with the global networks periodically. The actor will repeatedly sample trajectories from the game engine and calculate cumulative reward for each state-action pair. The communication of learner and actors are implemented with three shared buffers. Each buffer is divided into several entries, where each entry consists of several data instances.

5. Experiments

The experiments are designed to answer the following research questions. **RQ1:** How does DouZero compare with existing DouDizhu programs, such as rule-based strategies, supervised learning, RL-based methods, and MCTS-based solutions (Section 5.2)? **RQ2:** How will DouZero perform if we consider bidding phase (Section 5.3)? **RQ3:** How efficient is the training of DouZero (Section 5.4)? **RQ4:** How does DouZero compare with bootstrapping and actor critic

Algorithm 1 Actor Process of DouZero

```
1:  Input: Shared buffers $\mathcal{B}_L$, $\mathcal{B}_U$ and $\mathcal{B}_D$ with $B$ entries
    and size $S$ for each entry, exploration hyperparameter $\epsilon$,
    discount factor $\gamma$
2:  Initialize local Q-networks $Q_{\mathrm{L}}$, $Q_{\mathrm{U}}$ and $Q_{\mathrm{D}}$, and local
    buffers $\mathcal{D}_L$, $\mathcal{D}_U$ and $\mathcal{D}_D$
3:  for iteration = 1, 2, ... do
4:      Synchronize $Q_{\mathrm{L}}$, $Q_{\mathrm{U}}$ and $Q_{\mathrm{D}}$ with the learner process
5:      for t = 1, 2, ... T do                  ▷ Generate an episode
6:          $Q \leftarrow$ one of $Q_{\mathrm{L}}$, $Q_{\mathrm{U}}$, $Q_{\mathrm{D}}$ based on position
7:          $a_t \leftarrow \begin{cases} \arg\max_a Q(s_t, a) & \text{with prob } (1-\epsilon) \\ \text{random action} & \text{with prob } \epsilon \end{cases}$
8:          Perform $a_t$, observe $s_{t+1}$ and reward $r_t$
9:          Store $\{s_t, a_t, r_t\}$ to $\mathcal{D}_L$, $\mathcal{D}_U$, or $\mathcal{D}_D$ accordingly
10:     end for
11:     for t = T-1, T-2, ... 1 do ▷ Obtain cumulative reward
12:         $r_t \leftarrow r_t + \gamma r_{t+1}$ and update $r_t$ in $\mathcal{D}_L$, $\mathcal{D}_U$, or $\mathcal{D}_D$
13:     end for
14:     for $p \in \{L, U, D\}$ do  ▷ Optimized by multi-thread
15:         if $\mathcal{D}_p$.length $\geq L$ then
16:             Request and wait for an empty entry in $\mathcal{B}_p$
17:             Move $\{s_t, a_t, r_t\}$ of size $L$ from $\mathcal{D}_p$ to $\mathcal{B}_p$
18:         end if
19:     end for
20: end for
```

Algorithm 2 Learner Process of DouZero

```
1:  Input: Shared buffers $\mathcal{B}_L$, $\mathcal{B}_U$ and $\mathcal{B}_D$ with $B$ entries
    and size $S$ for each entry, batch size $M$, learning rate $\psi$
2:  Initialize global Q-networks $Q^g_{\mathrm{L}}$, $Q^g_{\mathrm{U}}$ and $Q^g_{\mathrm{D}}$
3:  for iteration = 1, 2, ... until convergence do
4:      for $p \in \{L, U, D\}$ do  ▷ Optimized by multi-thread
5:          if the number of full entries in $\mathcal{B}_p \geq M$ then
6:              Sample a batch of $\{s_t, a_t, r_t\}$ with $M \times S$ in-
                stances from $\mathcal{B}_p$ and free the entries
7:              Update $Q^g_{\mathrm{p}}$ with MSE loss and learning rate $\psi$
8:          end if
9:      end for
10: end for
```

methods (Section 5.5)? **RQ5:** Does the learned card playing strategies of DouZero align with human knowledge (Section 5.6)? **RQ6:** Is DouZero computationally efficient in inference compared with existing programs (Section 5.7)? **RQ7:** Can the two Peasants of DouZero learn to cooperate with each other (Section 5.8)?

5.1. Experimental Setup

A commonly used measure of strategy strength in poker games is exploitability (Johanson et al., 2011). However, in DouDizhu, calculating exploitability itself is intractable since DouDizhu has huge state/action spaces, and there are three players. To evaluate the performance, following (Jiang et al., 2019), we launch tournaments that include the two opponent sides of Landlord and Peasants. We reduce the variance by playing each deck twice. Specifically, for two competing algorithms A and B, they will first play as Landlord and Peasants positions, respectively, for a given deck. Then they switch sides, i.e., A takes Peasants position, and B takes Landlord position, and play the same deck again. To simulate the real environment, in Section 5.3, we further train a bidding network with supervised learning, and the agents will bid the Landlord in each game based on the strengths of the hand cards (more details in Appendix C.2). We consider the following competing algorithms.

- **DeltaDou:** A strong AI program which uses Bayesian methods to infer hidden information and searches the moves with MCTS (Jiang et al., 2019). We use the code and the pre-trained model provided by the authors. The model is trained for two months and is shown to have on par performance with top human players.

- **CQN:** Combinational Q-Learning (You et al., 2019) is a program based on card decomposition and Deep Q-Learning. We use the open-sourced code and the pre-trained model provided by the authors[3].

- **SL:** A supervised learning baseline. We internally collect $226,230$ human expert matches from the players of the highest level in league in our DouDizhu game mobile app. Then we use the same state representation and neural architecture as DouZero to train supervised agents with $49,990,075$ samples generated from these data. See Appendix C.2 for more details.

- **Rule-Based Programs:** We collect some open-sourced heuristic-based programs, including **RHCP**[4], an improved version called **RHCP-v2**[5], and the rule model in **RLCard** package[6] (Zha et al., 2019a). In addition, we consider a **Random** program that samples legal moves uniformly.

Metrics. Following (Jiang et al., 2019), given an algorithm A and an opponent B, we use two metrics to compare the performance of A and B:

- **WP** (Winning Percentage): The number of the games won by A divided by the total number of games.

[3] https://github.com/qq456cvb/doudizhu-C

[4] https://blog.csdn.net/sm9sun/article/details/70787814

[5] https://github.com/deecamp2019-group20/RuleBasedModelV2

[6] https://github.com/datamllab/rlcard

Table 1. Performance of DouZero against existing DouDizhu programs by playing 10,000 randomly sampled decks. Algorithm A outperforms B if WP is larger than 0.5 or ADP is larger than 0 (highlighted in boldface). The algorithms are ranked according to the number of the other algorithms that they beat. The full results of each position are provided in Appendix D.1.

Rank	A \ B	DouZero		DeltaDou		SL		RHCP-v2		RHCP		RLCard		CQN		Random	
		WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP
1	DouZero	-	-	**0.586**	**0.258**	**0.659**	**0.700**	**0.757**	**1.662**	**0.764**	**1.671**	**0.889**	**2.288**	**0.810**	**1.685**	**0.989**	**3.036**
2	DeltaDou	0.414	-0.258	-	-	**0.617**	**0.653**	**0.745**	**1.500**	**0.747**	**1.514**	**0.876**	**2.459**	**0.784**	**1.534**	**0.992**	**3.099**
3	SL	0.341	-0.700	0.396	-0.653	-	-	**0.611**	**0.853**	**0.632**	**0.886**	**0.813**	**1.821**	**0.694**	**1.037**	**0.976**	**2.721**
4	RHCP-v2	0.243	-1.662	0.257	-1.500	0.389	-0.853	-	-	**0.515**	**0.052**	**0.692**	**1.121**	**0.621**	**0.714**	**0.967**	**2.631**
5	RHCP	0.236	-1.671	0.253	-1.514	0.369	-0.886	0.485	-0.052	-	-	**0.682**	**1.259**	**0.603**	**0.248**	**0.941**	**2.720**
6	RLCard	0.111	-2.288	0.124	-2.459	0.187	-1.821	0.309	-1.121	0.318	-1.259	-	-	**0.522**	**0.168**	**0.943**	**2.471**
7	CQN	0.190	-1.685	0.216	-1.534	0.306	-1.037	0.379	-0.714	0.397	-0.248	0.478	-0.168	-	-	**0.889**	**1.912**
8	Random	0.011	-3.036	0.008	-3.099	0.024	-2.721	0.033	-2.631	0.059	-2.720	0.057	-2.471	0.111	-1.912	-	-

- **ADP** (Average Difference in Points): The average difference of points scored per game between A and B. The base point is 1. Each bomb will double the score.

We find in practice that these two metrics encourage different styles of strategies. For example, if using ADP as reward, the agent tends to be very cautious about playing bombs since playing a bomb is risky and may lead to larger ADP loss. In contrast, with WP as objective, the agent tends to aggressively play bombs even if it will lose because a bomb will not affect WP. We observe that the agent trained with ADP performs slightly better than the agent trained with WP in terms of ADP and vice versa. In what follows, we train and report the results of two DouZero agents with ADP and WP as objectives, respectively[7]. More discussions of the two objectives are provided in Appendix D.2.

We first launch a preliminary tournament by letting each pair of the algorithms play 10,000 decks. We then compute the Elo rating score for the top 3 algorithms for a more reliable comparison, i.e., DouZero, DeltaDou, and SL, by playing 100,000 decks. An algorithm wins a deck if it achieves higher WP or ADP summed over the two games played on this deck. We repeat this process five times with different randomly sampled decks and report the mean and standard deviation of the Elo scores. For the evaluation with the bidding phase, each deck is played six times with different perturbations of DouZero, DeltaDou, and SL in different positions. We report the result with 100,000 decks.

Implementation Details. We run all the experiments on a single server with 48 processors of Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz and four 1080 Ti GPUs. We use 45 actors, which are allocated across three GPUs. We run a learner in the remaining GPU to train the Q-networks. Our implementation is based on TorchBeast framework (Küttler et al., 2019). The detailed training curves are provided in

[7]For WP, we give a +1 or -1 reward to the final timestep based on whether the agent wins or loses a game. For ADP, we directly use ADP as the rewards. DeltaDou and CQN were trained with ADP and WP as objectives, respectively.

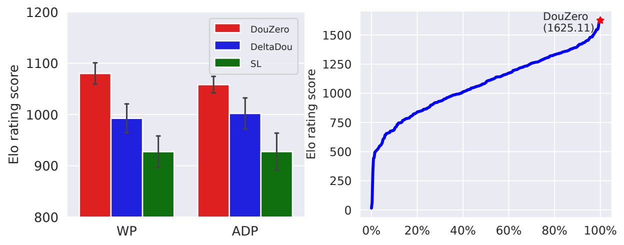


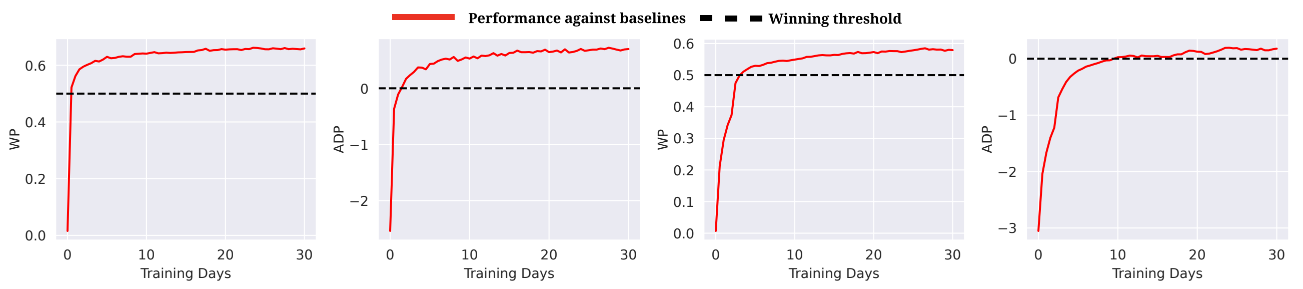
Figure 4. **Left:** Elo rating scores of DouZero, DeltaDou, and SL by playing 100,000 randomly sampled decks. We report the mean and standard deviation across 5 different random seeds. **Right:** Elo rating scores on Botzone, an online platform for DouDizhu competition. DouZero ranked the first among the 344 bots, achieving an Elo rating score of 1625.11 as of October 30, 2020.

Appendix D.5. Each shared buffer has $B = 50$ entries with size $S = 100$, batch size $M = 32$, and $\epsilon = 0.01$. We set discount factor $\gamma = 1$ since DouDizhu only has a non-zero reward in the last timestep and early moves are very important. We use ReLU as the activation function for each layer of MLP. We adopt RMSprop optimizer with a learning rate $\psi = 0.0001$, smoothing constant 0.99 and $\epsilon = 10^{-5}$. We train DouZero for 30 days.

5.2. Performance against Existing Programs

To answer **RQ1**, we compare DouZero with the baselines offline and report its result on Botzone (Zhou et al., 2018), an online platform for DouDizhu competition (more details are provided in Appendix E).

Table 1 summarizes the WP and ADP of head-to-head completions among DouZero and all the baselines. We make three observationss. First, DouZero dominates all the rule-based strategies and supervised learning, which demonstrates the effectiveness of adopting reinforcement learning in DouDizhu. Second, DouZero achieves significantly better performance than CQN. Recall that CQN similarly trains the Q-networks with action decomposition and DQN. The superiority of DouZero suggests that DMC is indeed an effective way to train the Q-networks in DouDizhu. Third, DouZero outperforms DeltaDou, the strongest DouDizhu



(a) WP against SL (b) ADP against SL (c) WP against DeltaDou (d) ADP against DeltaDou

Figure 5. WP and ADP of DouZero against SL and DeltaDou w.r.t. the number of training days. DouZero outperforms SL with 2 days of training, i.e., the overall WP is larger than the threshold of 0.5 and the overall ADP is larger than the threshold of 0, and surpasses DeltaDou within 10 days, using a single server with four 1080 Ti GPUs and 48 processors. We provide the full curves for each position and the curves w.r.t. timesteps in Appendix D.3.

Table 2. Comparison of DouZero, DeltaDou and SL with the bidding phase by playing 100,000 randomly sampled decks.

	DouZero	DeltaDou	SL
WP	**0.580**	0.461	0.381
ADP	**0.323**	-0.004	-0.320

AI in the literature. We note that DouDizhu has very high variance, i.e., to win a game relies on the strength of the initial hand card, which is highly dependent on luck. Thus, a WP of 0.586 and an ADP of 0.258 suggest a significant improvement over DeltaDou. Moreover, DeltaDou requires searching at both training and testing time. Whereas, DouZero does not do the searching, which verifies that the Q-networks learned by DouZero are very strong.

The left-hand side of Figure 4 shows the Elo rating scores of DouZero, DeltaDou, and SL by playing $100,000$ decks. We observe that DouZero outperforms DeltaDou and SL in terms of both WP and ADP significantly. This again demonstrates the strong performance of DouZero.

The right-hand side of Figure 4 illustrates the performance of DouZero on Botzone leaderboard. We note that Botzone adopts a different scoring mechanism. In addition to WP, it gives additional bonuses to some specific card categories, such as Chain of Pair and Rocket (detailed in Appendix E). While it is very likely that DouZero can achieve better performance if using the scoring mechanism of Botzone as the objective, we directly upload the pre-trained model of DouZero that is trained with WP as objective. We observe that this model is strong enough to beat the other bots.

5.3. Comparison with Bidding Phase

To investigate **RQ2**, we train a bidding network with supervised learning using human expert data. We place the top-3 algorithms, i.e., DouZero, DeltaDou, and SL, into the three seats of a DouDizhu game. In each game, we randomly choose the first bidder and simulate the bidding phase with the pre-trained bidding network. The same bidding network is used for all three algorithms for a fair comparison. The results are summarized in Table 2. Although DouZero is trained on randomly generated decks without the bidding network, we observe that DouZero dominates the other two algorithms in both WP and ADP. This demonstrates the applicability of DouZero in real-world competitions where the bidding phase needs to be considered.

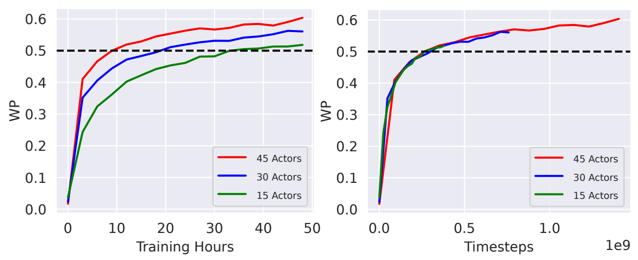


Figure 6. **Left:** The WP against SL w.r.t. training time using different number of actors **Right:** The WP against SL w.r.t. timesteps using different number of actors.

5.4. Analysis of Learning Progress

To study **RQ3**, we visualize the learning progress of DouZero in Figure 5. We use SL and DeltaDou as opponents to draw the curves of WP and ADP w.r.t. the number of training days. We make two observations as follows. First, DouZero outperforms SL in one day and two days of training in terms of WP and ADP, respectively. We note that DouZero and SL use the exactly same neural architecture for training. Thus, we attribute the superiority of DouZero to self-play reinforcement learning. While SL also performs well, it relies on a large amount of data, which is not flexible and could limit its performance. Second, DouZero outperforms DeltaDou in three days and ten days of training in terms of WP and ADP, respectively. We note that DeltaDou is initialized with supervised learning on heuristics and is trained for more than two months. Whereas, DouZero starts from scratch and only needs days of training to beat DeltaDou. This suggests that model-free reinforcement learning without search is indeed effective in DouDizhu.

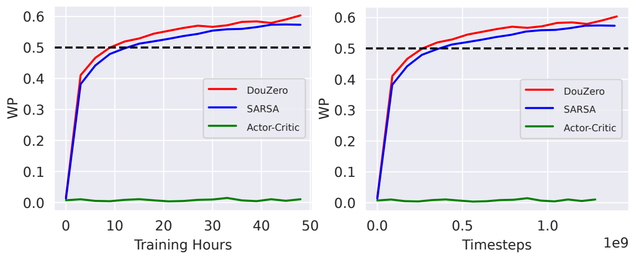


Figure 7. **Left:** The WP against SL w.r.t. training time for SARSA and Actor-Critic **Right:** The WP against SL w.r.t. timesteps for SARSA and Actor-Critic.

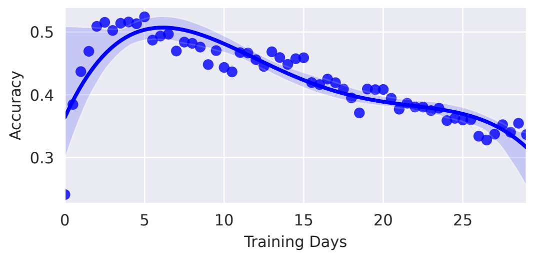


Figure 8. Average accuracy across the three positions on the human data w.r.t. the number of training days for DouZero. We fit the data points with a polynomial with four terms for better visualizing the trend. The accuracy for SL is 84.2%. DouZero aligns with human expertise in the first five days of training but discovers novel strategies beyond human knowledge in the later training stages. The curves for all the three positions are provided in Appendix D.4.

We further analyze the learning speed when using different numbers of actors. Figure 6 reports the performance against SL when using 15, 30, and 45 actors. We observe that using more actors can accelerate the training in wall-clock time. We also find that all three settings show similar sample efficiency. In the future, we will explore the possibility of using more actors across multiple servers to further improve the training efficiency.

5.5. Comparison with SARSA and Actor-Critic

To answer **RQ4**, we implement two variants based on DouZero. First, we replace the DMC objective with the Temporal-Difference (TD) objective. This leads to a deep version of SARSA. Second, we implement an Actor-Critic variant with action features. Specifically, we use Q-network as a critic with action features and train policy as an actor with action masks to remove illegal actions.

Figure 7 shows the results of SARSA and Actor-Critic with a single run. First, we do not observe a clear benefit of using TD learning. We observe that DMC learns slightly faster than SARSA in wall-clock time and sample efficiency. The possible reason is that TD learning will not help much in the sparse reward setting. We believe more studies are needed to understand when TD learning will help. Second, we observe the Actor-Critic fails. This suggests that simply adding action features to the critic may not be enough to resolve the complex action space issue. In the future, we will investigate whether we can effectively incorporate action features into the actor-critic framework.

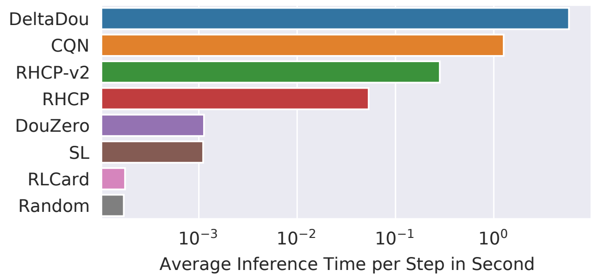


Figure 9. Comparison of inference time.

5.6. Analysis of DouZero on Expert Data

For **RQ5**, we calculate the accuracy of DouZero on the human data throughout the training process. We report the model trained with ADP as objective since the game app from which the human data is collected also adopts ADP. Figure 8 shows the results. We make two interesting observations as follows. First, at the early stages, i.e., the first five days of training, the accuracy keeps improving. This suggests that the agents may have learned some strategies that align with human expertise with purely self-play. Second, after five days of training, the accuracy decreases dramatically. We note that the ADP against SL is still improving after five days. This suggests that the agents may have discovered some novel and stronger strategies that humans can not easily discover, which again verifies the effectiveness of self-play reinforcement learning.

5.7. Comparison of Inference Time

To answer **RQ6**, we report the average inference time per step in Figure 9. For a fair comparison, we evaluate all the algorithms on the CPU. We observe that DouZero is orders of magnitude faster than DeltaDou, CQN, RHCP, and RHCP-v2. This is expected since DeltaDou needs to perform a large number of Monte Carlo simulations, and CQN, RHCP, and RHCP-v2 require expensive card decomposition. Whereas, DouZero only performs one forward pass of neural networks in each step. The efficient inference of DouZero enables us to generate a large number of samples per second for reinforcement learning. It also makes it affordable to deploy the models in real-world applications.

5.8. Case Study

To investigate **RQ7**, we conduct case studies to understand the decisions made by DouZero. We dump the logs of the competitions from Botzone and visualize the top actions with their predicted Q-values. We provide most of the case studies, including both good and bad cases, in Appendix F.

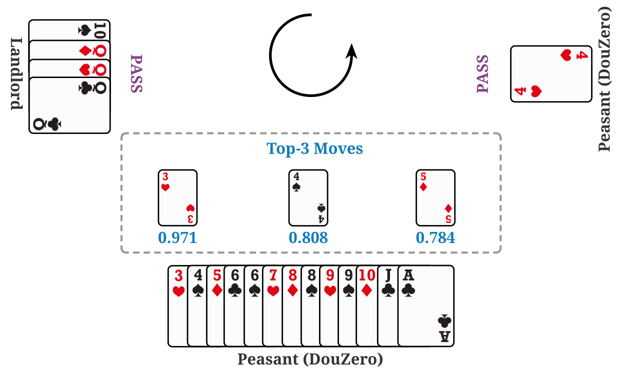


Figure 10. A case study dumped from Botzone, where the three players play cards in counter-clockwise order. The Peasant agent learns to play small Solo to cooperate with the other Peasant to win the game. Note that the other players' hands are showed face up solely for better visualization but are hidden in the real game. More case studies are provided in Appendix F.

Figure 10 shows a typical case when the two Peasants can cooperate to beat the Landlord. The Peasant on the right-hand side only has one card left. Here, the Peasant at the bottom can play a small Solo to help the other Peasant win. When looking into the top three actions predicted by DouZero, we make two interesting observations. First, we find that all the top actions outputted by DouZero are small Solos with high confidence to win, suggesting that the two Peasants of DouZero may have learned to cooperate. Second, the predicted Q-value of action 4 (0.808) is much lower than that of action 3 (0.971). A possible explanation is that there is still a 4 out there, so that playing 4 may not necessarily help the Peasant win. In practice, in this specific case, the other Peasant's only card is not higher than 4 in rank. Overall, action 3 is indeed the best move in this case.

6. Related Work

Search for Imperfect-Information Games. Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2008) is a leading iterative algorithm for poker games, with many variants (Lanctot et al., 2009; Gibson et al., 2012; Bowling et al., 2015; Moravčík et al., 2017; Brown & Sandholm, 2018; 2019a; Brown et al., 2019; Lanctot et al., 2019; Li et al., 2020b). However, traversing the game tree of DouDizhu is computationally intensive since it has a huge tree with a large branching factor. Moreover, most of the prior studies focus on zero-sum settings. While some efforts have been devoted to addressing the cooperative settings, e.g., with blueprint policy (Lerer et al., 2020), it remains challenging to reason about both competing and cooperation. Thus, DouDizhu has not seen an effective CFR-like solution.

RL for Imperfect-Information Games. Recent studies show that Reinforcement Learning (RL) can achieve competitive performance in poker games (Heinrich et al., 2015; Heinrich & Silver, 2016; Lanctot et al., 2017). Unlike CFR, RL is based on sampling so that it can easily generalize to large-scale games. RL has been successfully applied in some complex imperfect-information games, such as Starcraft (Vinyals et al., 2019), DOTA (Berner et al., 2019) and Mahjong (Li et al., 2020a). More recently, RL+search is explored and shown to be effective in poker games (Brown et al., 2020). DeltaDou adopts a similar idea, which first infers the hidden information and then uses MCTS to combine RL with search in DouDizhu (Jiang et al., 2019). However, DeltaDou is computationally expensive and heavily relies on human expertise. In practice, even without search, our DouZero outperforms DeltaDou in days of training.

7. Conclusions and Future Work

This work presents a strong AI system for DouDizhu. Some unique properties make DouDizhu particularly challenging to solve, e.g., huge state/action space and reasoning about both competing and cooperation. To address these challenges, we enhance classic Monte-Carlo methods with deep neural networks, action encoding, and parallel actors. This leads to a pure RL solution, namely DouZero, which is conceptually simple yet effective and efficient. Extensive evaluations demonstrate that DouZero is the strongest AI program for DouDizhu up to date. We hope the insight that simple Monte-Carlo methods can lead to strong policies in such a hard domain will motivate future research.

For future work, we will explore the following directions. First, we plan to try other neural architectures, such as convolutional neural networks and ResNet (He et al., 2016). Second, we will involve bidding in the loop for reinforcement learning. Third, we will combine DouZero with search at training and/or test time as in (Brown et al., 2020), and study how to balance RL and search. Fourth, we will explore off-policy learning to improve the training efficiency. Specifically, we will study whether and how we can improve the wall-clock time and the sample efficiency with experience replay (Lin, 1992; Zhang & Sutton, 2017; Zha et al., 2019b; Fedus et al., 2020). Fifth, we will try explicitly modeling collaboration of the Peasants (Panait & Luke, 2005; Foerster et al., 2016; Raileanu et al., 2018; Lai et al., 2020). Sixth, we plan to try scalable frameworks, such as SEED RL (Espeholt et al., 2019). Last but not least, we will test the applicability of Monte-Carlo methods on other tasks.

Acknowledgements

We thank our colleagues in Kuai Inc. for building the DouDizhu environment and the helpful discussions, Qiqi Jiang from DeltaDou team for helping us set up DeltaDou models, and Songyi Huang[8] from RLCard team for developing demo. We would also like to thank the anonymous reviewers and the meta-reviewer for the insightful feedback.

[8]https://github.com/hsywhu

References

Beleznay, F., Grobler, T., and Szepesvari, C. Comparing value-function estimation algorithms in undiscounted problems. *Technical Report TR-99-02, MindMaker Ltd*, 1999.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dkebiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. Heads-up limit hold'em poker is solved. *Science*, 347 (6218):145–149, 2015.

Brown, N. and Sandholm, T. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Brown, N. and Sandholm, T. Solving imperfect-information games via discounted regret minimization. In *AAAI Conference on Artificial Intelligence*, 2019a.

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

Brown, N., Lerer, A., Gross, S., and Sandholm, T. Deep counterfactual regret minimization. In *International Conference on Machine Learning*, 2019.

Brown, N., Bakhtin, A., Lerer, A., and Gong, Q. Combining deep reinforcement learning and search for imperfect-information games. *arXiv preprint arXiv:2007.13544*, 2020.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, 2018.

Espeholt, L., Marinier, R., Stanczyk, P., Wang, K., and Michalski, M. Seed rl: Scalable and efficient deep-rl with accelerated central inference. In *International Conference on Learning Representations*, 2019.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, 2020.

Foerster, J. N., Assael, Y. M., De Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016.

Gibson, R., Lanctot, M., Burch, N., Szafron, D., and Bowling, M. Generalized sampling and variance in counterfactual regret minimization. In *AAAI Conference on Artificial Intelligence*, 2012.

Gray, J., Lerer, A., Bakhtin, A., and Brown, N. Human-level performance in no-press diplomacy via equilibrium search. *arXiv preprint arXiv:2010.02923*, 2020.

Guo, Y., Oh, J., Singh, S., and Lee, H. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950*, 2018.

Hasselt, H. V. Double q-learning. In *Advances in Neural Information Processing Systems*, 2010.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, 2016.

Heinrich, J. and Silver, D. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Heinrich, J., Lanctot, M., and Silver, D. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*, 2015.

Jiang, Q., Li, K., Du, B., Chen, H., and Fang, H. Deltadou: Expert-level doudizhu ai through self-play. In *International Joint Conferences on Artificial Intelligence*, 2019.

Johanson, M., Waugh, K., Bowling, M., and Zinkevich, M. Accelerating best response calculation in large extensive games. In *International Joint Conferences on Artificial Intelligence*, 2011.

Küttler, H., Nardelli, N., Lavril, T., Selvatici, M., Sivakumar, V., Rocktäschel, T., and Grefenstette, E. Torchbeast: A pytorch platform for distributed rl. *arXiv preprint arXiv:1910.03552*, 2019.

Lai, K.-H., Zha, D., Li, Y., and Hu, X. Dual policy distillation. In *International Joint Conference on Artificial Intelligence*, 2020.

Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. Monte carlo sampling for regret minimization in extensive games. *Advances in Neural Information Processing Systems*, 2009.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in neural information processing systems*, 2017.

Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

Lerer, A., Hu, H., Foerster, J. N., and Brown, N. Improving policies via search in cooperative partially observable games. In *AAAI Conference on Artificial Intelligence*, 2020.

Li, J., Koyamada, S., Ye, Q., Liu, G., Wang, C., Yang, R., Zhao, L., Qin, T., Liu, T.-Y., and Hon, H.-W. Suphx: Mastering mahjong with deep reinforcement learning. *arXiv preprint arXiv:2003.13590*, 2020a.

Li, K., Xu, H., Zhang, M., Zhao, E., Wu, Z., Xing, J., and Huang, K. Openholdem: An open toolkit for large-scale imperfect-information game research. *arXiv preprint arXiv:2012.06168*, 2020b.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

Mania, H., Guy, A., and Recht, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

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

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

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

Panait, L. and Luke, S. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

Raileanu, R., Denton, E., Szlam, A., and Fergus, R. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine Learning*, 2018.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Szepesvári, C. Algorithms for reinforcement learning. *Morgan and Claypool*, 2009.

Terry, J. K., Black, B., Jayakumar, M., Hari, A., Sullivan, R., Santos, L., Dieffendahl, C., Williams, N. L., Lokesh, Y., Horsch, C., et al. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020.

Thrun, S. and Schwartz, A. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, 2016.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

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

Ye, D., Chen, G., Zhang, W., Chen, S., Yuan, B., Liu, B., Chen, J., Liu, Z., Qiu, F., Yu, H., et al. Towards playing full moba games with deep reinforcement learning. *arXiv preprint arXiv:2011.12692*, 2020a.

Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H., Yang, S., Wu, X., Guo, Q., et al. Mastering complex control in moba games with deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2020b.

You, Y., Li, L., Guo, B., Wang, W., and Lu, C. Combinational q-learning for dou di zhu. *arXiv preprint arXiv:1901.08925*, 2019.

Zahavy, T., Haroush, M., Merlis, N., Mankowitz, D. J., and Mannor, S. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

Zha, D., Lai, K.-H., Cao, Y., Huang, S., Wei, R., Guo, J., and Hu, X. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*, 2019a.

Zha, D., Lai, K.-H., Zhou, K., and Hu, X. Experience replay optimization. In *International Joint Conference on Artificial Intelligence*, 2019b.

Zha, D., Lai, K.-H., Zhou, K., and Hu, X. Simplifying deep reinforcement learning via self-supervision. *arXiv preprint arXiv:2106.05526*, 2021a.

Zha, D., Ma, W., Yuan, L., Hu, X., and Liu, J. Rank the episodes: A simple approach for exploration in procedurally-generated environments. In *International Conference on Learning Representations*, 2021b.

Zhang, S. and Sutton, R. S. A deeper look at experience replay. *NIPS Deep Reinforcement Learning Symposium*, 2017.

Zhou, H., Zhang, H., Zhou, Y., Wang, X., and Li, W. Botzone: an online multi-agent competitive platform for ai education. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pp. 33–38, 2018.

Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, 2008.