DouZero+: Improving DouDizhu AI by Opponent Modeling and Coach-guided Learning

1st Youpeng Zhao
University of Science and Technology of China
Hefei, China
zyp123@mail.ustc.edu.cn

2nd Jian Zhao
University of Science and Technology of China
Hefei, China
zj140@mail.ustc.edu.cn

3rd Xunhan Hu
University of Science and Technology of China
Hefei, China
cathyhxh@mail.ustc.edu.cn

4th Wengang Zhou
Institute of Artificial Intelligence
Hefei Comprehensive Nation Science Center;
University of Science and Technology of China
Hefei, China
zhwg@ustc.edu.cn

5th Houqiang Li
Institute of Artificial Intelligence
Hefei Comprehensive Nation Science Center;
University of Science and Technology of China
Hefei, China
lihq@ustc.edu.cn

***Abstract*—Recent years have witnessed the great breakthrough of deep reinforcement learning (DRL) in various perfect and imperfect information games. Among these games, DouDizhu, a popular card game in China, is very challenging due to the imperfect information, large state and action space as well as elements of collaboration. Recently, a DouDizhu AI system called DouZero has been proposed. Trained using traditional Monte Carlo method with deep neural networks and self-play procedure without the abstraction of human prior knowledge, DouZero has achieved the best performance among all the existing DouDizhu AI programs. In this work, we propose to enhance DouZero by introducing opponent modeling into DouZero. Besides, we propose a novel coach network to further boost the performance of DouZero and accelerate its training process. With the integration of the above two techniques into DouZero, our DouDizhu AI system achieves better performance and ranks top in the Botzone leaderboard among more than 400 AI agents, including DouZero.**

***Index Terms*—DouDizhu, Reinforcement learning, Monte-Carl Method, Opponent Modeling, Coach Network**

I. Introduction

During the development of machine learning, games usually serve as an important testbed as they are good abstraction of many real-world problems, and more objective compared to environments specially designed for testing AI since games are developed for humans. In recent years, significant progress has been made in perfect-information games such as Go [1]–[3], Shogi (Japanese chess) [4] and even fighting game [5]. The current research efforts are turning to more challenging imperfect information games (IIG) where agents may cooperate or compete with each other under a partially observable environment. Encouraging achievements have been made from two-player games, such as Texas Hold'em [6]–[8] to multi-player games, including multi-player Texas Hold'em [9], StarCraft [10], DOTA [11] and Japanese Mahjong [12].

This work was supported in part by the National Natural Science Foundation of China under Contract 61836011 and in part by the Youth Innovation Promotion Association CAS under Grant 2018497.

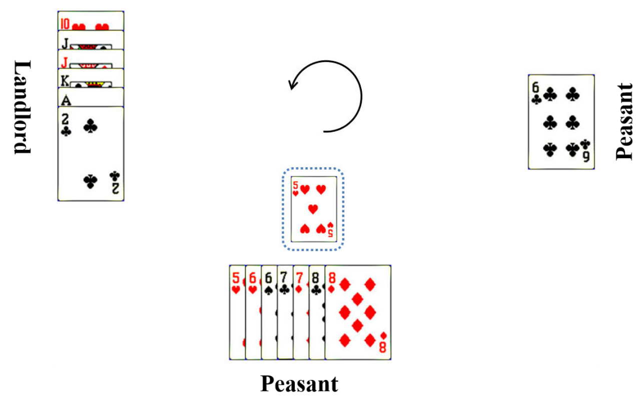


Fig. 1: A case example about cooperation in DouDizhu. If the Peasants learn to cooperate with each other, current player should play small Solo to let the teammate to win the game.

In this work, we are dedicated to designing an AI program for DouDizhu, *a.k.a,* Fighting the Landlord, which is the most popular card game in China. The two characteristics of this game make it challenging for developing AI programs. First, this game involves both cooperation and competition simultaneously in a partially observable environment. To be specific, the two Peasant agents play as a team to fight against the Landlord agent. Taking Figure 1 as an example, it demonstrates a typical case where the Peasant at the bottom can play a small Solo card for the winning of his partner. This property makes classical algorithms for Poker games, such as Counterfactual Regret Minimization (CFR) [13] and its variants not suitable in such a complex three-player setting. Second, the state and action space in DouDizhu is large and complex due to the combination of cards and the complex rules compared to other card games. There are thousands of

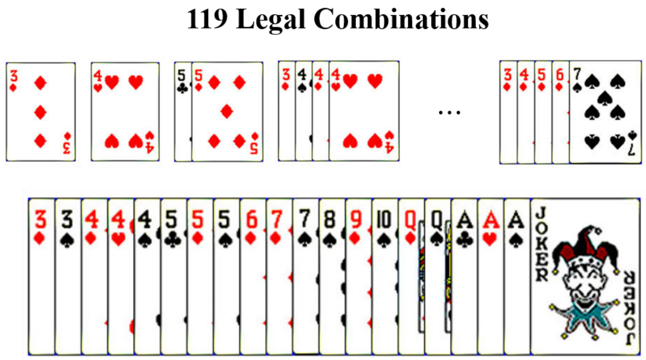


Fig. 2: A hand and its corresponding legal moves.

possible combinations of cards where different subsets of these combinations are legal to different hands. Figure 2 exhibits an example of a hand that has 119 legal moves such as Solo, Pair, Trio, Chain of Solo and so on. The performance of Deep Q-Learning (DQN) [14] will be greatly affected due to the overestimating issue in large action space [15] while policy gradient methods such as A3C [16] fail to leverage the action features, limiting the capability of generalizing over unseen actions. In this way, previous work has shown that DQN and A3C have a poor performance in DouDizhu, as AI programs trained with these algorithms can not beat simple rule-based agents even after long time of training [17].

Despite the challenges mentioned above, some achievements have been made in building DouDizhu AI. To deal with the large action space in DouDizhu, Combinatorial Q-Network (CQN) [17] decouples the actions into decomposition selection and final move selection. However, this processing technique relies on human prior knowledge and costs a lot of time, which limits its performance. In fact, CQN does not have preponderance over the heuristic rule-based model. DeltaDou [18] is the first bot that manages to achieve top human-level performance. It makes use of an AlphaZero-like algorithm, which combines neural networks with Fictitious Play Monte Carlo Tree Search (FPMCTS), and an inference algorithm in a self-play procedure. However, DeltaDou requires to pre-train a kicker network depending on heuristic rules to reduce the action space size, which may affect its strength if the output of the kicker network is not optimal. What's more, the inference and search are so computationally expensive that the training of DeltaDou takes a lot of time. Recently, DouZero [19] has attracted considerable attention due to its outstanding performance in this complex game. It utilizes self-play deep reinforcement learning without the abstraction of state/action space and human knowledge. It combines classical Monte-Carlo methods [20] with deep neutral networks to handle the large state and action space, which opens another door for such complex and large-scale games.

In this work, we improve DouZero by introducing opponent modeling and coach-guided learning. Opponent modeling aims to determine a likely probability distribution for the opponents' hidden cards, which is motivated by the fact that human players will try to predict the opponents' cards to help them determine the policy. Due to the complexity of DouDizhu, a lot of actions may be appropriate when making the decision. In this case, analyzing the opponents' cards will be of great importance because grasping this information helps the bot choose the optimal move. On the other hand, we propose coach-guided learning to fasten the training of the AI. Due to the large information space in this game, the training of the AI program for DouDizhu costs a lot of time. Considering the fact that the outcome of DouDizhu depends heavily on the initial cards of three players, quite a few games are of little value for learning. To this end, we design a novel coach network to pick matched openings so that the models can learn from more valuable data without wasting time to play valueless games, thus accelerating the training process. Through integrating these techniques into DouZero, our DouDizhu AI program achieves a better performance than the original DouZero and ranks the first on the Botzone [21]–[23] leaderboard among more than four hundred agents, including DouZero.

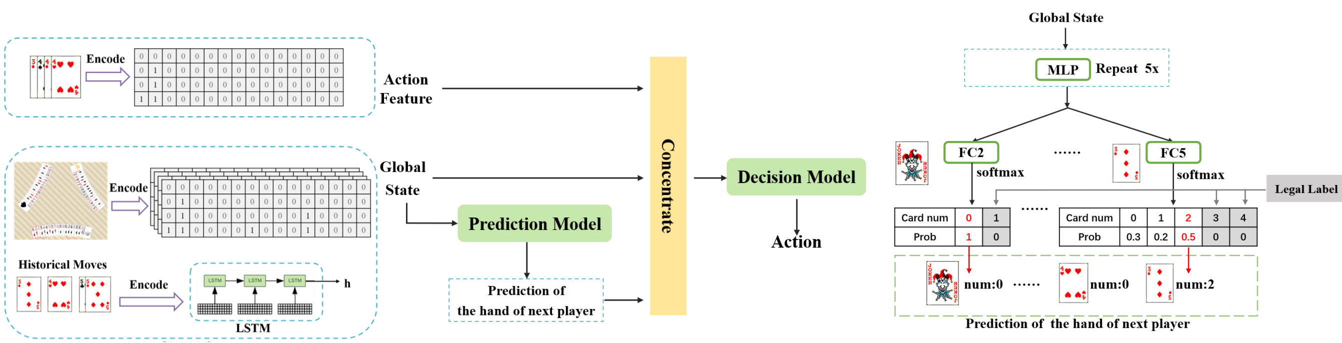
II. Related Work

In this section, we briefly introduce the application of reinforcement learning in imperfect-information games as well as the works about opponent modeling.

A. Reinforcement Learning for Imperfect-Information Games

Recent years have witnessed the successful application of reinforcement learning in some complex imperfect-information games. For instance, there are lots of works about reinforcement learning for poker games [6], [24]. Different from Counterfactual Regret Minimization (CFR) [13] that relies on game-tree traversals, RL grasps knowledge through interactions with the environment so that it is also suitable for large-scale games. OpenAI, DeepMind and Tencent have utilized this technique to build their game AI in DOTA [11], StarCraft [10] and Honor of Kings [25], respectively, and acquired amazing achievements, proving the effectiveness of reinforcement learning in imperfect-information games. More recently, there are some research efforts that combine reinforcement learning with search and have shown its effectiveness in poker games such as heads-up no-limit Texas Hold'em poker and DouDizhu [18], [26].

However, due to the complexity of DouDizhu, classical reinforcement learning methods such as DQN [14] and A3C [16] exhibit poor performance in this game as discussed above. Even improved methods such as Combinatorial Q-Network (CQN) [17] and DelaDou [18] have limited performance as they need some prior knowledge. To this end, DouZero [19] utilizes Monte-Carlo methods [20] and manages to defeat all DouDizhu AI programs by now. We note that this technique is also adopted in some other game AIs, such as a modern board game, Kingdomino, and a kind of new chess, Tibetan Jiuqi [27], [28]. But unlike these environments, DouDizhu is more complex due to its characteristics. The amazing performance of DouZero reveals the good results of Monte-Carlo methods



(a) The overall framework

(b) The details about prediction model

Fig. 3: An overview of the framework that combines opponent modeling with DouZero and the details about the prediction model. The prediction model takes the state information as input and outputs the probability of the number of every card in the hand of the next agent. The decision model is trained using deep Monte-Carlo algorithm like DouZero. The prediction result is concatenated with the state features as well as action features and all these information is input to decision model to decide which action to take. As for the prediction model, it can be viewed as a multi-head classifier, which consists of a layer of LSTM to encode historical moves, five shared layers of MLP and multi-head FC layers to output the probability. The LSTM layer is contained in the extraction of state information, so it is not shown in the Figures. We can extract "legal label" from the state information, which represents how many cards of each kind the next player has at most. Figure (b) shows an example where the next player has at most two Card 3. This information can be used as a mask to prevent the prediction model from giving a wrong prediction that next player has more than two Card 3 in his hand.

in such large-scale complex card games, providing new insight into future research on such problems.

B. Opponent Modeling for Games

In human practice, gaining an abstract description of the opponent will give the player a clear advantage in games, especially imperfect-information games. As a result, opponent modeling has attracted substantial attention in game AI. For example, Southey *et al.* [29] put forward a Bayesian probabilistic model for poker games which infers a posterior over opponent strategies and makes an appropriate response to that distribution. In another complex imperfect-information game, Mahjong, an AI bot is designed based on opponent modeling and Monte Carlo simulation [30]. In this work, the opponent models are trained with expert game records and the bot decides the move using the prediction results and Monte-Carlo simulation.

Recently, inspired by the success of reinforcement learning, many researchers combine opponent modeling with reinforcement learning and have made much progress. In combination with deep Q-learning, opponent modeling achieves superior performance over DQN and its variants in a simulated soccer game and popular trivia game [31]. Knegt *et al.* [32] introduces the opponent modeling technique into an arcade video game using reinforcement learning, which helps the agent predict opponents' actions and significantly improves the agent's performance. In addition, opponent modeling can be adopted in multi-agent reinforcement learning problems where RL agents are designed to consider the learning of other agents in the environment when updating their own policies [33]. Another promising solution is to mimic human players by combining opponent models used by expert players and reinforcement learning [34]. All the above works demonstrate that combining opponent modeling with reinforcement learning is beneficial to achieve performance gain in multi-agent imperfect-information games, which also inspires this work.

III. Preliminary

In this section, we first discuss the main algorithm of DouZero, *i.e.* Deep Monte Carlo (DMC), which adopts deep neural networks for function approximation in Monte Carlo method. Then, we briefly describe the details of DouZero system.

As a key technique in reinforcement learning, Monte Carlo (MC) method learns value functions and optimal policies from experience, namely, sampling sequences of states, actions and rewards from actual or simulated interactions with the environment [20]. This technique is designed for episodic tasks, where experience can be divided into episodes that eventually terminate, and it updates the value estimation and policy only when an episode is completed. To be specific, after each episode, the observed returns are used for policy evaluation and then the policy can be improved at the visited states in the episode. The optimization procedure of a policy π in MC methods is intuitively described as follows:

1) Interact with the environment for an episode using π.
2) For each state-action pair (s, a) visited in the episode, calculate and update $Q(s, a)$ with the average return.
3) For each state s in the episode, update the policy: $\pi(s) \leftarrow argmax_{\mathbf{a} \in A} Q(s, \mathbf{a})$.

When putting MC methods into practice, epsilon-greedy policy can be used to balance between exploration and exploitation in Step 1. Also, deep neural networks can be naturally adopted in the above procedure, leading to Deep Monte-Carlo (DMC). In this way, the Q-table $Q(s,a)$ can be replaced by neural networks which can be optimized with mean-square-error (MSE) loss in Step 2.

As DouDizhu is a typical episodic task, MC is naturally suitable for this problem. What's more, DMC requires a large amount of experience for training while it's easy to generate data efficiently in parallel, which can also alleviate the issue of variance. In addition, adopting DMC in DouDizhu has some clear advantages compared to other reinforcement learning algorithms, such as policy gradient methods and deep Q-learning, which can be referred to in DouZero [19]. Owing to the advantages that DMC has in DouDizhu, DouZero adopts this algorithm and achieves an outstanding performance.

In the implementation of DouZero system, it makes use of a self-play procedure, where the actors play games to generate samples and the learner updates the network using these data. The input of the network consists of state features and action features. The state feature represents the information that is known to the player, while the action feature encodes the set of all legal moves in current state. Specifically, the action in action features is encoded with a one-hot 4×15 card matrix. For the state features, they contain matrices that represent information of card combinations and some one-hot vectors that represent information about number in this game such as the number of cards of other players, and the number of bombs played so far. For the architecture, historical moves are encoded by a layer of LSTM and this information is concatenated with other features. Also, there are six layers of MLP with a hidden size of 512 to output Q values, which serves as the main body of models that makes decision.

Besides, to improve the training efficiency, the system implement DMC in a parallel way with multiple actors and one learner. Three global networks are maintained in learner process for the three players. These networks are updated to approximate the target values based on samples produced by actors. And each actor keeps three local networks which are synchronized with the global ones periodically. The communication between the learner and actors is realized with three shared experience buffers. In this way, the system can be trained in an effective self-play procedure.

IV. Method

In this section, we introduce opponent modeling and coach network in our design and describe how they are applied.

A. Opponent Modeling

Opponent modeling studies how to construct models to predict about various properties of the modeled agents, *e.g.* actions, goals and so on. Classic methods such as policy reconstruction [35] and plan recognition [36] tend to develop parametric models for agent behaviours. These methods tend to decouple the interactions between the modeled agent and others to simplify the modeling process, which may introduce bias when there exists coupling between agent interactions.

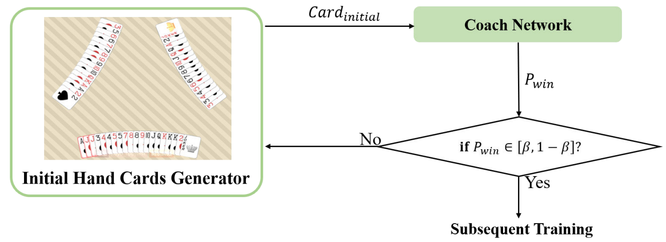


Fig. 4: The overview of the framework that utilizes coach network. In this figure, we use the $Card_{initial}$, P_{win} and β to represent generated initial hand cards, the predicted probability of winning for Landlord and the threshold value, respectively. The coach network is composed of one embedding layer and several fully connected layers and the model takes $Card_{initial}$ as input and outputs P_{win}. If P_{win} is in the range defined, which is decided by β, the game with such $Card_{initial}$ will be carried on and generates samples for training. Otherwise, another initial hand cards will be generated.

In this way, executing opponent modeling when concurrently training all the agents in a self-play procedure is more natural [37] and suitable to the training procedure of DouDizhu AI system. What's more, concurrent learning helps opponent modeling adapt to different levels of the agent as it has witnessed the evolution of the agent's skills during training.

When adopting opponent modeling in DouDizhu, we predict the hand of the next player as knowing this information is enough to grasp the hands of both other players in DouDizhu. As for the implementation of opponent modeling, we can naturally take advantage of deep neural networks to make predictions. For clear description, we call the part of opponent modeling as "prediction model" and the part that makes decisions as "decision model". Following the practice of DouZero that trains three models for the three players in the game, we also train three prediction models for opponent modeling. The prediction model can be viewed as a multi-head classifier and outputs the probability of the number of every kind of card in the hand of the next agent. To be specific, it has to predict how many Card 3, how many Card 4, *etc*, the next player has in his hand. Since the environment of DouDizhu is easy to realize, we can acquire the true hand of the next player and use it as labels to train the prediction model. What's more, taking Card 3 as an example, we can also know how many card 3 of one kind the next player has at most, which can be calculated by the agent's own hand and how many Card 3 has been played. We call this information "legal label" and this information can be utilized as a mask to prevent prediction model from making impossible prediction.

As for other details, we make use of the same state features as DouZero and the architecture of prediction models is also similar to DouZero except for the final layer which works as a multi-head classifier where each head outputs the prediction of one kind of card. This model is trained using cross-entropy loss function. As for the decision model, we concatenate the prediction results of prediction model as well as original state features for state input of decision models. To sum up, the overview of the framework that combines opponent modeling with DouZero is shown in Figure 3.

B. Coach-guided Learning

During the training of DouDizhu AI system, we discover that the training process costs a lot of time. To this end, we propose a method to help the agent master the skills faster. In this work, our DouDizhu AI system does not have a bidding phase as the bidding network in DouZero is trained with supervised learning. In other words, the initial hand cards of the three players are fixed at the beginning of the game. However, the players aim to empty his own hand cards before others in DouDizhu, making the quality of the initial hand cards have a great impact on the result of this game. If one player gets a very strong hand at the beginning, he can win easily as long as he does not make serious mistakes. In this way, such initial cards are of little value for learning as they can hardly help the agent learn new knowledge. On the other hand, if one player always plays matches where the initial hand cards are relatively balanced, he can learn some skills faster and better as he will lose and receive a negative reward if he makes any unsuitable decision. In the setting of DouZero, we uncover that the initial cards of the three players are generated randomly so that quite a few samples may be not matched in strength. However, the actors still have to play the game using these initial cards that are heavily unbalanced, which also takes much time. If we only allow the actors to generate samples that are based on balanced initial hand cards, the agent can learn faster and form policies that can deal with such situation.

Based on the above discussion, we propose a "coach network" to identify whether the initial hand cards are balanced in strength. It takes the initial hand cards of the three players as input and outputs the predicted probability of winning for the Landlord in one game, which we call P_{win}. Then we can set a threshold, which is represented with β, to filter out the games whose P_{win} is too small or too big, as is shown in Figure 4. In this case, there is no need for the actors to play with these initial hand cards, thus setting aside time to carry on more valuable matches. It is noted that the threshold is set to 0 at first and increases through the training process so that the agent can learn necessary skills that are enough to deal with unbalanced matches.

The input of coach network is the vectors of initial hand cards for Landlord and Peasants, whose dimensions are 20 and 17, respectively. For the architecture of coach network, it consists of an embedding layer to process the input vectors and three layers of fully connected layers to extract representations and make predictions. As our DouDizhu AI system is trained in a self-play manner, the coach network is also concurrently trained with the decision models. The results of self-play games can be used as labels for training the model. What's more, we only need to train one coach network for prediction as this module has nothing to do with the positions in DouDizhu. In other words, our coach network only works at the beginning of one game to pick suitable initial data and does not influence the subsequent processes. Therefore such idea can also be transferred into the development of other similar game AIs and benefits the training.

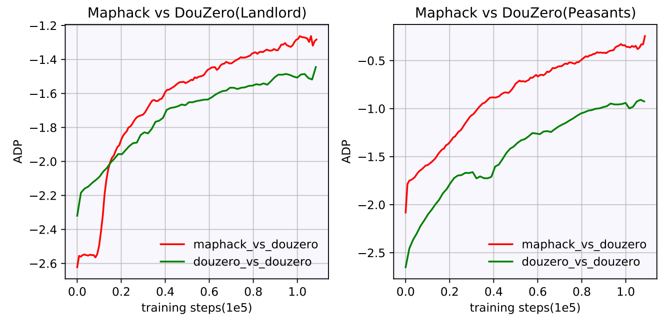


Fig. 5: ADP of "maphack" models, which can see the hand cards of the next player, and DouZero models. Both these models are tested with DouZero baseline that is fully trained with ADP. "Landlord" means that the models play as Landlord against Peasants of DouZero baseline and the same goes for the reverse. One training step means one update of networks in learner.

V. Experiment

In this section, we conduct experiments to demonstrate the effectiveness of the improvement that we introduce to DouZero. To be specific, we first evaluate the performance of opponent modeling and coach network, respectively, and then combine them together. All experiments are conducted on a server with 4 Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz and GeForce RTX 2080Ti GPU. Our codes are available at https://github.com/submit-paper/Doudizhu.

A. Experiment Settings

In order to evaluate the performance of the model, we follow what DouZero [19]and Deltadou do [18] and launch tournaments that include both Landlord and Peasants. To be specific, for two competing algorithms A and B, they will first play as Landlord and Peasants, respectively, for a given deck. Then the sides have to be switched, *i.e.* A plays the role of Peasants and B takes Landlord position, and they play the same deck again. To show the performance of the model in the training process, we execute the test for 10000 games every 30 minutes. As our DouDizhu AI is based on DouZero, we just compare the performance between them. We make use of the open-source models of DouZero as the opponent. To demonstrate the improvement, we also train an original DouZero to intuitively exhibit the performance difference. As for the evaluation metrics, we follow DouZero and use Average Difference in Points (ADP). This metric indicates the average difference of points scored per game between algorithm A and B. Specifically, if A wins a game, it will be rewarded with 1 and its opponent will get a reward of -1. In addition, every time one player plays a bomb in the game, the point they get will double.

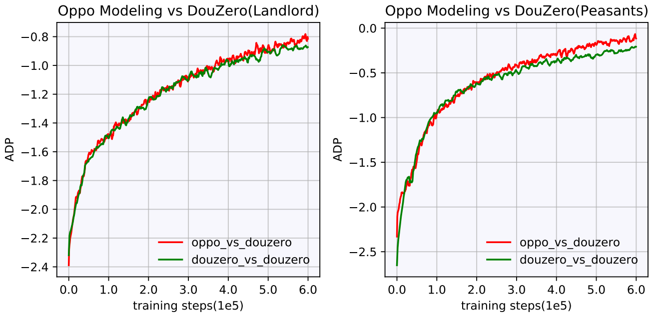


Fig. 6: ADP of models, which combine opponent modeling and DouZero, and DouZero models. Both these models are tested with DouZero baseline that is fully trained with ADP. "Landlord" means that the models play as Landlord against Peasants of DouZero baseline and the same goes for the reverse. One training step means one update of networks in learner.

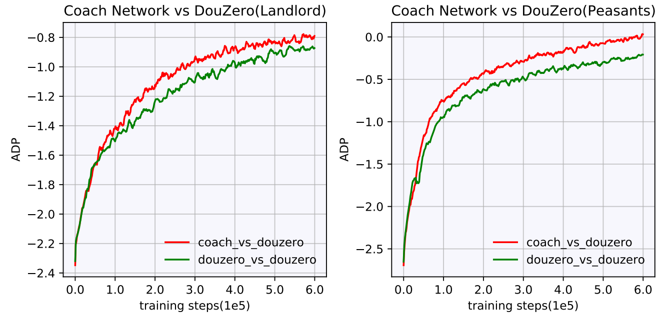


Fig. 7: ADP of models, which combine coach network with DouZero, and DouZero models. Both these models are tested with DouZero baseline that is fully trained with ADP. "Landlord" means that the models play as Landlord against Peasants of DouZero baseline and the same goes for the reverse. One training step means one update of networks in learner.

Our implementation is based on DouZero and training schedules such as the number of actors and training hyper-parameters are kept the same as the default ones. DouZero provides a fully trained model which is trained using ADP. We train our AI system and an original DouZero system with ADP as objective and compare their performance with the baseline model. For consistency, we use the metric of ADP when evaluating the performance of the models. The difference between the performances of these two AI systems can demonstrate the effectiveness of our methods.

B. Evaluation on Opponent Modeling

In this part, we demonstrate the effectiveness of introducing opponent modeling to DouDizhu. As the state features utilized by DouZero contain all the information that can be known, the information about the hand cards of the next player is included implicitly while the idea of opponent modeling is essentially making such information explicit. In order to investigate whether such an idea helps the agents learn better, we firstly make a pre-experiment where we add the hand cards of the next player into state features directly, whose result is shown in Figure 5. It can be observed that adding the hand cards of the next player into state features indeed boosts the performances of the agents, especially for Peasants. We assume that the obvious improvement of Peasants is attributed to the fact that knowing the hand cards of the next player helps Peasants not only choose cards that the Landlord can't afford but also cooperate with the teammate better. Whereas for the Landlord, knowing the hand cards of next player indeed helps to make decisions, but if the hand is weak, even having such information can not help a lot. To sum up, the result of the pre-experiment illustrate that introducing explicit representations of the next player's hand cards improves the performance of DouDizhu AI.

After verifying the validity of our idea, we concurrently train the prediction models as well as the decision models as is discussed in Section IV-A and the result is shown in Figure 6. It reveals that introducing opponent modeling to DouZero mainly improves the performance of models of Peasants, which is corresponding to the analysis above. Although the models perform worse than DouZero at first because the network has to take more features as input and has more neurons, which will slow down learning, they manage to grasp more knowledge after enough training and achieve a performance better than DouZero.

C. Evaluation on Coach Network

Apart from the experiments above, we also conduct experiments to show how coach network performs in DouDizhu game. The training procedure is discussed in Section IV-B. The threshold β starts from 0 and increases by 0.05 after every 20000 training steps and its upper limit is set to be 0.3. The result of the experiment is shown in Figure 7 and the significant improvement proves the effectiveness of this method. It can be observed that the improvement of Peasants is also greater than that of Landlord. Considering that Peasants have an advantage in this game due to cooperation, this phenomenon is acceptable as they can learn more skills in balanced games. Besides, even coach-guided learning strategy only controls the initial state of the game, the improvement it can bring is significant. This fact reveals that the luck factor plays an important role in such kind of imperfect-information games. To this end, our method can be migrated into other environments, helping game AI achieve better performance.

What's more, we also show some cases about the prediction of our coach network from games on Botzone platform, which is illustrated in Table I. In case 1, it can be observed that the Landlord is allocated with a very strong hand, which consists of most cards of high rank and cards of low rank that can compose other combinations so that the Landlord can win the game easily. As for case 2, even Landlord has a bomb in his hand, the hand cards of Peasants are also very good. What's worse, the Landlord also has quite a few cards of low rank

	Landlord	Landlord_down	Landlord_up	Prediction of P_{win} for Landlord	Actual result(Landlord)
Case1	3455677789JQKAAAA22R	334569TTTJJQQQKK2	344566788899TJK2B	0.9932	Win
Case2	45667788889TTTKKA22B	334567TJJJQQQQK22	33445567999JKAAAR	0.1726	Lose
Case3	3455556677799JJQKAAB	3467889TTQKKK222R	33446889TTJJQQAA2	0.5843	Lose

TABLE I: Case study to show the effect of coach network. It predicts the winning probability of Landlord based on the initial hand cards of the three players. We pick some cases from games from Botzone to show the predicted results of coach network and also show the actual result from the view of the Landlord. To be mentioned, T means 10, J means Jack, Q means Queen, K means King, A means Ace, B means Black Joker, and R means Red Joker.

that are difficult to play out. In case 3, the initial hand cards are relatively balanced. However, the Peasant win the game finally, indicating the importance of cooperation. This example illustrates that the balanced samples can indeed help the agents learn cautious policy and cooperation better, thus proving the correctness of our idea.

D. Combination of Two Methods

From the above discussion, it is known that both our improvements can help enhance the performance of DouZero. The result of combining these two methods is shown in Figure 8. As the improvement of coach network is more obvious than opponent modeling, to intuitively demonstrate whether the combination of these two techniques brings further improvement, we also add the result of just using coach network in the figure. It can be observed the performance is a little worse than just using coach network at first, which is consistent with the discussion of just introducing opponent modeling. To be mentioned, when the performance of the models reaches a certain level, achieving a little improvement is very difficult so the progress that combining the two methods makes is not that apparent. However, further improvement still proves the effectiveness of combination of the two methods.

To comprehensively compare the performance of our DouDizhu AI, we upload our final model to BotZone [22], an online platform with DouDizhu competition. There more than 20 games apart from DouDizhu supported by the platform, including Go, Mahjong, Chess and so on. More than 3500 users on this platform upload their bot programs to compete with others. Botzone maintains a leaderboard for each game, which ranks all the bots in the leaderboard by their Elo rating scores. The evaluation of DouDizhu in Botzone Elo system is also similar to the form of tournament introduced above. Although Elo rating is generally considered as a stable measurement of relative strength, the characteristic of high variance of this game still makes the Elo ranking suffer from fluidity. What's more, due to the limit of server resources, Elo rating games are not scheduled very frequently so that it may take a lot of time to achieve a stable ranking. However, keeping a high ranking can still prove the strength of one AI system. Even if DouZero has obvious superiority over other DouDizhu AI systems trained by reinforcement learning, it has ranked about 20th so far on Botzone leaderboard as most bots are realized by strong heuristic rules. Nonetheless, our DouDizhu AI has always ranked top five, even ranked first for

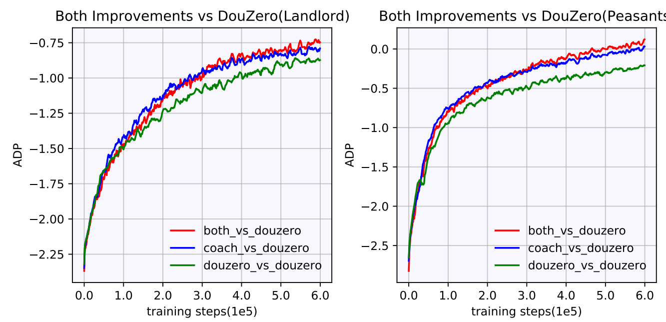


Fig. 8: ADP of models, which combine both improvements with DouZero, and DouZero models. Both these models are tested with DouZero baseline that is fully trained with ADP. "Landlord" means that the models play as Landlord against Peasants of DouZero baseline and the same goes for the reverse. For comparison, the result of models improved by coach network is also included.

several months, proving the effectiveness of the improvements that we make.

VI. Conclusion and Future Work

In this work, we put forward some improvements to the state-of-the-art DouDizhu AI program, DouZero. Inspired by the human player's prediction about opponents' hand cards in practice, we introduce opponent modeling. Based on the nature of high variance of this game, we originally propose a coach network to pick valuable samples to accelerate the training. The outstanding performance of our AI on the Botzone platform proves the effectiveness of our improvement.

Although our DouDizhu AI performs well after adopting these techniques, there is still room for improvement. First, to show the effect of our improvement, we do not change the architectures of neural networks in DouZero unless necessary. We plan to try other neural networks such as ResNet [38]. Second, we find that there are still some cases where the model can not make good decisions. We hope to combine search with our AI to enhance the performance as search proves to be effective in research about game AI [39], [40]. Finally, we will investigate how to improve the sample efficiency with experiment replay [41] as it still costs a lot of time even utilizing our coach network. In addition, we will also try to transfer our methods to other games for stronger game AIs.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] T. Cazenave, "Improving model and search for computer go," in *IEEE Conference on Games (CoG)*, 2021, pp. 1–8.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[5] D.-W. Kim, S. Park, and S.-i. Yang, "Mastering fighting game using deep reinforcement learning with self-play," in *IEEE Conference on Games (CoG)*, 2020, pp. 576–583.

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

[7] N. Brown and T. Sandholm, "Superhuman ai for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, 2018.

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

[9] N. Brown and T. Sandholm, "Superhuman ai for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[10] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

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

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

[13] T. W. Neller and M. Lanctot, "An introduction to counterfactual regret minimization," in *Educational Advances in Artificial Intelligence (EAAI)*, vol. 11, 2013.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[15] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor, "Learn what not to learn: Action elimination with deep reinforcement learning," *arXiv preprint arXiv:1809.02121*, 2018.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[17] Y. You, L. Li, B. Guo, W. Wang, and C. Lu, "Combinatorial q-learning for dou di zhu," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, 2020, pp. 301–307.

[18] Q. Jiang, K. Li, B. Du, H. Chen, and H. Fang, "Deltadou: Expert-level doudizhu ai through self-play." in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2019, pp. 1265–1271.

[19] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, and J. Liu, "Douzero: Mastering doudizhu with self-play deep reinforcement learning," *arXiv preprint arXiv:2106.06135*, 2021.

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

[21] H. Zhou, Y. Zhou, H. Zhang, H. Huang, and W. Li, "Botzone: A competitive and interactive platform for game ai education," in *Proceedings of the ACM Turing 50th Celebration Conference-China*, 2017, pp. 1–5.

[22] H. Zhou, H. Zhang, Y. Zhou, X. Wang, and W. Li, "Botzone: an online multi-agent competitive platform for ai education," in *ACM Conference on Innovation and Technology in Computer Science Education*, 2018, pp. 33–38.

[23] H. Zhang, G. Gao, W. Li, C. Zhong, W. Yu, and C. Wang, "Botzone: A game playing system for artificial intelligence education," in *International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, 2012, p. 1.

[24] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," *arXiv preprint arXiv:1711.00832*, 2017.

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

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

[27] M. Gedda, M. Z. Lagerkvist, and M. Butler, "Monte carlo methods for the game kingdomino," in *IEEE Conference on Computational Intelligence and Games (CIG)*, 2018, pp. 1–8.

[28] J. Zhou, "Design and application of tibetan long chess using monte carlo algorithm and artificial intelligence," in *Journal of Physics: Conference Series*, vol. 1952, no. 4, 2021, p. 042104.

[29] F. Southey, M. P. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and C. Rayner, "Bayes' bluff: Opponent modelling in poker," *arXiv preprint arXiv:1207.1411*, 2012.

[30] N. Mizukami and Y. Tsuruoka, "Building a computer mahjong player based on monte carlo simulation and opponent models," in *IEEE Conference on Computational Intelligence and Games (CIG)*, 2015, pp. 275–283.

[31] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé III, "Opponent modeling in deep reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1804–1813.

[32] S. J. Knegt, M. M. Drugan, and M. A. Wiering, "Opponent modelling in the game of tron using reinforcement learning." in *International Conference on Agents and Artificial Intelligence (ICAART)*, 2018, pp. 29–40.

[33] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," *arXiv preprint arXiv:1709.04326*, 2017.

[34] L. F. Teófilo, N. Passos, L. P. Reis, and H. L. Cardoso, "Adapting strategies to opponent models in incomplete information games: a reinforcement learning approach for poker," in *International Conference on Autonomous and Intelligent Systems*, 2012, pp. 220–227.

[35] D. Carmel and S. Markovitch, "Model-based learning of interaction strategies in multi-agent systems," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 10, no. 3, pp. 309–332, 1998.

[36] M. Fagan and P. Cunningham, "Case-based plan recognition in computer games," in *International Conference on Case-Based Reasoning*, 2003, pp. 161–170.

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

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[39] B. Bouzy, A. Rimbaud, and V. Ventos, "Recursive monte carlo search for bridge card play," in *IEEE Conference on Games (CoG)*, 2020, pp. 229–236.

[40] S. Ariyurek, A. Betin-Can, and E. Surer, "Enhancing the monte carlo tree search algorithm for video game testing," in *IEEE Conference on Games (CoG)*, 2020, pp. 25–32.

[41] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv preprint arXiv:1712.01275*, 2017.