Applied Intelligence (2023) 53:15372–15389
https://doi.org/10.1007/s10489-022-04281-x

A Q-based policy gradient optimization approach for Doudizhu

Xiaomin Yu[1,2] · Yisong Wang[1,3] · Jin Qin[1] · Panfeng Chen[1]

Accepted: 18 October 2022 / Published online: 16 November 2022

Abstract
Deep reinforcement learning (DRL) has recently been employed in various games, with which superhuman intelligence has been achieved, including Atari, Go, no-limit, and Texas hold'em. However, this technique has not been fully considered for Doudizhu which is a popular poker game in Asia and involves confrontation and cooperation among multiple players with imperfect information. In this paper we present a new deep reinforcement learning approach NV-Dou for the game Doudizhu. It adopts a variant of neural fictitious self-play to approximate the Nash equilibria of the game. The loss functions of the neural network integrate Q-Based policy gradient (mean actor critic) with advantage learning and proximal policy optimization. In addition, parametric noises are adopted for the fully connected layers in the neural network. The experimental results show that it needs only a few hours of training and achieves almost state-of-the-art performance comparing with the well-known open implementations RHCP, CQL, MCTS and others for Doudizhu.

Keywords Deep reinforcement learning · Doudizhu · Multiple players · Nash equilibria · RHCP

1 Introduction

Deep reinforcement Learning (DRL) has achieved superhuman performance in various games, including AlphaGo [27] and AlphaGo Zero [28, 29] for the board game Go, the variant of counterfactual regret minimization (CFR$^+$) [5] for heads-up limit hold 'em poker, and Deepstack [23], Libratus [7], and Pluribus [6] for two and multiplayer heads-up no-limit hold 'em poker. While the board game Go has perfect information, where players can exactly observe the full state of the game, a poker game has only imperfect information.

Unfortunately, deep reinforcement learning has not been fully considered for Doudizhu, which is a typical three-player imperfect information poker game and is very popular in China with a large number of players. DeltaDou [15] is an exception to our best, it took 2 months to train its model on 68 CPUs and was not open. It remains a challenge to build a high level open AI for Doudizhu with reasonable resources.

To the best of our knowledge, nearly all of the current excellent DRL algorithms for poker games use Monte Carlo tree search (MCTS) [9, 14], counterfactual regret minimization (CFR), or their improved technology as prior knowledge. MCTS, which includes four processes (selection, expansion, simulation, and backpropagation), chooses actions according the upper confidence bounds (UCBs). The UCB is calculated by the processes of expansion, simulation, and backpropagation, and these processes require many computing resources; otherwise, it is difficult to achieve the desired effect. CFR chooses an action according a policy, which is calculated based on the regret value of each action in the game tree, so CFR is very difficult to use in extensive-form game scenarios with a large action space and state space. Therefore, it remains a

✉ Yisong Wang
yswang@gzu.edu.cn

XiaoMin Yu
musexiaoyu521@outlook.com

Jin Qin
jqin1@gzu.edu.cn

Panfeng Chen
gzupanda@outlook.com

1 College of Computer Science and Technology, Guizhou University, Guiyang, Guizhou, China

2 Guizhou Key Laboratory of Economics System Simulation, Guizhou University of Finance and Economics, Guiyang, Guizhou, China

3 Institute for Artificial Intelligence, Guizhou University, Guiyang, Guizhou, China

challenge to create a time-efficient reinforcement learning framework for Doudizhu.

In this paper, we propose a DRL framework for Doudizhu, that does not need prior knowledge, MCTS, or CFR. The framework can not only train a professional-level agent for Doudizhu but also can save considerable training time and computing resources. Our framework includes the aspects below so that it can be adapted to other three-players game environments like Doudizhu.

First, we propose a new exploration policy, that uses a noisy network with noisy buffers in our framework. The noisy buffers are used to select high-quality noise to help Peasant players explore new strategies. In the environment of a three-player game, it is easy to cause exploration disorder. The noisy buffer can ensure the stability of the two Peasants' explorations so that it can obtain a overall strategy for all agents' exploration. The results show that the noisy buffer is very effective for the improvement of the Peasants' policies. In addition, noisy buffers are very effective for solving the problem of exploration disorder in three-player game environments such as Doudizhu.

Second, we propose a new policy update method, that combines advantage learning with the Q-based policy gradient (mean actor critic) [2] in our framework. Unlike the advantage actor critic (A2C) [22] method that directly predicts the state value, our method uses the expected value of actions' probability distribution (actor network) and the state-actions' values (critic network) as the state value. In addition, we use a variant of proximal policy optimization (PPO) [26], and it controls the amplitude of policy updating and considers the part that is beyond the scope prescribed by PPO in the process of policy update. In this way, we can not only promise the stability of policy update but also ensure the integrity of the updated information. In our framework, we consider the removed part by normalization in addition to limiting optimization amplitude.

Finally, we propose a variant of Neural Fictitious Self Play (NFSP), which is used for the process of producing the experiences and the training of the models. In our framework, we remove the traditional NFSP method's average policy network. In this way, we can improve the stability of the overall three-player environment. The extensive experimental results show that it needs only a few hours of training and achieves almost state-of-the-art performance comparing with the well-known open implementations RHCP, CQL, MCTS and others for Doudizhu.

The rest of the paper is organized as follows. After a brief discussing the previous works in the next section, the game of Doudizhu is introduced in Section 3. In Section 4, the theoretical foundations and implementation methods of the framework are presented. Specifically, Doudizhu is formalized as an extensive-form game in Section 4.1; the state representation of Doudizhu is given in Section 4.2; in Section 4.3, an exploration policy with noise pools and noise selection method are introduced; and in Section 4.4, the details of the policy optimization are presented. In Section 5, the implementation details and experimental results are provided and analyzed. A broad comparison is presented in Section 6. Finally, the conclusion and future work are given in Section 7.

2 Related works

In recent years, there have been some works on Doudizhu. Even though some of them achieved good results, there is still room for improvement in terms of the winning percentage, training time, and computing resources. In addition, many of them need prior knowledge; otherwise, these methods cannot achieve ideal results.

The MCTS algorithm for Doudizhu was proposed by Powley, Whitehouse, and Cowling in 2011 [24]. Its main idea is to calculate the UCB value of each node (a node represents a kind of state in our game scenario) by the process of selection, expansion, simulation, and backpropagation. Finally, the agents select actions based on the UCB of each state. Some amount of time for the determination because of the properties of Monte Carlo. In addition, MCTS is DeltaDou's [14] method for control and optimization.

RHCP[1] is an algorithm of Doudizhu based on actions' values, and it is also an important baseline in the field of Doudizhu research. The main idea of RHCP is that the action performed can maximize the total value of the remaining hand cards. First, RHCP enumerates all alternative actions and stores them in a list. Second, for each action in the list, the algorithm calculates the value of the remaining hand cards except those in the action. Finally, the algorithm chooses the action that makes the total value of the remaining hand cards maximum compared to other actions. The algorithm calculates the total value of hand cards by splitting it into the smallest combination of rule actions. Splitting the hand cards is computationally expensive and will slow down the response speed of the algorithm. To reduce computation consumption, the author added some fixed playing rules into the algorithm. There are two fixed rules, which apply to both the active and passive card playing process. First, the algorithm will choose the action directly, which is consisted of all the cards in the hand. Second, the algorithm performs the action whose probability of being suppressed is very low when the hand cards can only be split into two actions. For the actively playing cards, additional rules are adopted. If the alternative

[1]https://ninesun.blog.csdn.net/article/details/70787814

actions contain trio with solo, trio with pair, plane with solo, or plane with pair, the other actions are not used for action selection calculations. If the alternative actions do not contain the above priority actions, then the algorithm selects one card whose value is least, and it is not included in a bomb. If all alternative actions are bombs or rockets, go straight for the least valuable bomb. In terms of the capability and response time, RHCP still has room for improvement.

Combinational Q-learning for Doudizhu (CQL) [33] makes use of pokers splitting and Q-Learning [31]. In the process of training, there are two models in the method: the decomposition proposal network (DPN) and move proposal Network (MPN). The DPN chooses the best decomposition combination of hand cards, and each element of the combination is a Doudizhu's rule action. Then the MPN chooses the action in the decomposition combination. The two choices are made based on Q-learning [31]. The action space is effectively reduced by the DPN so that the difficulty of making action choices is greatly reduced. The results show that this makes the Q-leaning converge to a good result. However, much time is spent on poker splitting in this method. Even though the method uses a dancing link [17] to split the pokers to find legal card groups, the computational complexity remains high because of the large action space of Doudizhu. In the experiment with this method, the authors used a 32-core AMD Threadripper CPU and 1080Ti GPU, and they spent twenty days training their model.

DeltaDou is a satisfactory algorithm to the best of our knowledge [15]. It extends AlphaGo Zero [28] to a multiplayer incomplete information game. In DeltaDou, a separate game tree was created for each player for simulation and calculation of upper confidence bound applied to trees (UCT), and a policy network is used to approximate opponent responses during simulation. The type of tree search is called Fictitious Play MCTS. In addition, there is a component, which uses Bayesian methods to infer hidden information, and it can effectively assist the decision of the specified player. However, the calculation of UCT and the Bayesian inference are computationally expensive. Even if 68 CPUs were used, two months was still spent to achieve the desired results, and our algorithm uses much less computing resources.

The playing strategy network (PSN) [30] for Doudizhu was proposed in 2021. It adopted the approach of supervised learning to predict an action based on the current state. In this approach, the author used the residual network so that the prediction accuracy was improved compared to the authors' previous work [21]. However, the ability of the model in this method depends on the quality of the data set, and the model can't improve itself through self-play.

In the same year that PSN was proposed, a method that combines self-play and supervised learning was used for Doudizhu [20]. In this approach, each role's winning data are stored in the corresponding buffer after self-play. Then, the data are used to train the corresponding model in a supervised learning manner. In this method, the number of actions of Doudizhu is abstracted to 182, and it ignores actions such as trio with single, trio with pair, plane with solo, plane with pair, quad with solo and quad with pair.[2]

CFR is an excellent iterative method to solve poker problems, and the CFR$^+$ has been used to solve heads-up limit hold 'em poker in 2015 [5]. The main idea of CFR is that use the regret value for a decision, and the regret value of each action at the decision point is calculated by traversing the game tree. The main challenges for this approach are computation (traverse the game tree) and storage (store the regret value and average strategy during the computation), and the challenge is even more difficult for the Doudizhu problem with huge state and action space. Thus, CFR's solution to Doudizhu has not been seen.

Recently, a superhuman AI for multiplayer poker was proposed [8]. It uses Monte Carlo counterfactual regret minimization (MCCFR) [18] as a priori knowledge for exploration and optimization. In general, MCCFR allocates the terminal history to some blocks, and the union of all blocks contains all terminal history. In each iteration, the algorithm only updates the information set of each history in the sampled block. The key advantage of MCCFR is that it avoids traversing the entire game tree while getting regret values be unchanged in expectation so that it is very effective for reducing computational complexity. However, the performance of MCCFR depends on the sampling profile of blocks. In contrast, our algorithm has no special requirements on the sampling distribution. In addition, there has been some work on quantifying sophistication of Doudizhu [12, 13]. In this work, the concept of game refinement (GR) was used to quantify the sophistication of Doudizhu, and it provides a different perspective to better understand the design and direction of optimizing the game.

3 The game of Doudizhu

Doudizhu[3] is a very popular card game in Asia with three players. In the game, the roles of the three players are the Landlord, Peasant Up, and Peasant Down. In the process of the game, players play their hands in counterclockwise order. The landlord plays first. Peasant Up is the player on

[2]http://rlcard.org/games.html#action-abstraction-of-dou-dizhu

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

the right before Landlord, while Peasant Down is the player on the left after Landlord.

The game of Doudizhu uses a deck of 54 cards, which contains 15 different types. They are 3, 4, 5, 6, 7, 8, 9, 10 (T), J, Q, K, A, 2, black joker (B), and red joker (R), which are ordered by their ranks from low to high. There are four cards (heart, spade, club and diamond) of each type, except for the black joker and red joker.

At the beginning of the game, each player gets 17 cards at random, and the cards are only known by the owner. In addition, three public cards are known by all players. Next, every player takes turns to bid for the three cards, and the bid of the player who wants to obtain the three cards must be higher than current bids. At each time, everyone has the option to pass. The player, that obtains the three cards becomes the Landlord when the other players pass consecutively. The winning bid determines the stake for the game. A special case that exists in the bidding process is that no one wants the three cards. In this case, the game will restart.

The goal of the game is to be the first to play all the cards in one's hand. However, if a Peasant defeats the Landlord, the two Peasants receive the reward paid by the Landlord. In other words, the goal of the two Peasants is consistent. Therefore, the Peasants can cooperate to beat the Landlord. In the beginning, the Landlord plays his cards first and then the other players play their cards in a fixed order.

4 Method

In this section, we introduce the components of our Doudizhu framework and how we improve existing technologies to enable the framework to strike a balance between exploration and exploitation, stable optimization, and stable training in the three-player game.

To balance exploration and exploitation, we use a noisy network to determine the exploration policies, and we proposnoise buffers for the two Peasants to reduce the randomness of noise generation to ensure exploration and stability. In Fig. 1, Fully Connected Layer 1 and Fully Connected Layer 2 are noised to be responsible for estimating the state-action values and the probability distribution of the policy.

To solve the problem of stable updating, we adopted a Q-based policy gradient (e.g., mean actor critic) [2] method combining advantage learning [4] and a variant of PPO to update the policy to ensure the stable updating of each player's policy and the overall stability of the three-player game environment.

To solve the problem of stable training, we propose a variant method of NFSP for training. In this process, the average policy network of the traditional NFSP is removed to reduce the uncertainty during training.

4.1 Extensive-form games of doudizhu

Extensive-form games are a model of sequential interaction involving multiple agents, the representation of which is based on a game tree. The *extensive-form game* for Doudizhu is a tuple $\langle \mathcal{I}, \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where

- $\mathcal{I} = \{1, 2, 3\}$ indicates the three players in Doudizhu, in which 1 is for the landlord, 2 indicates Peasant Down and, 3 indicates Peasant Up;
- $\mathcal{S}$ is the state space of players;
- $\mathcal{A}$ is the joint action space of players;
- r is the reward function for all agents, $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$;
- p is the state transition probability defined as $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ and, $p\left(s, \mathbf{a}, s'\right)$ (or $p(s'|s, \mathbf{a})$) is the probability of executing action a at state s, resulting in the next state s';
- $\gamma \in [0, 1]$ indicates the discount factor.

$\mathcal{S}$, $\mathcal{A}$ and r can be factorized into an agent's state, action space and reward function as $\mathcal{S}^i$, $\mathcal{A}^i$ and r^i $(1 \leq i \leq 3)$. At each timestep, an agent takes an action $a^i \in \mathcal{A}^i$, forming a joint action $\mathbf{a} \in \mathcal{A} = \times_{\{i=1,2,3,\cdots,N\}}\mathcal{A}^i$, and the reward of each agent is $r^i(s, \mathbf{a})$. Note that $r^1(s, a) = -2r^2(s, a) = -2r^3(s, a)$ for any terminal state s. Thus, the Douizhu game is zero-sum and extensive-form.

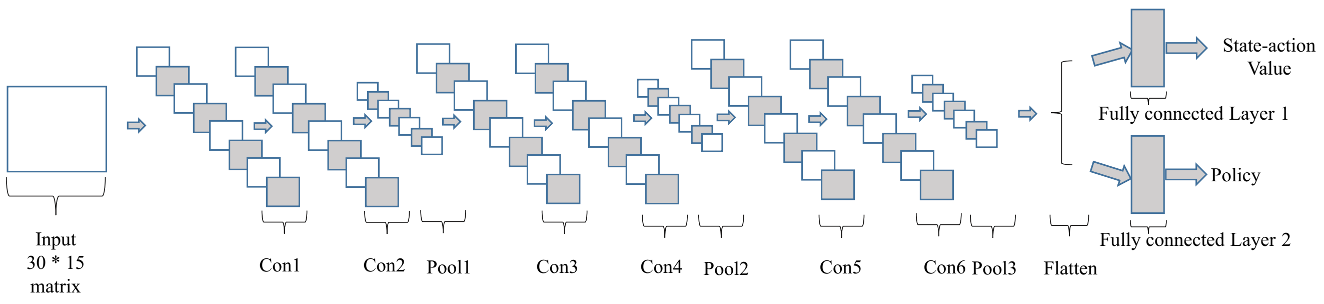
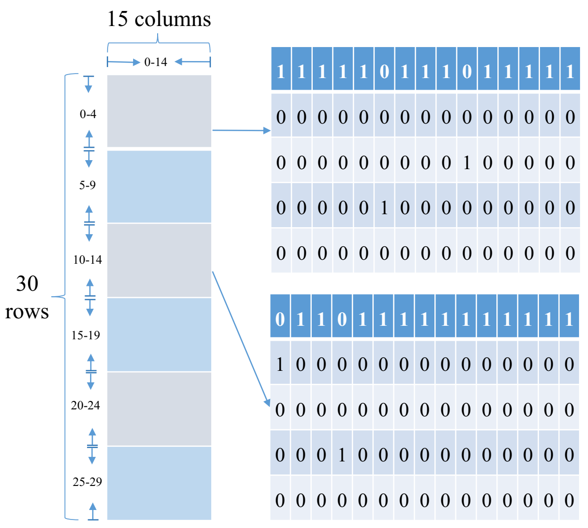


Fig. 1 The convolutional neural network structure for Doudizhu

Line	Feature
0-4	The current hand cards of the player
5-9	The union of the other two players' hands
10-14	The previous player's action
15-19	The action of the player before previous one
20-24	The last action of the current player
25-29	The union of all played cards on the table



1	1	1	1	1	0	1	1	1	0	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2 The state representation for Doudizhu

4.2 State representation

In our framework, we use a noisy network to predict the probability distribution of rule actions (policy) and state-action values. Figure 1 shows the overall structure of the network. The input of the network is a state of the game, which represents the current situation that a player faces. We use a matrix of size 30×15 to represent a state. Each entry of the matrix is taken as either 1 or 0. In Fig. 2, each of the five rows in the matrix forms a group, whose meaning is illustrated on the left. The 15 columns correspond to poker card types, which range from 3 to the red joker, and the rows show whether a poker exists and the number of pokers. The first row represents whether a poker exists (0 and 1 represent existence and nonexistence, respectively). The second to fifth rows represent the number of pokers. In particular, there is one card of a certain type if the value in the second row is 1. If the value in the fifth row is 1, it means that there are four cards of that type. In addition, the meaning of each group is shown on the left side of Fig. 2 There are six groups in the matrix. The representation of each group is given in RLCard [34] format. In Fig. 2 the detail of first group shows that the current player's cards are 8, 8, 8, Q, Q. The detail of third group shows that the previous player's action is 3, 6, 6, 6 (trio with solo).

4.3 Exploration policy

The exploration policy plays a key role in avoiding local optimization. We propose a new strategy based on noisy network for exploration [11] to achieve a better balance between exploration and exploitation, and to make the three-player game environment more stable.

The agent's policy is interfered with by adding noise to the fully connected layers of the network. Figure 3 illustrates the process of adding noises to the full connected layer, which has m inputs and i outputs.

In the fully connected layer, the parameters and biases include $\mathbf{W}$ and $\mathbf{b}$ in the following equation:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the input of the layer, $\mathbf{W} \in \mathbb{R}^{n \times m}$ is the weight matrix, and $\mathbf{b} \in \mathbb{R}^n$ is the bias vector. In the process of training, the layer's original parameters and noise are optimized together. Let $\mathbf{\Sigma}$ and σ be the weight of noise for $\mathbf{W}$ and $\mathbf{b}$ respectively. They are usually a constant matrix and vector. Let $\mathbf{E}$ and ε be the noises. The noised

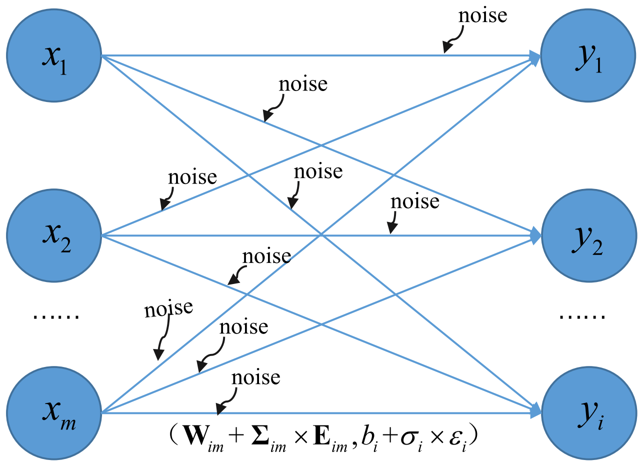

x_1
x_2
……
x_m
y_1
y_2
……
y_i
noise
noise
noise
noise
noise
noise
noise
noise
noise
$(\mathbf{W}_{im} + \mathbf{\Sigma}_{im} \times \mathbf{E}_{im}, b_i + \sigma_i \times \varepsilon_i)$

Fig. 3 The structure of the fully connected layers with noises

fully connected layers are then computed in terms of the following equation:

$$\mathbf{y} = (\mathbf{W} + \mathbf{\Sigma} \odot \mathbf{E})\mathbf{x} + \mathbf{b} + \sigma \odot \varepsilon \tag{2}$$

where the dimensions of $\mathbf{\Sigma}$ and $\mathbf{E}$ are the same as the parameter matrix $\mathbf{W}$, and σ and ε have the same size as that of $\mathbf{b}$. The operator $\odot$ is element-wise multiplication. Each noise consists of parameters' noise ($\mathbf{E}$) and biases' noise (ε^b). In Fig. 3, the tuple at the bottom of the figure shows the parameters and biases after adding noise to $\mathbf{W}_{im}$ and b_i.

Two buffers are created for each Peasant. One is used to store its noise and the other is used to store its score (the two-peasant score is the sum of the Peasant's scores) corresponding to its noise. The noise is randomly generated or chosen from the noise buffer in terms of the probability distribution of its scores, that are obtained by our model against the baseline model RHCP.[4] Formally, let the noise buffer and the score buffer be two sets $\Psi_n = \{\psi_n^1, \psi_n^2, ..., \psi_n^J\}$ and $\Psi_s = \{\psi_c^1, \psi_c^2, ..., \psi_c^J\}$, respectively. The probability of the j-th noise being selected from the noise buffer is:

$$\frac{\psi_c^j}{\sum_{j=1}^{J} \psi_c^j}, \quad j = 1, 2, \ldots, J. \tag{3}$$

In the beginning of training, the noise buffer is not very useful because of the randomness of its initialization policy. In the later stages of training, there is little change in strategy because of the very small error. Thus, it is better to select noise from the noisy buffer according to its past performance than to randomly generate noise. Since the exploration strategy chosen by the two Peasants in this method is the one with the best performance of the two strategies together, it has great significance for escaping the disorder of the Peasants' exploration. The convergence and stability of the network will also be greatly improved because choosing the noise in the buffer greatly reduces the randomness of the whole environment.

4.4 Q-based policy optimization

To ensure the stability and convergence of the network, our policy optimizing method integrates Q-based Policy Gradient, A2C [4] and a variant of PPO [26], named QAAC.

Let θ be the parameters of the network including the noise parameters. Unlike A2C, our critic model is used to compute the estimated value of state-action value $Q(s, a; \theta)$. The actor's probability distribution of actions $\pi(a|s; \theta)$ is adjusted according to the advantage of $Q(s, a)$ over the state value $V(s; \theta)$:

$$V(s; \theta) = \sum_a Q(s, a; \theta) * \pi(a|s; \theta) \tag{4}$$

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

where $Q(s, a; \theta)$ is the estimated value of action a in state s. The advantage $A(s, a; \theta)$ of the state-action $Q(s, a)$ is defined as:

$$A(s, a; \theta) = Q(s, a) - V(s; \theta) \tag{5}$$

where $Q(s, a)$ is the true value of action a that is executed in the state s. In what follows, the subscript t denotes "at time step t" for convenience. To update the policy in the proper scope, the PPO technique is adapted according to

$$\eta_t(\theta) = \text{clip}(\kappa_t(\theta), 0, 1 + \epsilon) + \max\left(0, 1 - \frac{1 + \epsilon}{\kappa_t(\theta)}\right) \tag{6}$$

where

- ϵ is a hyperparameter;
- $\text{clip}(\kappa_t(\theta), 0, 1 + \epsilon)$ is to remove the incentive for moving $\kappa_t(\theta)$ outside of the interval $[0, 1 + \epsilon]$;
- $\max(0, 1 - \frac{1+\epsilon}{\kappa_t(\theta)})$ is to regularize $\text{clip}(\kappa_t(\theta), 0, 1 + \epsilon)$;
- $\kappa_t(\theta)$ is the probability ratio

$$\frac{\pi\ (a_t \mid s_t; \theta)}{\pi\ (a_t \mid s_t; \theta_{old})} \tag{7}$$

with θ_{old} being the vector of policy parameters before the update.

The actor's optimization objective is then defined as:

$$L^{ACTOR}(\theta) = \hat{\mathbb{E}}_t\ (\eta_t(\theta) \log \pi(a_t \mid s_t; \theta) A(s_t, a_t; \theta)) \tag{8}$$

The objective function of the critic in the actor-critic framework is then defined as:

$$L^{CRITIC}(\theta) = \hat{\mathbb{E}}_t \left[(Q(s_t, a_t) - V(s_t; \theta))^2 \right]. \tag{9}$$

4.5 Strategies and equilibria

As an important concept of Game Theory, Nash equilibrium has been successfully applied in many fields such as politics, biology, and sports. In politics, Nash equilibrium is adapted to design the model of multi-party, competition under proportional rule [25]. In biology, the evolutionarily stable strategy (ESS) [10] who comes from Nash equilibrium is an important means of analyzing adaptive strategies of organisms. In sports, strategic sports games like American football [32], baseball [1], and soccer [3] are formalized as game processes that include competition and cooperation and achieve Nash equilibrium. In the sports game processes, all players adjust their actions for the overall benefit of the team until they can not increase the team's reward by changing their behavior (the strategy profile satisfy Nash equilibrium at this moment). The adjustment of each player's action is fulfilled by against with the opponents based on the specific circumstances of the match and the different payoff values of the players' preference. Similar to the above sports games, two teams (landlord and farmers) in Doudizhu adapt their behaviors until no

```
Input: The following are hyperparameters:
$\xi$: the number of iterations; $\gamma$: the discount factor; $\omega$: the learning rate; $bs$: the batch sizes; $\alpha$: noise selecting
probability; $bfs$: the length limit of experience buffer; $nfs$: the length limit of noise buffer; $f$: noise update frequency;
$nw$: noisy weight; $om$: the opponent model for test in training; $tn$: the number of tests against $om$
/* $i=1$: the landlord, $i=2$: the peasant down, $i=3$: the peasant up */
1  Initialize the network with random parameters $\theta_i$ which includes $\theta_i^\pi$ and $\theta_i^q$ $(i = 1, 2, 3)$;
2  $M_i^{RL} \leftarrow []$,  $N\_M_i \leftarrow []$,  $S\_M_i \leftarrow []$,  $r_i \leftarrow 0$,
3  $N\_M_i.append($a random noise $(\mathbf{E}_i, \varepsilon_i)$ for the full-connected layers of $\theta_i)$,  $(i = 1, 2, 3)$ ;
4  foreach $it = 1, \ldots, \xi$ do
5      $M_i \leftarrow []$ $(i = 1, 2, 3)$;
6      Initializes the hand card sets of three players in a game $\Gamma$;
7      $t_i \leftarrow 0$ $(i = 1, 2, 3)$;
8      $i \leftarrow 0$;
9      repeat
10         $i \leftarrow (i + 1)\%3 + 1$; $t_i \leftarrow t_i + 1$;
11         Execute action $a$ at the current state $s$ of $\Gamma$ according to the policy $\pi(a|s; \theta_i^\pi)$;
12         $M_i.append([s, a, 0, \pi(\cdot|s; \theta_i^\pi)])$;
13     until the game $\Gamma$ is over ($i$ is the winner);
       /* set the discounted reward of winner(s), and the winners' rewards are 1 */
14     if the landlord wins the game $\Gamma$ then
15         $M_1[k][2] \leftarrow \gamma^{t_1-k+1}$ $(k = 0, \ldots, t_1 - 1)$;
16     else
17         $M_2[k][2] \leftarrow \gamma^{t_2-k+1}$ $(k = 0, \ldots, t_2 - 1)$; $M_3[k][2] \leftarrow \gamma^{t_3-k+1}$ $(k = 0, \ldots, t_3 - 1)$;
18     end
19     $M_i^{RL} \leftarrow M_i^{RL} + M_i$ $(i = 1, 2, 3)$;
20     if $len(M_i^{RL}) > nbf$ then Remove the first $len(M_i^{RL}) - bfs$ elements of $M_i^{RL}$  $(i = 1, 2, 3)$;
21     foreach $i = 1, 2, 3$ do
22         Sample a batch size $bs[i]$ of samples $D$ from $M_i^{RL}$;
23         Compute $\kappa(\theta_i^\pi)$ for each sample of $D$ according to (7);
24         Update $\theta_i^\pi$ with gradient descend and learning rate $\omega$ on loss according to (8);
25         Update $\theta_i^q$ with gradient descend and learning rate $\omega$ on loss according to (9);
26     end
27     if $it$ % $f = 0$ then
28         $S\_M_1.append($The overall scores of $\theta_1^\pi$ against the model $om$ as peasants in $tn$ runs$)$;
29         $S\_M_2.append($The overall scores of $\theta_2^\pi$ and $\theta_3^\pi$ against the model $om$ as landlord in $tn$ runs$)$;
30         $S\_M_3 \leftarrow S\_M_2$;
31         foreach $i = 1, 2, 3$ do
32             if $len(S\_M_i) > nbs$ then Remove the first element of $S\_M_i$ and $N\_M_i$;
33             if random()$\leq \alpha$ then
34                 $E_i, \varepsilon_i \leftarrow$ randomly choose a noise from $N\_M_i$ in terms of $softmax(S\_M_i)$;
35             else
36                 $E_i, \varepsilon_i \leftarrow$ a random noise for the full-connected layers of $\theta_i$;
37             end
38             $N\_M_i.append((E_i, \varepsilon_i))$
39             add the noise $E_i, \varepsilon_i$ to $\theta_i^\pi$ and $\theta_i^q$ with the weight $nw$ according to (2);
40         end
41     end
42 end
```

Algorithm 1 NV-Dou.

improvement of rewards, that can be formalized as a game to achieve Nash equilibrium as follows.

A *strategy of player* i, written π_i, is a function that assigns a distribution over the actions of i to the state of player i. By Π_i we denote the set of all strategies of player i. A *strategy profile* π consists of a strategy π_i for each player, i.e. $\pi = \{\pi_1, \pi_2, \pi_3\}$. By π_{-i} we denote all the strategies in π except π_i. A *history* of player i is a finite sequence $(a_1^i, a_2^i, \ldots, a_k^i)$ of actions of player i. In particular, if a_k^i is the last action of player i in a game then the history is *terminal*, and it is *nonterminal* otherwise. By Z^i we denote the set of terminal histories of player i. The overall value to player i of a strategy profile π is then the expected payoff of the resulting terminal state:

$$u_i(\pi) = \sum_{z^i \in Z^i} u_i(z^i) p^\pi(z^i) \tag{10}$$

where $u_i(z^i)$ is the cumulative reward of player i reaching history z^i and $p^\pi(z^i)$ is the probability of producing the history z^i of player i according to policy π.

We denote $\beta(\pi_{-i})$ the best response policy of player i with respect to the other players' policies that can maximize the expected payoff and the value of the strategy:

$$b_i(\pi_{-i}) = u_i(\beta(\pi_{-i}), \pi_{-i}) = \max_{\pi_i^* \in \Pi_i} u_i\left(\pi_i^*, \pi_{-i}\right) \tag{11}$$

where π_i^* is the optimal policy of the optimized process, and the Nash equilibrium satisfies a strategy profile:

$$u_i(\pi_i, \pi_{-i}) \geq b_i(\pi_{-i}) \qquad i \in \{1, 2, 3\} \tag{12}$$

where each agent's expected reward is greater than or equal to the player's best response.

In traditional NFSP [16], there are two policies (the best policy and average policy). The best policy and average policy are represented by DRL model ($\mathbb{M}_{RL}$) and supervised learning model ($\mathbb{M}_{SL}$) respectively, and $\mathbb{M}_{SL}$ is used to approximate the best responses of the agents using the supervised classification of the data. However, in the process of optimal response approximation, the uncertainty of the overall environment increases so that the $\mathbb{M}_{SL}$ is not suitable for a three-player game environment such as Doudizhu.

In our framework, we only use the $\mathbb{M}_{RL}$ to achieve the aim reaching the Nash equilibrium and the results show that our framework is better than NFSP. At the same time, our framework achieves good results without the components CFR and MCTS. The overall framework of our Doudizhu AI, named NV-Dou, is shown in Algorithm 1.

In the pseudocode, lines 1 to 3 initialize the network parameters, variables and buffers. The parameters θ_i^π and θ_i^q represent the parameters of player i's actor network and critic network respectively. The buffers N_M_i and S_M_i store the noisy parameters and the noisy parameters' scores respectively. The buffer M_i^{RL} store all historical data of player i. On line 5, the buffer M_i keeps the data of a complete historical track to compute the backup value of the final result. Lines 8 to 12 obtain the data of experience. Lines 13 to 20 are the processes of determining final dismissal and back propagation. Lines 21 to 26 are the process of updating the networks' parameters, and it is also the process of improving the policy to obtain a greater reward until $b_i(\pi_{-i})$ is obtained. The optimal response policy $(\beta_i(\pi_{-i}))$ is obtained, when $b_i(\pi_{-i})$ is achieved. Lines 27 to 41 are the processes of selecting and adding noise to the network.

5 Experiments

In this section, the implementation of NV-Dou and its experiments are reported in detail. In particular, NV-Dou is compared with the basic A2C, QAAC and QANFSP. It is also compared with the open implementations[5] RHCP, CQL, MCTS, Random and Rule. According to the cards in the hand and the last played cards in the deck, the 'Random' approach randomly chooses an action to play, while the 'Rule' approach chooses the the least rank action to play. NV-Dou is not compared with DeltaDou due to its inaccessibility.

5.1 Environmental settings

In this paper, we use RLCard [34] as the environment, and we can train our method by setting our AI algorithm to the Doudizhu environment (RLCard). In the environment, Doudizhu's actions are abstracted into 309 kinds. There are a large number of actions in Doudizhu. The environment ignores some details of the actions. For example, a trio with an individual card, 3334, 3335, and 3336 can be regarded as the same class. The action space can be reduced by this method. In Table 2 of Appendix , the actions[6] are described in detail. In addition, the actions' IDS are shown. The implementation and experiments with our NV-Dou can be found at Github.[7]

5.2 Implementation details

There are three agents in our training process, one Landlord and two Peasants. We use the computer with one 8-core 8-thread CPU and one RTX2060 GPU (the performance is equivalent to that of the GTX1070) to run our experiments.

[5]https://github.com/qq45cvb/doudizhu-C
[6]http://rlcard.org/games.html#action-abstraction-of-dou-dizhu
[7]https://github.com/qq-ship/NV-Dou

Table 1 Hyperparameters in our experiment

Name	Value
Number of iterations ξ	1 million
Learning rate ω	0.0005
Decay factor γ	0.9
Buffer size bfs	50000
Noisy weight nw	0.08
ϵ of actor	1.0
The probability choosing noise α	0.4
Noisy buffer size nfs	20
Noise updating frequency f	2000
Batch size for training landlord's network	64
Batch size for training peasants' networks	32

In this paper, we adopt a convolution network as the actor-critic network to approximate the state-action value and the policy that is the actions' probability distribution under a state. The network uses six convolution layers. Each convolution layer has 64 kernels of size 3 $\times$ 3, and the activation function tanh. There is a pooling layer with 64 kernels of size 2 $\times$ 2 between every two convolution layers. In addition, The fully connected layers are of size 309. In Fig. 1 shows the whole construction of the network. The fully connected layer was used with the noise, and the noise was also optimized. The noise was randomly generated once per 2000 iterations. In the process of training, we used the Adam method to update the network's parameters. The batch sizes of Landlord and Peasants were 64 and 32, respectively, and we obtained the mean of the data as losses. In our experiment, we trained the agents for 1 million episodes. All hyper-parameters are presented in Table 1.

5.3 Comparison with the basic method

In this section, we show the performance of the method that we improve, and compare it with the basic methods. In our paper, we create a framework that adapts to Doudizhu, and it is based on A2C [19] and NFSP. We name our policy optimization method QAAC because it is a combination of the Q-based policy gradient and A2C.

We performed three experiments, and the methods that we used are our NV-Dou (NV-Dou combines QAAC with a variant of the NFSP framework), A2C, and QAAC with traditional NFSP (QANFSP). In the experiment, we trained a million times for each method, and we tested them every 2000 iterations. We used the RHCP as the opponent for testing. Figure 4 shows the results for the three methods playing the roles of the Landlord (left) and Peasants (right), The red, green and yellow lines represent the winning rates of NV-Dou, A2C, and QAAC with traditional NFSP (QANFSP). We find that our improved method achieves much better results than the basic method, and it is very obvious at the training of Peasants. Additionally, the NFSP variant significantly improved overall winning percentage. In sum, our improved method greatly improved the stability of exploration, optimization and training and could solve the problems of exploration disorder and falling into local optima.

5.4 Comparison to the Doudizhu baseline

In this section, we evaluate the power of our algorithm by comparing it with the baseline algorithms. Since there is no public rank line model for Doudizhu, we set our model against other open algorithms (Random, Rule, CQL [33],

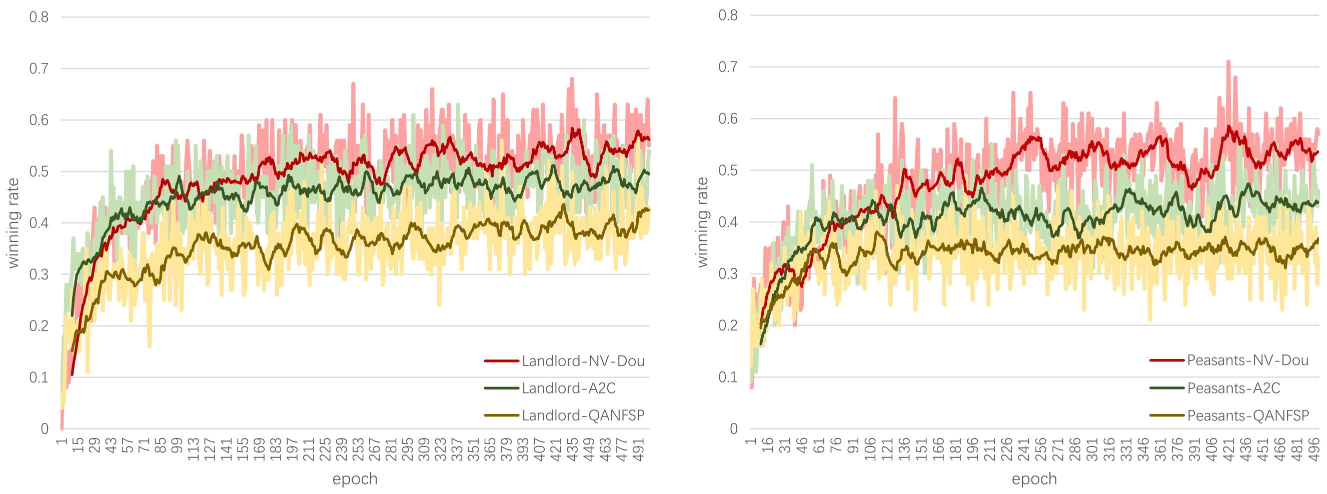


Fig. 4 The learning curve of NV-Dou, A2C, and QAAC with the traditional NFSP(QANFSP)

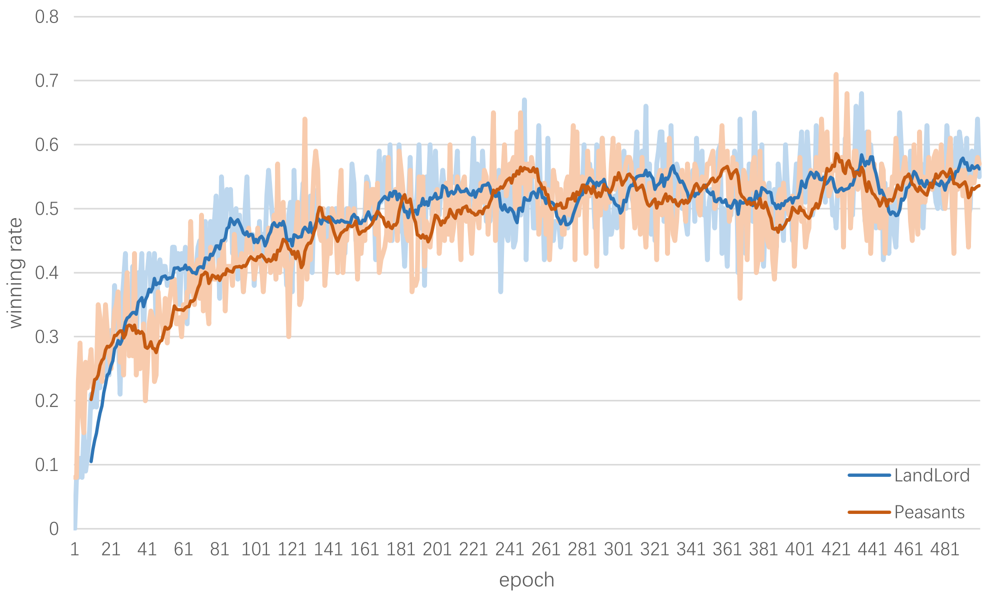


Fig. 5 The winning rate against RHCP in the training of NV-Dou

RHCP,[8] MCTS [24]) playing different roles (Landlord and Peasants), and evaluate the performance of our approach. In our method, the two Peasants worked together by sharing rewards, and we treat Peasant Up and Peasant Down as a whole (Fig. 5).

Figures 6 and 7 show all the testing winning rates and scores for our algorithm playing the Landlord and Peasants against the baseline algorithms respectively. In the scenario of Doudizhu, the Landlord's score is the sum of the scores lost by the two Peasants if the Landlord wins the game and vice versa. In addition, the score will double, every time a bomb or a rocket comes up. We play the games 100 times per epoch, and the winning rates and scores are the average values of 20 epochs.

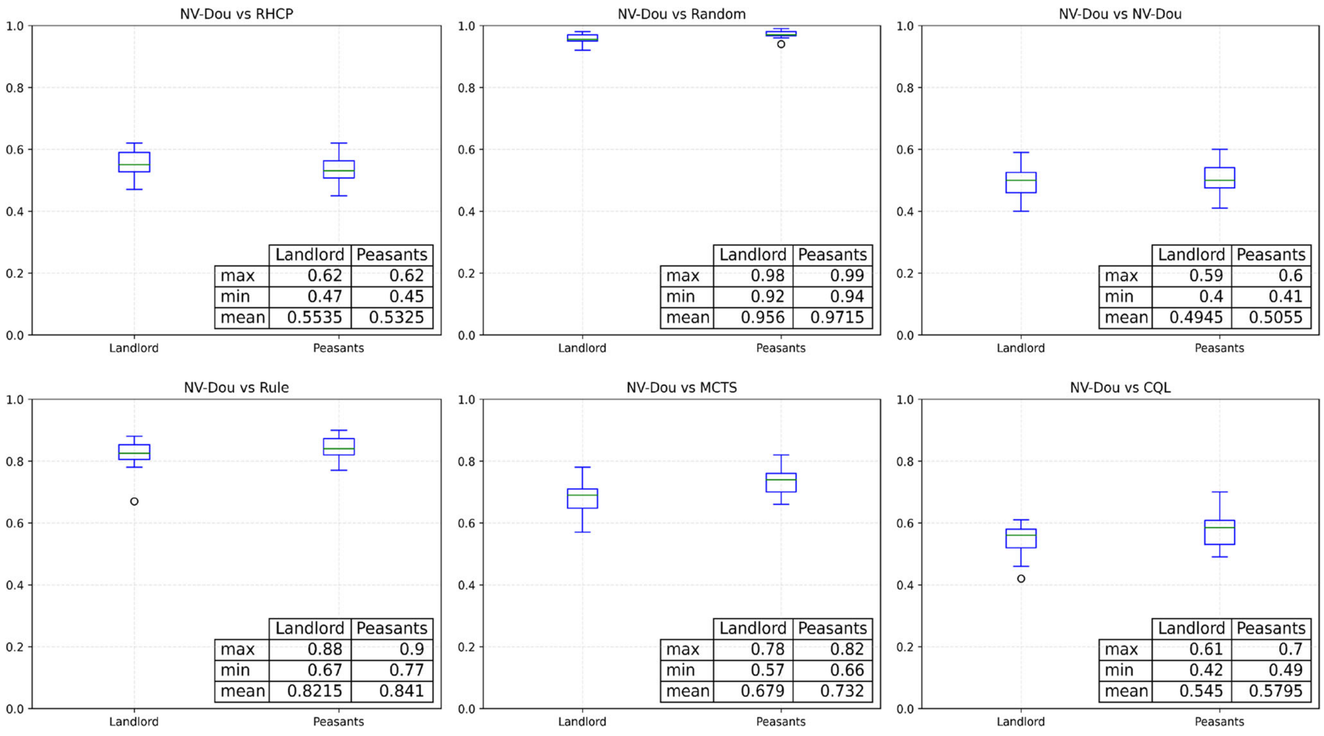
According to the winning rates and scores, we can see that no matter what role our algorithm plays, it is far better than the Random (according to all cards in the hand, it splits all the actions that can be played, then chooses an action randomly among them to play) and Rule algorithm (According to all cards at hand, it splits out all the actions that can be played, and chooses the action that has the lowest rank among them to play) in terms of the winning rates and scores.

In the testing against RHCP, our algorithm could beat RHCP, regardless of the role it played (Tables 3, 4 and 5 of Appendix). The Landlord's average winning rate and the score of the epoch were 0.5535 and 24.3, and the Peasants' winning rate and the score of the epoch were 0.5325 and 12.2. DeltaDou [15] was the best-known algorithm, that solves the problem of Doudizhu in terms of winning rates until now. However, the details that were made public in the paper [15] cannot support the algorithm reconstruction. Therefore, we directly cite the paper's results against RHCP for comparison. In the process of training, the highest winning rates (winning times in one hundred games) of the Landlord and Peasants are 0.68 and 0.71, and they are close to DeltaDou's winning rate (0.75) shown in the paper [15]. In addition, we calculated 66 hours for our model's training, and the training time of DeltaDou is two months. In addition, the equipment that we used contains only one 8-core 8-thread CPU, and one RTX2060 GPU (the performance is equivalent to a GTX1070). Compared to the equipment that was shown in the paper [15] for DeltaDou's training, the computing power of our equipment is far inferior.

The main RHCP idea is assigning a value to each action and dividing the cards in the hand into the fewest rule actions of Doudizhu. Then, it takes the action based on the current rule. In addition, the action that was taken must maximize the total value of all remaining actions. We know that RHCP only considers the impact of current actions on the next state, and does not consider the impact on the final result. This is the main reason why our algorithm is superior to RHCP. Since the value for each action of a state is determined from the decay value of the final result, every action our algorithm takes is designed to ultimately win. In addition, RHCP does not consider collaboration. In contrast, the peasants in our algorithm collaborate by sharing rewards. This advantage is also a reason why our algorithm is superior to RHCP.

In the testing against MCTS [24], we used the 50 determinations with 250 UCT iterations [24], and we made this consistent with the paper of Powley [24]. In addition, we used ten-thread parallelization in this method. In Figs. 6 and 7, the testing results show that our algorithm is absolutely

[8]https://ninesun.blog.csdn.net/article/details/70787814



NV-Dou vs RHCP

	Landlord	Peasants
max	0.62	0.62
min	0.47	0.45
mean	0.5535	0.5325

NV-Dou vs Random

	Landlord	Peasants
max	0.98	0.99
min	0.92	0.94
mean	0.956	0.9715

NV-Dou vs NV-Dou

	Landlord	Peasants
max	0.59	0.6
min	0.4	0.41
mean	0.4945	0.5055

NV-Dou vs Rule

	Landlord	Peasants
max	0.88	0.9
min	0.67	0.77
mean	0.8215	0.841

NV-Dou vs MCTS

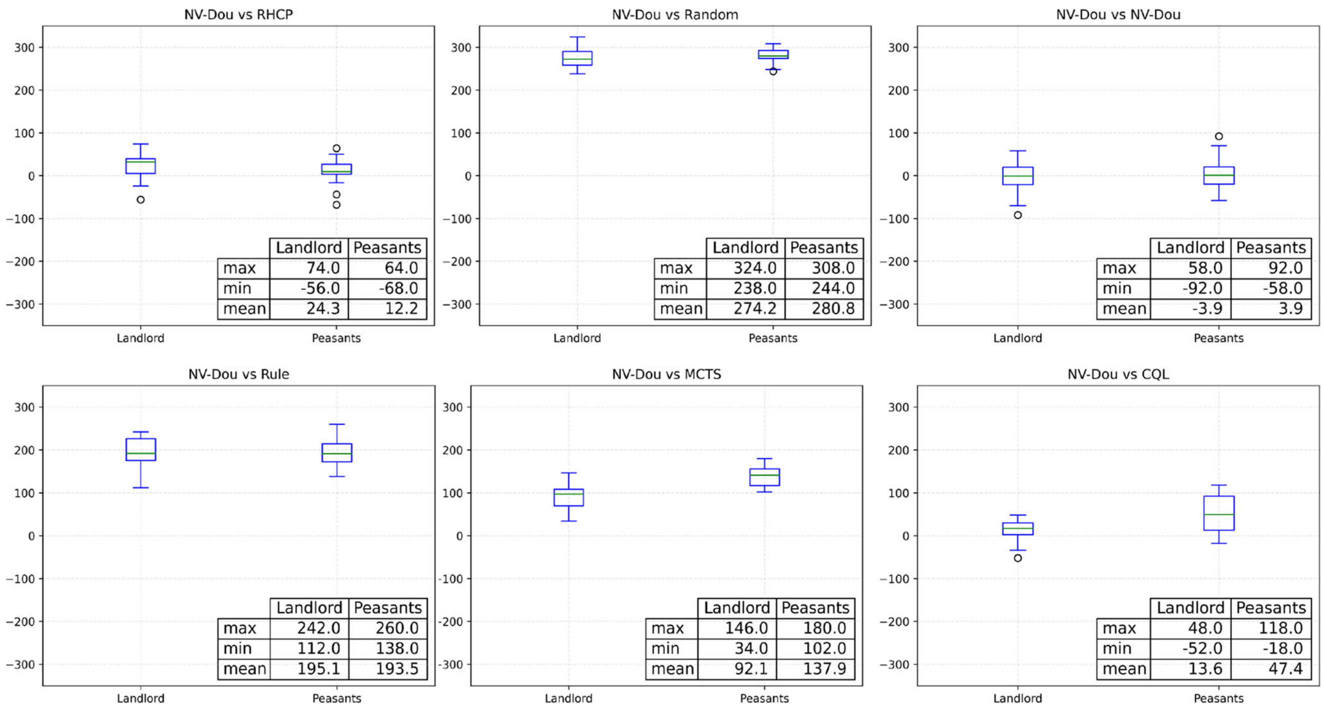
	Landlord	Peasants
max	0.78	0.82
min	0.57	0.66
mean	0.679	0.732

NV-Dou vs CQL

	Landlord	Peasants
max	0.61	0.7
min	0.42	0.49
mean	0.545	0.5795

Fig. 6 NV-Dou's winning rates against other algorithms in different roles

superior in both winning rates and the scores. Each epoch's average winning rates and scores of our algorithm as the Landlord are 0.679 and 92.1 and as Peasants, they are 0.732 and 137.9. In the process of testing, some actions are not in the opponent's rules, and we ignore the game and deal the cards again when this happens. Our method is superior to the MCTS method based on the results of testing.



NV-Dou vs RHCP

	Landlord	Peasants
max	74.0	64.0
min	-56.0	-68.0
mean	24.3	12.2

NV-Dou vs Random

	Landlord	Peasants
max	324.0	308.0
min	238.0	244.0
mean	274.2	280.8

NV-Dou vs NV-Dou

	Landlord	Peasants
max	58.0	92.0
min	-92.0	-58.0
mean	-3.9	3.9

NV-Dou vs Rule

	Landlord	Peasants
max	242.0	260.0
min	112.0	138.0
mean	195.1	193.5

NV-Dou vs MCTS

	Landlord	Peasants
max	146.0	180.0
min	34.0	102.0
mean	92.1	137.9

NV-Dou vs CQL

	Landlord	Peasants
max	48.0	118.0
min	-52.0	-18.0
mean	13.6	47.4

Fig. 7 NV-Dou's scores against other algorithms in different roles

The main idea of MCTS is to construct a game tree based on the current state and calculate the UCT of each state according to the final result. It selects the action based on the UCT. However, current computing resources and storage resources cannot perform the construction of a complete game tree of Doudizhu. Therefore, the UCT, that we obtain is incomplete and inaccurate. This is the main reason why the ability of our algorithm is superior to that of MCTS.

CQL is a method that was proposed to solve the problem of Doudizhu by reinforcement learning proposed in 2020. In the same operating environment, our models made 3433 decisions in 10 seconds, and the average decision time was 0.0029 seconds. The time that was spent in CQL was 1173 seconds for 3125 decisions, and the average decision time was 0.3754 seconds. The response speed of our algorithm is much better than that of CQL. The results of Figs. 6 and 7 show that our algorithm is also superior in terms of winning rates and scores to CQL. Each epoch's average winning rate and score for our algorithm as the Landlord are 0.545 and 13.6 and as Peasants, 0.5795 and 47.4. In addition, the process of CQL training took twenty days with equipment that included a 32-core AMD Threadripper CPU and 1080Ti GPU.

CQL selects an action based on the best combination of actions, and they are obtained from the splitting of the cards in the hand. This method reduces the influence of inferior actions on selection. Therefore, the power of the algorithm is greatly improved. However, the deep Q-learning used in CQL is not suitable for a three-person game environment such as Doudizhu. By contrast, our algorithm improves the ability to adapt to unstable environments in all respects so that it can steadily improve its training ability. This is why it is better than CQL even if it does not split the best combination of cards.

We can see that our method based agents achieve better performance playing against CQL, RHCP, and MCTS based agents. In addition, the experiment with our algorithm has advantages in terms of the use of computing resources, training time, and running time.

6 Discussion

The proposed framework integrates Q-Based policy gradient (mean actor critic) with A2C and PPO. By reducing the variance of policy gradients, our framework achieves a more stable policy optimization process and obtains better performance than that of A2C. On the other hand, the average policy of traditional NFSP is discarded to avoid the instability caused by policy fitting. In this way, every agent achieves a reliable training setting.

Since the RHCP approach traverses all possible decompositions of hand cards, it is every time consuming, even though various approaches have been proposed to reduce the number of its iterations. Similarly, since the MCTS approach utilizes UCB as the basis of making decisions and the UCB is calculated by a group of simulating game trees, it requires enormous computing resources as well. Thus, the two rule-based approaches response much slower than NV-Dou in game playing, though they need not a training.

Both CQL and DeltaDou ensure the convergence of agents' policy since they make use of deep reinforcement learning techniques. However, they require lots of computing resources to select a suitable action from a large number actions by an action filter in advance for training. By contrast, the proposed method can achieve a convergent policy without the help of the action filter in advance. Thus, it saves a lot of computing resources than that of CQL and DeltaDou.

As far as we know, most reinforcement learning methods for the problems of Go [29] and poker games [7] combined action filters. Despite the different methods of action filter, they all search high-quality actions from game trees for the policy optimization of reinforcement learning. Compared to them, the proposed method focuses on the stability of policy optimization. It enables each agent to optimize its policy from complex and diverse data by reducing the variance of policy gradient. The proposed method realizes learning from scratch because it does not need action filter during its training.

7 Conclusion and future work

In this paper, we present a Doudizhu AI (NV-Dou) and create a stable training method for a game of three players. This method is the first pure deep reinforcement learning algorithm for Doudizhu without prior knowledge or other poker game components such as MCTS, and CFR. In addition, the experiments can be performed with very limited hardware resources and training time. NV-Dou applies the policy gradient method to the framework of Doudizhu, which overcomes the instability of policy optimization by reducing the variance of policy gradients and improving the adaptability of exploration methods during policy optimization. By this means, the problem that the policies of agents cannot converge to a Nash equilibrium is avoided. Instead of filtering out good actions, NV-Dou optimizes the policies directly from trajectories. This is the main difference from other reinforcement learning algorithms like DeltaDou and CQL and is also the critical reason for the low consumption of computing resources. This shows that a pure reinforcement learning without the help of other components, such as CFR, MCTS or MCCFR, can also be adapted to the Doudizhu, which includes multiplayer.

It is worthy of investigation the proposed NV-Dou approach for other multi-agents and imperfect information poker games at first. Note further that the goal of the two Peasants is the same and we use only reward sharing in the game. Thus, we would like to find a new method to reinforce the cooperation of the two Peasants. We would like to achieve the purpose that the Peasants can adjust their actions to maximize the benefit of the group instead of just focusing on their own rewards.

Appendix

Table 2 The action categories in Doudizhu

ID	Action-name	Description
0-14	Solo	Any single card, e.g., 3 and 5 has ID 0 and 2 respectively
15-27	Pair	Two matching cards of equal rank, e.g, 33, 44.
28-40	Trio	Three individual cards of the same rank, e.g., 555, 666.
41-53	Trio with solo	Three individual cards of the same rank with a solo, e.g., 5553, 666.4
54-66	Trio with pair	Three cards of the same rank with a pair, e.g., 55533, 66644.
67-102	Chain of solo	Five or more consecutive individual cards, e.g., 56789, 3456789, 89TJQK.
103-154	Chain of pair	A sequential pairs cards and the length is at least 3, e.g., 334455, 667788, 99TTJJQQKK.
155-199	Chain of trio	Two or more consecutive trios, e.g., 333444, 333444555666.
200-237	Plane with solo	Two or more consecutive trios with each carrying an individual card, e.g., 44455567, 88899945.
238-267	Plane with pair	Twoor more consecutive trios with each carrying a pair, e.g., 4445556677, 8889994455.
268-280	Quad with solo	Four cards of the same rank with two distinct individual cards, e.g., 555533, 555534.
281-293	Quad with pair	Four cards of the same rank two sets of pair, e.g., 55553344.
294-306	Bomb	Four cards of the same rank, e.g., 5555, 6666.
307	Rocket	The red and black joker together
308	Pass	The player does not play a card.

ID is numbered from the lower rank to upper rank in each action actegory

Table 3 Game records played by NV-Dou and RHCP

No.rounds	Role	Current hand cards	Current action
1	Landlord:NV-Dou-landlord	3, 4, 6, 6, 6, 7, 8, 8, 8, 9, 9, T, T, J, Q, K, A, 2, 2, B	8883
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, J, J, K, A, A, 2	pass
	Peasant up:RHCP	4, 4, 5, 6, 7, 7, 8, T, J, Q, Q, Q, K, K, A, 2, R	QQQ5
2	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, T, T, J, Q, K, A, 2, 2, B	pass
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, J, J, K, A, A, 2	pass
	Peasant up:RHCP	4, 4, 6, 7, 7, 8, T, J, K, K, A, 2, R	77
3	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, T, T, J, Q, K, A, 2, 2, B	TT
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, J, J, K, A, A, 2	JJ
	Peasant up:RHCP	4, 4, 6, 8, T, J, K, K, A, 2, R	KK
4	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, J, Q, K, A, 2, 2, B	pass
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, K, A, A, 2	pass
	Peasant up:RHCP	4, 4, 6, 8, T, J, A, 2, R	4
5	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, J, Q, K, A, 2, 2, B	Q
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, K, A, A, 2	K
	Peasant up:RHCP	4, 6, 8, T, J, A, 2, R	2

Table 3 (continued)

No.rounds	Role	Current hand cards	Current action
6	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, J, K, A, 2, 2, B	B
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, A, A, 2	pass
	Peasant up:RHCP	4, 6, 8, T, J, A, R	R
7	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, J, K, A, 2, 2	pass
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, A, A, 2	pass
	Peasant up:RHCP	4, 6, 8, T, J, A	8
8	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, J, K, A, 2, 2	J
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, A, A, 2	2
	Peasant up:RHCP	4, 6, T, J, A	pass
9	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, K, A, 2, 2	pass
	Peasant down:RHCP	3, 3, 3, 4, 5, 5, 5, 7, 9, 9, T, A, A	5554
	Peasant up:RHCP	4, 6, T, J, A	pass
10	Landlord:NV-Dou-landlord	4, 6, 6, 6, 7, 9, 9, K, A, 2, 2	6664
	Peasant down:RHCP	3, 3, 3, 7, 9, 9, T, A, A	pass
	Peasant up:RHCP	4, 6, T, J, A	pass
11	Landlord:NV-Dou-landlord	7, 9, 9, K, A, 2, 2	K
	Peasant down:RHCP	3, 3, 3, 7, 9, 9, T, A, A	A
	Peasant up:RHCP	4, 6, T, J, A	pass
12	Landlord:NV-Dou-landlord	7, 9, 9, A, 2, 2	2
	Peasant down:RHCP	3, 3, 3, 7, 9, 9, T, A	pass
	Peasant up:RHCP	4, 6, T, J, A	pass
13	Landlord:NV-Dou-landlord	7, 9, 9, A, 2	7
	Peasant down:RHCP	3, 3, 3, 7, 9, 9, T, A	9
	Peasant up:RHCP	4, 6, T, J, A	T
14	Landlord:NV-Dou-landlord	9, 9, A, 2	A
	Peasant down:RHCP	3, 3, 3, 7, 9, T, A	pass
	Peasant up:RHCP	4, 6, J, A	pass
15	Landlord:NV-Dou-landlord	9, 9, 2	2
	Peasant down:RHCP	3, 3, 3, 7, 9, T, A	pass
	Peasant up:RHCP	4, 6, J, A	pass
16	Landlord:NV-Dou-landlord	9, 9	99

NV-Dou’s landlord model (NV-Dou-Landlord) takes the role of Landlord, and the two peasants’ algorithm are RHCP. The Current Hand Cards column represents the current handheld cards at the current round and Current Action column denotes the player’s action at this round. Finally, NV-Dou wins the game

Table 4 Game records played by RHCP and NV-Dou

No.rounds	Role	Current hand cards	Current move
1	Landlord:RHCP	3, 3, 4, 4, 6, 6, 6, 7, 8, 8, 9, T, J, Q, K, K, A, 2, 2, R	33
	Peasant down:NV-Dou-peasant-down	4, 4, 5, 5, 6, 7, 7, 9, T, T, T, J, Q, Q, A, A, A	44
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, K, K, 2, 2, B	KK
2	Landlord:RHCP	4, 4, 6, 6, 6, 7, 8, 8, 9, T, J, Q, K, K, A, 2, 2, R	22
	Peasant down:NV-Dou-peasant-down	5, 5, 6, 7, 7, 9, T, T, T, J, Q, Q, A, A, A	pass
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, 2, B	pass
3	Landlord:RHCP	4, 4, 6, 6, 6, 7, 8, 8, 9, T, J, Q, K, K, A, R	89TJQKA
	Peasant down:NV-Dou-peasant-down	5, 5, 6, 7, 7, 9, T, T, T, J, Q, Q, A, A, A	pass
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, 2, B	pass
4	Landlord:RHCP	4, 4, 6, 6, 6, 7, 8, K, R	7
	Peasant down:NV-Dou-peasant-down	5, 5, 6, 7, 7, 9, T, T, T, J, Q, Q, A, A, A	9
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, 2, B	2
5	Landlord:RHCP	4, 4, 6, 6, 6, 8, K, R	R
	Peasant down:NV-Dou-peasant-down	5, 5, 6, 7, 7, T, T, T, J, Q, Q, A, A, A	pass
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, B	pass
6	Landlord:RHCP	4, 4, 6, 6, 6, 8, K	66644
	Peasant down:NV-Dou-peasant-down	5, 5, 6, 7, 7, T, T, T, J, Q, Q, A, A, A	TTT55
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, B	pass
7	Landlord:RHCP	8, K	pass
	Peasant down:NV-Dou-peasant-down	6, 7, 7, J, Q, Q, A, A, A	AAA6
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, B	pass
8	Landlord:RHCP	8, K	pass
	Peasant down:NV-Dou-peasant-down	7, 7, J, Q, Q	77
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, B	pass
9	Landlord:RHCP	8, K	pass
	Peasant down:NV-Dou-peasant-down	J, Q, Q	QQ
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 5, 7, 8, 8, 9, 9, J, J, Q, 2, B	pass
10	Landlord:RHCP	8, K	pass
	Peasant down:NV-Dou-peasant-down	J	J

RHCP takes the role of Landlord, and the two Peasants' algorithms are Peasant down (NV-Dou-Peasant-Down) and peasant up (NV-Dou-Peasant-Up) models. The Current Hand Cards column represents the current handheld cards at the current round and Current Action column denotes the player's action at this round. Finally, Peasant Down (NV-Dou-Peasant-Down) wins the game

Table 5 Game records played by RHCP and NV-Dou

No.rounds	Role	Current hand cards	Current move
1	Landlord:RHCP	4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, T, Q, K, A, A, 2, 2	556677
	Peasant down:NV-Dou-peasant-down	3, 3, 4, 4, 5, 7, 8, 8, T, J, J, K, K, A, 2, B, R	pass
	Peasant up:NV-Dou-Peasant-Up	3, 3, 5, 7, 8, 8, 9, T, T, J, J, Q, Q, Q, K, A, 2	TTJJQQ
2	Landlord:RHCP	4, 4, 6, 6, 9, 9, 9, T, Q, K, A, A, 2, 2	pass
	Peasant down:NV-Dou-peasant-down	3, 3, 4, 4, 5, 7, 8, 8, T, J, J, K, K, A, 2, B, R	pass
	Peasant up:NV-Dou-peasant-up	3, 3, 5, 7, 8, 8, 9, Q, K, A, 2	33
3	Landlord:RHCP	4, 4, 6, 6, 9, 9, 9, T, Q, K, A, A, 2, 2	66
	Peasant down:NV-Dou-peasant-down	3, 3, 4, 4, 5, 7, 8, 8, T, J, J, K, K, A, 2, B, R	88
	Peasant up:NV-Dou-Peasant-Up	5, 7, 8, 8, 9, Q, K, A, 2	pass
4	Landlord:RHCP	4, 4, 9, 9, 9, T, Q, K, A, A, 2, 2	22
	Peasant down:NV-Dou-peasant-down	3, 3, 4, 4, 5, 7, T, J, J, K, K, A, 2, B, R	BR
	Peasant up:NV-Dou-peasant-up	5, 7, 8, 8, 9, Q, K, A, 2	pass
5	Landlord:RHCP	4, 4, 9, 9, 9, T, Q, K, A, A	pass
	Peasant down:NV-Dou-peasant-down	3, 3, 4, 4, 5, 7, T, J, J, K, K, A, 2	3
	Peasant up:NV-Dou-peasant-up	5, 7, 8, 8, 9, Q, K, A, 2	Q
6	Landlord:RHCP	4, 4, 9, 9, 9, T, Q, K, A, A	K
	Peasant down:NV-Dou-peasant-down	3, 4, 4, 5, 7, T, J, J, K, K, A, 2	A
	Peasant up:NV-Dou-peasant-up	5, 7, 8, 8, 9, K, A, 2	pass
7	Landlord:RHCP	4, 4, 9, 9, 9, T, Q, A, A	pass
	Peasant down:NV-Dou-peasant-down	3, 4, 4, 5, 7, T, J, J, K, K, 2	3
	Peasant up:NV-Dou-peasant-up	5, 7, 8, 8, 9, K, A, 2	5
8	Landlord:RHCP	4, 4, 9, 9, 9, T, Q, A, A	Q
	Peasant down:NV-Dou-peasant-down	4, 4, 5, 7, T, J, J, K, K, 2	pass
	Peasant up:NV-Dou-peasant-up	7, 8, 8, 9, K, A, 2	K
9	Landlord:RHCP	4, 4, 9, 9, 9, T, A, A	A
	Peasant down:NV-Dou-peasant-down	4, 4, 5, 7, T, J, J, K, K, 2	2
	Peasant up:NV-Dou-peasant-up	7, 8, 8, 9, A, 2	pass
10	Landlord:RHCP	4, 4, 9, 9, 9, T, A	pass
	Peasant down:NV-Dou-peasant-down	4, 4, 5, 7, T, J, J, K, K	5
	Peasant up:NV-Dou-peasant-up	7, 8, 8, 9, A, 2	9
11	Landlord:RHCP	4, 4, 9, 9, 9, T, A	T
	Peasant down:NV-Dou-peasant-down	4, 4, 7, T, J, J, K, K	pass
	Peasant up:NV-Dou-peasant-up	7, 8, 8, A, 2	A
12	Landlord:RHCP	4, 4, 9, 9, 9, A	pass
	Peasant down:NV-Dou-peasant-down	4, 4, 7, T, J, J, K, K	pass
	Peasant up:NV-Dou-peasant-up	7, 8, 8, 2	88
13	Landlord:RHCP	4, 4, 9, 9, 9, A	99
	Peasant down:NV-Dou-peasant-down	4, 4, 7, T, J, J, K, K	KK
	Peasant up:NV-Dou-Peasant-Up	7, 2	pass

Table 5 (continued)

No.rounds	Role	Current hand cards	Current move
14	Landlord:RHCP	4, 4, 9, A	pass
	Peasant down:NV-Dou-peasant-down	4, 4, 7, T, J, J	7
	Peasant up:NV-Dou-peasant-up	7, 2	2
15	Landlord:RHCP	4, 4, 9, A	pass
	Peasant down:NV-Dou-peasant-down	4, 4, T, J, J	pass
	Peasant up:NV-Dou-peasant-up	7	7

RHCP takes the role of Landlord, and the two peasants' algorithms are peasant down (NV-Dou-Peasant-Down) and peasant up (NV-Dou-Peasant-Up) models. The Current Hand Cards column represents the current handheld cards at the current round and Current Action column denotes the player's action at this round. Finally, Peasant Up(NV-Dou-Peasant-Up) wins the game

Acknowledgements We kindly thank all the anonymous reviewers whose comments improved this work to a great extent. This work is supported by the National Natural Science Foundation of China under Grants 61976065 and U1836205, Guizhou Science and Technology Foundation under Grant Qiankehejichu[2020]1Y420 and Guizhou Science Support Project (No. 2022-259).

References

1. Alvarado M, Rendón AY (2012) Nash equilibrium for collective strategic reasoning. Expert Syst Appl 39(15):12014–12025
2. Asadi K, Allen C, Roderick M, Mohamed A-R, Konidaris GD, Littman ML (2017) Mean actor critic. arXiv:1709.00503
3. Azar OH, Bar-Eli M (2011) Do soccer players play the mixed-strategy nash equilibrium? Appl Econ 43(25):3591–3601
4. Babaeizadeh M, Frosio I, Tyree S, Clemons J, Kautz J (2017) Reinforcement learning through asynchronous advantage actor-critic on a GPU. In: 5th international conference on learning representations, ICLR 2017, Toulon, April 24–26. Conference track proceedings. OpenReview.net
5. Bowling M, Burch N, Johanson M, Tammelin O (2015) Heads-up limit hold'em poker is solved. Science 347(6218):145–149
6. Brown N, Lerer A, Gross S, Sandholm T (2019) Deep counterfactual regret minimization. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th international conference on machine learning, ICM. 9–15 June 2019. Long Beach, California, vol 97 of Proceedings of machine learning research, pp 793–802. PMLR
7. Brown N, Sandholm T (2018) Superhuman ai for heads-up no-limit poker: libratus beats top professionals. Science 359(6374):418–424
8. Brown N, Sandholm T (2019) Superhuman ai for multiplayer poker. Science 365(6456):885–890
9. Cowling PI, Powley EJ, Whitehouse D (2012) Information set Monte Carlo tree search. IEEE Trans Comput Intell AI Games 4(2):120–143
10. Deng X, Wang Z, Qi L, Deng Y, Mahadevan S (2014) A belief-based evolutionarily stable strategy. J Theor Biol 361:81–86
11. Fortunato M, Azar MG, Piot B, Menick J, Hessel M, Osband I, Graves A, Mnih V, Munos R, Hassabis D, Pietquin O, Blundell C, Legg S (2018) Noisy networks for exploration. In: 6th international conference on learning representations, ICLR 2018, Vancouver, BC, April 30–May 3 2018, conference track proceedings. OpenReview.net
12. Gao Y, Li W, Khalid MNA, Iida H (2020) Quantifying attractiveness of incomplete-information multi-player game: case study using doudizhu. In: Computational science and technology. Springer, pp 301–310
13. Gao Y, Li W, Xiao Y, Khalid MNA, Iida H (2020) Nature of attractive multiplayer games: case study on China's most popular card game—doudizhu. Information 11(3):141
14. Heinrich J, Silver D (2015) Smooth UCT search in computer poker. In: Yang Q, Wooldridge MJ (eds) Proceedings of the twenty-fourth international joint conference on artificial intelligence, IJCAI, Buenos Aires, Argentina, July 25–31, 2015. AAAI Press, pp 554–560
15. Jiang Q, Li K, Du B, Chen H, Fang H (2019) Deltadou: expert-level doudizhu AI through self-play. In: Kraus S (ed) Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI 2019, Macao, China, August 10–16, 2019, pp 1265–1271. ijcai.org
16. Kawamura K, Mizukami N, Tsuruoka Y (2017) Neural fictitious self-play in imperfect information games with many players. In: Cazenave T, Winands MHM, Saffidine A (eds) Computer games - 6th workshop, CGW 2017, held in conjunction with the 26th international conference on artificial intelligence, IJCAI 2017, Melbourne, VIC, Australia, August, 20, 2017, Revised selected papers, vol 818 of Communications in computer and information science. Springer, pp 61–74
17. Knuth DE (2000) Dancing links. arXiv:0011047
18. Lanctot M, Waugh K, Zinkevich M, Bowling MH (2009) Monte Carlo sampling for regret minimization in extensive games. In: Bengio Y, Schuurmans D, Lafferty JD, Williams CKI, Culotta A (eds) Advances in neural information processing systems 22: 23rd annual conference on neural information processing systems 2009. Proceedings of a meeting held 7–10 December 2009, Vancouver, British Columbia, Canada. Curran Associates, Inc., pp 1078–1086
19. Li P, Bing L, Lam W (2018) Actor-critic based training framework for abstractive summarization. arXiv:1803.11070
20. Li S, Li S, Cao H, Meng K, Ding M (2020) Study on the strategy of playing doudizhu game based on multirole modeling. Complexity 2020
21. Li S, Wu R, Bo J (2019) Study on the play strategy of dou dizhu poker based on convolution neural network. In: 2019 IEEE international conferences on ubiquitous computing & communications (IUCC) and data science and computational intelligence (DSCI) and smart computing, networking and services (SmartCNS). IEEE, pp 702–707
22. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: Balcan M-F, Weinberger KQ (eds) Proceedings of the 33rd international conference on machine

learning, ICML 2016, New York City, NY, USA, June 19–24, 2016, vol 48 of JMLR workshop and conference proceedings, pp 1928–1937, JMLR.org
23. Moravčík M, Schmid M, Burch N, Lisỳ V, Morrill D, Bard N, Davis T, Waugh K, Johanson M, Bowling M (2017) Deepstack: expert-level artificial intelligence in heads-up no-limit poker. Science 356(6337):508–513
24. Powley EJ, Whitehouse D, Cowling PI (2011) Determinization in Monte-Carlo tree search for the card game dou di zhu. Proc Artif Intell Simul Behav:17–24
25. Schofield N, Sened I (2002) Local nash equilibrium in multiparty politics. Ann Oper Res 109(1):193–211
26. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv:1707.06347
27. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap TP, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489
28. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T et al (2018) A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362(6419):1140–1144
29. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y, Lillicrap TP, Hui F, Sifre L, van den Driessche G, Graepel T, Hassabis D (2017) Mastering the game of go without human knowledge. Nature 550(7676):354–359
30. Tan G, Wei P, He Y, Xu H, Shi X (2021) Solving the playing strategy of dou dizhu using convolutional neural network: a residual learning approach. J Comput Methods Sci Eng 21(1):3–18
31. Wang Z, Schaul T, Hessel M, van Hasselt H, Lanctot M, de Freitas N (2016) Dueling network architectures for deep reinforcement learning. In: Balcan M-F, Weinberger KQ (eds) Proceedings of the 33rd international conference on machine learning, ICML 2016, New York City, NY, USA, June 19–24, 2016, vol 48 of JMLR workshop and conference proceedings, pp 1995–2003, JMLR.org
32. Yee A, Rodríguez R, Alvarado M (2014) Analysis of strategies in American football using nash equilibrium. In: International conference on artificial intelligence: methodology, systems, and applications. Springer, pp 286–294
33. You Y, Li L, Guo B, Wang W, Lu C (2020) Combinatorial q-learning for dou di zhu. In: Proceedings of the sixteenth AAAI conference on artificial intelligence and interactive digital entertainment, AIIDE'20. AAAI Press
34. Zha D, Lai K-H, Cao Y, Huang S, Wei R, Guo J, Hu X (2019) Rlcard: a toolkit for reinforcement learning in card games. arXiv:1910.04376

Xiaomin Yu received his M.S. from Central South University, Chang Sha, China, in 2016, who is currently working toward the Ph.D. degree at Guizhou University, Gui Yang, China. His research interests include machine learning, deep learning, reinforcement learning, artificial intelligence, and pattern recognition.

Yisong Wang is currently a Professor at Guizhou University. He received B.S., M.S. and Ph.D. degrees from Guizhou University in 1998, 2004 and 2007, respectively. Dr. Wang currently serves as an Associate Editor of the Annals of Mathematics and Artificial Intelligence (2018-). His main research interests include knowledge representation and reasoning, nonmonotonic reasoning and answer set programming in particular, inductive logic programming and their applications.

Jin Qin received his B.S. and M.S. degrees from Guizhou University, Guiyang, China, in 2000 and 2003, respectively, and Ph.D. degree from the University of Science and Technology, Beijing, China, in 2012. His research interests include computational intelligence and machine learning.

Panfeng Chen received the B.S. degree in science from the China Three Gorges University, in 2007 and the M.S. degree in science from Central China Normal University in 2011, respectively. He recieved his Ph.D. degree from Guizhou University in 2021, Guiyang, Guizhou, China. His main interesting includes knowledge presentation and reasoning, especially knowledge graph related problems.