Combining Tree Search and Action Prediction for State-of-the-Art Performance in DouDiZhu

Yunsheng Zhang[1*], **Dong Yan**[1†], **Bei Shi**[2], **Haobo Fu**[3], **Qiang Fu**[2], **Hang Su**[1], **Jun Zhu**[1] and **Ning Chen**[4]

[1]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Lab, Tsinghua University
[2]Tencent AI Lab
[3]Tencent
[4]Independent Researcher
ys-zhang18@mails.tsinghua.edu.cn, sproblvem@gmail.com, {beishi, haobofu, leonfu}@tencent.com, {suhangss, dcszj, ningchen}@mail.tsinghua.edu.cn

Abstract

AlphaZero has achieved superhuman performance on various perfect-information games, such as chess, shogi and Go. However, directly applying AlphaZero to imperfect-information games (IIG) is infeasible, due to the fact that traditional MCTS methods cannot handle missing information of other players. Meanwhile, there have been several extensions of MCTS for IIGs, by implicitly or explicitly sampling a state of other players. But, due to the inability to handle private and public information well, the performance of these methods is not satisfactory. In this paper, we extend AlphaZero to multiplayer IIGs by developing a new MCTS method, Action-Prediction MCTS (AP-MCTS). In contrast to traditional MCTS extensions for IIGs, AP-MCTS first builds the search tree based on public information, adopts the policy-value network to generalize between hidden states, and finally predicts other players' actions directly. This design bypasses the inefficiency of sampling and the difficulty of predicting the state of other players. We conduct extensive experiments on the popular 3-player poker game DouDiZhu to evaluate the performance of AP-MCTS combined with the framework AlphaZero. When playing against experienced human players, AP-MCTS achieved a 65.65% winning rate, which is almost twice the human's winning rate. When comparing with state-of-the-art DouDiZhu AIs, the Elo rating of AP-MCTS is 50 to 200 higher than them. The ablation study shows that accurate action prediction is the key to AP-MCTS winning.

1 Introduction

Compared to the much progress on solving perfect-information games [Silver *et al.*, 2018], imperfect information games (IIGs) are more challenging and relatively under-explored. The representative progress on IIGs is on solving poker games with counterfactual regret minimization (CFR) [Zinkevich *et al.*, 2008]. For example, DeepStack [Moravčík *et al.*, 2017] and Libratus [Brown and Sandholm, 2018] have beaten top professional players in Heads-Up No-Limit Texas Hold'em using CFR. However, these methods require a complete expansion of the whole game tree to calculate counterfactual values, thereby heavily relying on a domain-specific abstraction [Brown and Sandholm, 2019] to reduce the game scale [Sandholm, 2015]. Existing improvements such as MCCFR [Lanctot *et al.*, 2009] can reduce but not fundamentally eliminate the heavy dependence on domain knowledge. When applying these methods to other scenarios, or even other poker games, users need to have a deep insight on both the target scenario and the CFR algorithm in order to design an appropriate abstraction method.

In contrast, Monte Carlo Tree Search (MCTS) without any abstraction or prior knowledge has proven its success on large-scale zero-sum games. Especially, AlphaZero [Silver *et al.*, 2017] achieves superhuman performance on chess, shogi and Go based on MCTS, which scales more efficiently. Compared with CFR, MCTS grows the tree asymmetrically via the selection of leaf nodes to concentrate on the more promising subtrees. Moreover, in the simulation step, the method only needs to complete one path from the leaf node to the terminal node instead of the full expansion as in CFR. It contributes to an efficient traversal of the game tree. However, the direct application of AlphaZero for IIGs is infeasible because traditional MCTS methods cannot handle missing information of other players. The hidden information makes constructing a search tree problematic.

Some attempts has been made to extend MCTS to handle hidden information in games. For instance, information set MCTS (ISMCTS) [Cowling *et al.*, 2012] grows a tree over the information sets for each player instead of constructing a separate tree for each determinization. However, ISMCTS suffers from the information leaking problem [Furtak and Buro, 2013]. Fictitious Play MCTS (FPMCTS) [Jiang *et al.*, 2019] explicitly estimates a posterior distribution in a Bayesian manner, samples a concrete state from the distri-

*This work was partially done when Yunsheng Zhang was an intern at Tencent AI Lab.

†Contact Author

bution, and then predicts actions via policy-value networks according to the sampled state. However, the sampled state is usually inaccurate because of the vast state space and the unobservable information of opponents. Another drawback of FPMCTS is that it has to generate a policy network from a heuristic AI's self-play results to bootstrap the reinforcement learning procedure.

In this paper, we propose *Action-Prediction MCTS* (AP-MCTS), a variant of MCTS that extends AlphaZero to multiplayer IIGs. Specifically, to handle the hidden information of opponents, we sample their actions from the policy-value network when the node in MCTS represents the opponents. Unlike existing methods based on MCTS, we exploit the data from self-play to train the policy-value network and predict the actions directly. Without knowing the underlying state, it has an advantage that our framework avoids the inaccurate sampling of states from opponents. We test our benchmark on DouDiZhu (which has 3-players), the most popular poker game in China. Since playing a Nash equilibrium in games with three or more players may not be wise [Brown and Sandholm, 2019], finding or even approximating a Nash equilibrium is hard [Rubinstein, 2018]. Thus, our goal in this paper is to build a DouDiZhu AI, which has a high Elo rating[1]. We invited experienced human players to play hundreds of games with our AI, and AP-MCTS defeats them with 65.65% winning rate. We also validate our method by playing our AI against previous SOTA DouDiZhu AIs and achieve a significant advantage.

Compared with previous state-of-the-art baseline methods [You *et al.*, 2019; Jiang *et al.*, 2019], our framework uses less training time (reduced from several months to one week) while achieving better performance. Furthermore, we do not need any heuristic AI or rules to help the agent for bootstrapping. Overall, our contributions are as follows:

- We propose a general training framework for large scale IIGs, which can handle hidden information and long sequences in the game.
- We train an expert-level DouDiZhu AI, which outperforms all other state-of-the-art methods largely. We open source the code and neural network weights to facilitate following research works. [2]

2 Background and Related Work

2.1 DouDiZhu

DouDiZhu, also known as China Competitive Poker, is a trendy poker game in China, and has been used as a challenging benchmark of imperfect-information games [Powley *et al.*, 2011]. DouDiZhu is a 3-player zero-sum game, where one landlord plays against two peasants. It uses a standard deck of 54 cards, including two Jokers. At the beginning of the game, each player receives 17 cards and then bids for the landlord position. The highest bidder becomes the landlord and gets three extra cards. The other two players become peasants and form a team to play against the landlord.

Cards are played in rounds, starting from the landlord. At each round, the current player plays one category of cards (e.g., solo, pair, trio, etc.), and then every player in order plays the same category of cards or pass. The round ends when two players choose a pass, the other player wins the round and starts the next round. The goal of the game is to play all cards in hand. If the landlord is the first of the three to play all the cards, he/she wins. Otherwise, the peasants win.

The average length of the decision sequence is about 30, which is much longer than that of the Texas Hold'em decision sequence. Furthermore, at the beginning of the game, the number of the possible situations for opponent's hand cards is about 2×10^6, far more than the 10^3 of Texas Hold'em.

2.2 Monte Carlo Tree Search and AlphaZero

Monte Carlo Tree Search (MCTS) [Coulom, 2006] is widely used in game playing, it grows the search tree asymmetrically by concentrating on those states that are more likely to arrive. Many researchers try to extend MCTS to imperfect-information games (IIGs). For instance, [Powley *et al.*, 2011] introduce a way to extend MCTS to IIGs by determinization, which is a well-known method first introduced by [Whitehouse *et al.*, 2011]. It transforms the imperfect information into the perfect information by determining the opponent's state and runs the original MCTS algorithm at the perfect information state. Determinization has two critical problems, namely non-locality and strategy fusion [Whitehouse *et al.*, 2011]. Strategy fusion means that an agent cannot make different decisions from different states in the same information set, while non-locality means that the different determinization of the same information set may derive very distinct probabilities distributions. ISMCTS [Cowling *et al.*, 2012] aims to resolve the strategy fusion problem while still suffers from the information leaking problem, as pointed out by [Furtak and Buro, 2013]. FP-MCTS [Jiang *et al.*, 2019] derives its strategy based on a sampled state, which is usually inaccurate because of the enormous state space and the unobservable information of opponents.

AlphaZero achieves dramatic performance improvement in perfect-information games by combining MCTS and policy-value networks [Silver *et al.*, 2018]. Conventionally, the policy-value network of AlphaZero takes the real state as input, which can only be speculated but not directly observed in IIGs. Previous works either abandon the planning part completely [You *et al.*, 2019] or adopt a Bayesian method to infer the real state [Jiang *et al.*, 2019], but the impact of the inability to know the real state on performance has not been well resolved.

3 Method

An essential feature of imperfect-information games is that different roles in the game have different information perspectives. Taking DouDiZhu as an example, an audience can only see public actions (e.g., play a card); a player can see his/her hands additionally, while the dealer knows the hands of all players. For convenience, we name these perspectives as public information [Johanson *et al.*, 2011], private information and perfect information, correspondingly.

[1] Elo rating is a widely used method to calculate the relative skill levels of players, named after its creator Arpad Elo.

[2] https://github.com/1310183534/DouDiZhu.

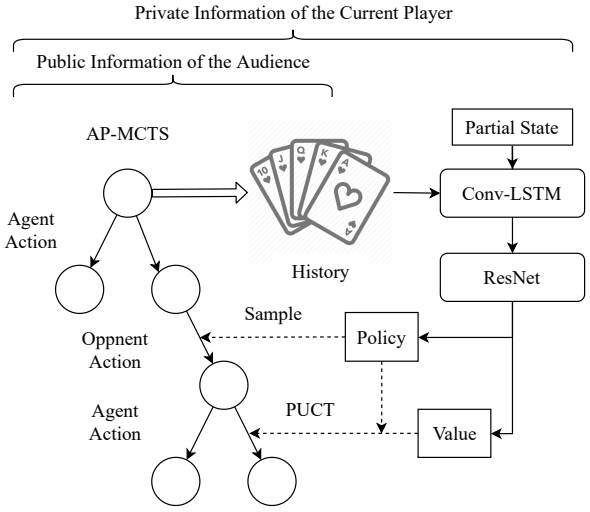

Figure 1: The architecture of Action-Prediction MCTS. The node of the search tree denotes the public information, e.g., the play history, and the partial state denotes the hand cards of the current player. On each decision point, they are fed together to the neural network to generate the policy and value of current situation. Note that the current player does not know the opponent's hand cards. Thus, the partial state always encodes current player's hand cards, no matter who's turn to play. Different decision points are differed by identity channels.

We now formally present the Action-Prediction MCTS (AP-MCTS). AP-MCTS builds the search tree on the audience's public information while feeding the accompanying policy-value network with the private information of the current player. In AP-MCTS, there is only one search tree and an accompanying policy-value network, as shown in Fig. 1. The fundamental difference between AP-MCTS and previous methods is that AP-MCTS chose the action based on the play history instead of the opponents' state. AP-MCTS adopts the LSTM to extract the play history information and adopts the convolution layer to extract the hand cards information. Such information is all that the player can access while playing. It is enough to decide which action to take without knowing the opponents' state. Reasoning the opponents' state is costly and unnecessary. Actually, human players also make decisions directly based on the current play history, rather than guessing all possible states and extrapolating from those states. AP-MCTS uses the neural network's powerful generalization ability to learn the mapping between play history and the outcome. The role of different players is indicated by the identity channels, and the hand cards of the current player are encoded in other channels (details are described in Tab 1). This architecture enables sharing common experiences while retaining the motivation of exploration without assuming the knowledge of any information unknown in the real game.

3.1 Action-Prediction MCTS for Imperfect Information Games

Based on this architecture, AP-MCTS predicts actions instead of states to make decisions with imperfect information. The working procedure of AP-MCTS is shown in Alg. 1.

On each node of the current player, AP-MCTS selects the action according to the Predictor Upper Confidence Tree (PUCT) algorithm [Silver *et al.*, 2017], while on each node of the opponent, AP-MCTS directly samples the action from the policy distribution given by the policy-value network. A tricky part is how to ensure that the sampled opponent action is legal. Intuitively, opponents may have different legal actions depending on some of state variables that are privately known only to them. But as long as we notice the fact that all cards remain in the standard deck are legal actions, only except those ones that are already played or held by the current player. Since the play history and hand cards are encoded in the input of the accompany policy-value network, AP-MCTS is able to filter out illegal moves.

The policy-value network is denoted by f_θ, where θ is the neural network parameters. The input of f_θ is the partial state s (hand cards of the current player) and play history h. The output of f_θ can be expressed as $f_\theta = (P_\theta(s,h), V_\theta(s,h))$. During each simulation, for the AP-MCTS agent α, it only uses the searching algorithm to improve its own policy and regards other players' policy as fixed. Specifically, if it is α's turn, we select action a using a a variant of the PUCT algorithm:

$$a = \arg\max_a Q(u,a) + c_{\text{puct}} P_\theta(u,a) \frac{\sqrt{N(u)}}{1+N(u,a)}, \quad (1)$$

where u is the node representing (s,h), $P_\theta(u,a) = \Pr(\text{a}|\text{s},\text{h})$ estimates the probability of choosing action a for given partial state s and decision sequence h, $N(u,a)$ is the

Algorithm 1 Action-Prediction Monte-Carlo Tree Search

Require: partial state s, history h, neural network parameter θ, game simulator $\mathcal{G}$, game reward $R(R(s,h) \in \{-1,0,1\}$ means lose, draw, win)
Function: SIMULATE(u,s,h)
  if IS-TERMINAL(s,h) **then**
    return $r \sim R(s,h)$
  end if
  if $u \notin T$ **then**
    Add the node u to the tree T
    return $r \sim V_\theta(s,h)$
  end if
  if IS-AGENT'S-TURN(s,h) **then**
    $a =$ PUCT-SELECT(u,s,h)
  else
    $a \sim P_\theta(s,h)$
  end if
  $(s',h') \leftarrow \mathcal{G}(s,h,a)$
  Find the node u' representing (s',h') in the tree T.
  $r \leftarrow$ SIMULATE(u',s',h')
  $N(u) \leftarrow N(u)+1$
  $W(u) \leftarrow W(u)+r$
  $Q(u) \leftarrow \frac{W(u)}{N(u)}$
  return r
Function: AP-MCTS(s,h)
  Create the node u_{root} represents (s,h)
  Create the tree T with root u_{root}
  while within computational budget **do**
    SIMULATE(u_{root},s,h)
  end while
  return $\arg\max_a N(u_{\text{root}},a)$

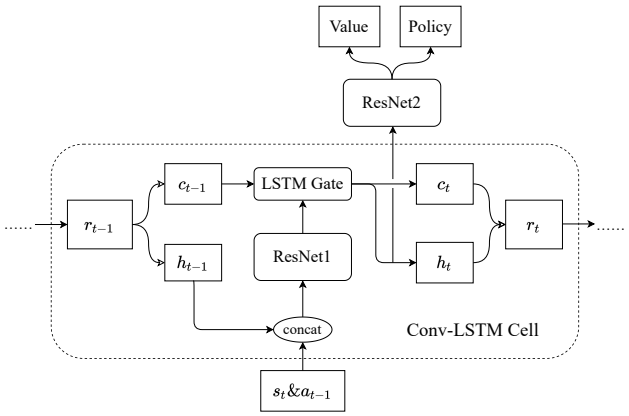

Figure 2: Neural Network Architecture

visit count for action a in node u, $Q(u, a)$ is the mean action-value.c_{puct} is a constant determining the level of exploration. If it is not α's turn, it will sample action a from move distribution estimated by the policy network directly $a \sim P_\theta(s, h)$. This shows that if we can fit the other players' policy distribution $P_\theta(s, h)$ well, we do not need to use any determinization, and the search algorithm can still work.

As shown in Fig 2, the policy-value neural network consists of two modules, Conv-LSTM and ResNet [He *et al.*, 2015]. The Conv-LSTM module encodes historical actions, and the ResNet module takes h_t as input to calculate the value v and the policy p for the agent. The ResNet1 contains eight residual blocks with a width of 160, The ResNet2 contains sixteen residual blocks with a width of 256. Short-term memory h_t and long-term memory c_t each have 128 channels. Note that, there is only one instance of the neural network. All self-play data that played by the landlord and the peasant are fed to it to achieve the maximum sampling efficiency.

3.2 Input and Output Encoding of Neural Network

There are 15 ranks of cards in DouDiZhu, of which {*A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K*} and two Jokers. Each rank except the Joker has four cards. So we adopt a 4×15 matrix to encode the card information. The input of the neural network (s_t, a_{t-1}) is represented as 14 channels, and each channel is a 4×15 matrix. The meaning of each channel is described in Table 1. Note that, DouDiZhu is played in turn. The position of a player determines the play order. It is represented by one-hot encoding in channel 1-3, 4-6 and 7-9.

The neural network outputs a scalar v and a vector p, where $v \in [-1, 1]$ is a scalar evaluation , and p is the probability distribution of selecting each action. The tricky part of the output encoding is that DouDiZhu allows players to combine different card types then play them together. This combination will make the action space huge. So we decompose those combination as individual actions, and represent them by a vector of length 356. Details are shown in Table 2.

4 Experiment

In this section, we conduct extensive experiments to evaluate our AP-MCTS method. First, we played hundreds of games against human players, and AP-MCTS achieved a winning rate of over 65%. Second, we compare AP-MCTS with other state-of-the-art methods on the DouDiZhu game. The result shows that AP-MCTS outperforms all those methods. Finally, we conduct an ablation study to illustrate how each module of AP-MCTS affects the final performance.

Channel	Description
0	The hand cards of the player controlled by the agent. If the type $i(0 \leq i < 15)$ cards in the player's hand cards is more than $j(0 \leq j < 4)$, then the element of j-th row and i-th column is 1, otherwise 0.
1-3	Position of the player controlled by the agent.
4-6	Position of the current player.
7-9	Position of the landlord player.
10	last action(if it is not kicker). Similar to channel 0.
11	last action(if it is kicker). Similar to channel 0.
12	The three extra cards that the landlord received.
13	Bid action. The entire plane is filled with b, where $b \in \{0, 1, 2, 3\}$ indicates the bid.

Table 1: Input Encoding

Action	Description	Action	Description
0	pass	108-143	solo chain
1-14	bomb	144-195	pair chain
15-29	solo	196-240	trio chain
30-42	pair	241-308	trio chain with kicker
43-55	trio	309-323	solo kicker
56-81	trio with kicker	324-336	pair kicker
82-107	four with kicker	337-351	extra cards
		352-355	bid

Table 2: Output Encoding

4.1 Experiment Settings and Hyper-Parameters

In our experiment, we evaluated the relative strength of our agents and baselines by Elo Rating System with Bayesian Logistic Regression [Coulom, 2008]. Elo Rating System is a widely used rating system that is used to rank a group of players. Players with higher Elo rating have a higher probability of defeating players with lower Elo rating. We estimate the probability of one landlord player a_0 defeating two peasant players $a1$, $a2$ by a logistic function $P(a_0 \text{ defeat } a_1, a_2)$, described below:

$$P(a_0 \text{ defeat } a_1, a_2) = \frac{1}{1 + 10^{c_{\textbf{elo}} \cdot ((e(a_1)+e(a_2))/2 - e(a_0))}} \quad (2)$$

where $e(a)$ denotes the Elo rating of player a. We set the $c_{\textbf{elo}}$ to the standard constant $\frac{1}{400}$. This means that if an agent's Elo is 100 higher than another agent's Elo, then it has a 64% probability of winning in games.

We use 96 CPUs and 24 GPUs for generating self-play games, and 4 GPUs for training the neural network. The

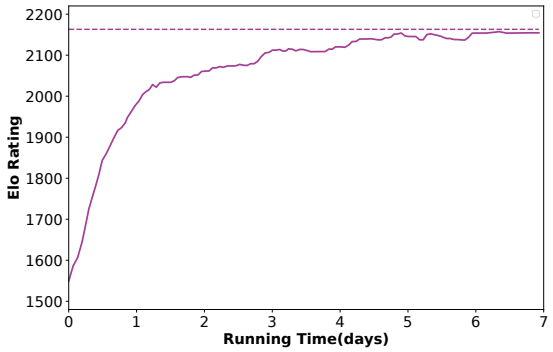

Figure 3: Growth of Elo rating over time. The horizontal line represents the Elo score of the strongest agent through reinforcement learning, which is 2154.

training process and the self-playing process are performed simultaneously. About 200,000 self-play games can be generated in one day. The training data is the latest 300,000 self-play games data. Neural network parameters are optimized by stochastic gradient descent with momentum[Rumelhart *et al.*, 1986]. We set the initial learning rate as 0.01 and mini-batch size as 1024, then adjust the learning rate to 0.001 on the fourth day. We use truncated backpropagation through time[Jaeger, 2002] to calculate the gradient of LSTM part and the longest back propagation length is 16 time-steps.

Our self-playing reinforcement learning started with random neural network parameters and ran for seven days. The player α_{θ_t} controlled by the latest neural network parameters θ_t are used to generate self-play games. α_{θ_t} selects each action using 800 AP-MCTS simulations during inference. Every four hours, we save the current version of our agent in an agent pool. The growth of Elo rating over time is shown in Figure 3. We use Bayesian Logistic Regression to calculate the Elo rating of each agent, and the game is played by three agents randomly selected from the pool. We can see that our method converges on the sixth day.

4.2 Play with Experienced Human Players

In order to test the ability of our agent against humans, we invited 15 DouDiZhu enthusiasts for testing. Three of these players are in the top 5% of a DouDiZhu online platform. Without sponsorship, these are the highest level of human players we could find. Our agent plays a total of 294 games with them. Each game consists of one human player and two agents. The results are shown in Table 3.

Since the winning or losing of the DouDiZhu game is closely related to the hand cards, which is completely random, a total of 65.65% of the winning rate can show that our agent is much stronger than ordinary experienced human players.

	Number of Matches	Winning Rate
Human as Landlord	173	36.42 %
Human as Peasant	121	31.40 %
Total	294	34.35 %

Table 3: AI against experienced human player

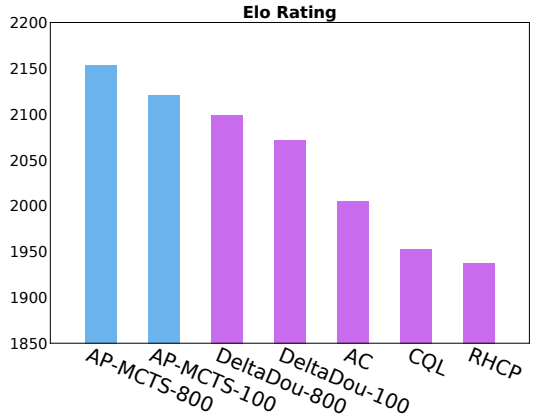

Figure 4: Comparison between our method, Actor-Critic, RHCP, CQL and DeltaDou. The numbers 800 and 100 indicate the times of simulations for MCTS during the period of self-play in each step.

4.3 Comparison with State of The Art AI for Solving DouDiZhu

In this section, we compare the performance of our framework with the previous state-of-the-art solution on the entire game by the Elo Rating System.

As far as we know, there are mainly two branches of methods applied to the game of DouDiZhu. One is based on pure reinforcement learning. You et al. [2019] propose a method of Combinational Q-Learning for DouDiZhu. Considering numerous card combinations at each time step leading to a huge number of actions, they employ a two-stage network to reduce action space and also leverage order invariant max-pooling operations to extract relationships between primitive actions. Their experiments results demonstrate that Combinational Q-Learning outperforms other standard reinforcement learning baselines such as A2C [Mnih *et al.*, 2016] and DQN [Mnih *et al.*, 2013]. The final agents can be competitive to human. The other branch uses variants of MCTS to construct DouDiZhu AI. Whitehouse et al. [Whitehouse *et al.*, 2011] applied the technique of determinization coupled with MCTS into DouDiZhu. DeltaDou [Jiang *et al.*, 2019] exploits FPMCTS and clams that it dominates all other Doudizhu AI's they are aware of. In their experiment, DeltaDou shows a comparable strength against human experts.

We use the following methods as our baselines:

- **Actor-Critic** [Konda and Tsitsiklis, 2000]. We use the standard actor-critic method as one of our baselines.
- **RHCP**. It is an open-source heuristics-based algorithm available online [3]. It decides the hand to play purely by some handcrafted hand value estimation function.
- **Combinational Q-learning (CQL)** [You *et al.*, 2019]. As mentioned in Section 2, it is one of the state-of-the-art methods which is based on pure reinforcement learning.
- **DeltaDou** [Jiang *et al.*, 2019]. It is a DouDiZhu expert-level AI that uses deep learning, Bayesian method and

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

Fictitious Play MCTS(FP-MCTS) as mentioned in Section 2.

We have tried to train DeltaDou through self-play reinforcement learning from random neural network parameters, but the performance of the agent does not improve. It is because their methods need a hand-coded heuristic algorithm to bootstrap the training process in the original paper [Jiang *et al.*, 2019]. Therefore, we used a well-trained agent to generate a dataset containing 300,000 games data. For a fair comparison, we train the neural networks used in AP-MCTS and DeltaDou on this generated dataset and calculate their Elo rating, respectively.

The results compared with all the baseline methods are shown in Figure 4. It demonstrates that our method outperforms all the baseline methods with a substantial margin. Especially, our method AP-MCTS outperforms DeltaDou with an improvement of 55 Elo scores over the range of times of simulations. Compared with CQL, the state-of-the-art method based on pure reinforcement learning, our method can still achieve an improvement of 201 Elo scores. That is, our method has a winning rate of about 57.9% against Deltadou and a 76.1% winning rate against CQL.

4.4 Ablation Study

Effect of Action Prediction

In order to show that it is necessary to predict the opponent's action in AP-MCTS, we replace this module with randomly selecting one of all possible opponent's actions, and named this method as Random Action-Prediction MCTS (RAP-MCTS). We compare the ability of agents using AP-MCTS and RAP-MCTS. The Elo ratings are shown in Table 4. All agents use the same neural network parameters.

Table 4 demonstrates that without the accurate prediction of the actions from opponents, the more number of simulation times, the strength of the agent will become worse. It proves the effectiveness of our action prediction module.

Both our AP-MCTS method and FP-MCTS predict actions of opponents during the inference of the game tree in MCTS. The difference is that we predict actions of opponents directly via the policy network while FP-MCTS use a Bayesian way to predict actions over the space of the state. We compare the performance of predicting actions of opponents between our method and the Bayesian method used in DeltaDou. The training dataset and test dataset each contains 300,000 and 3,000 self-play games data. The neural network architecture of the Bayesian method used in the experiment is the same with our neural network except for the Conv-LSTM part. The

Method	Simulation Times	Elo Rating
AP-MCTS	800	2148
AP-MCTS	100	2121
Policy Network	-	2094
RAP-MCTS	100	2089
RAP-MCTS	800	2067

Table 4: Comparison between the agents using AP-MCTS and RAP-MCTS

	MSE	Cross-Entropy	Accuracy
Conv-LSTM	0.612	0.912	67.5%
Bayes	0.686	1.081	63.3%

Table 5: Prediction comparison between Conv-LSTM used in our method and Bayesian method used in DeltaDou.

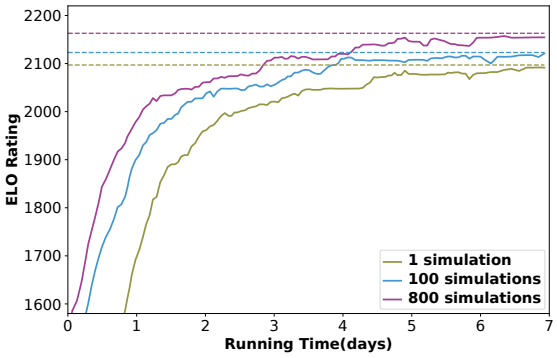

Figure 5: Elo rating with different number of simulations. Note that the neural network at the same time is same for all the three settings. The only difference is the simulation times of AP-MCTS for the inference of the next motion.

Bayesian method use 1000 times of determinization. The results are shown in Table 5. We conclude that our methods can predict actions of opponents better than DeltaDou.

Effect of Simulation Times

To investigate the effectiveness of MCTS, we use the same neural network but set the number of simulation times during inference as 1, 100, and 800, respectively. More simulation times indicate that MCTS can evaluate the next move in a more accurate way. In particular, using 1 simulation in AP-MCTS means directly acting according to the prediction of policy head.

The final results are shown in Figure 5. It can be seen from the results that at the beginning of self-play reinforcement learning when the model parameters are approximately random, the agent using 800 simulations has a great advantage over the agent using 1 simulation. At the end of self-play reinforcement learning, the agent using 800 simulations still has 55.8% probability of defeating the agent using 100 simulations, and 59% probability of defeating the agent using 1 simulation.

5 Conclusion and Future Work

In this paper, we propose AP-MCTS and a neural network structure that can efficiently process historical information, and extend the AlphaZero algorithm to large-scale imperfect information games. The experiments show that our framework is able to beat previous state-of-the-art DouDiZhu solutions. In addition, we have open-sourced our code and model parameters.

In future work, we will extend our framework to more complicated games, e.g. Stratego. Small experiments demonstrate the universality of our framework, the only change of porting it to Stratego is input and output encoding.

References

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

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

[Coulom, 2006] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *5th International Conference on Computer and Games*, 2006.

[Coulom, 2008] Rémi Coulom. Whole-history rating: A bayesian rating system for players of time-varying strength. In H. Jaap van den Herik, Xinhe Xu, Zongmin Ma, and Mark H. M. Winands, editors, *Computers and Games*, pages 113–124, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Cowling *et al.*, 2012] Peter I Cowling, Edward J Powley, and Daniel Whitehouse. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143, 2012.

[Furtak and Buro, 2013] Timothy Furtak and Michael Buro. Recursive monte carlo search for imperfect information games. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8. IEEE, 2013.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[Jaeger, 2002] Herbert Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. *GMD-Forschungszentrum Informationstechnik, 2002.*, 5, 01 2002.

[Jiang *et al.*, 2019] Qiqi Jiang, Kuangzheng Li, Boyao Du, Hao Chen, and Hai Fang. Deltadou: expert-level doudizhu ai through self-play. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1265–1271. AAAI Press, 2019.

[Johanson *et al.*, 2011] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *Twenty-Second International Joint Conference on Artificial Intelligence*, volume 11, pages 258–265, 2011.

[Konda and Tsitsiklis, 2000] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[Lanctot *et al.*, 2009] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pages 1078–1086, 2009.

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

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Moravčík *et al.*, 2017] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[Powley *et al.*, 2011] Edward J Powley, Daniel Whitehouse, and Peter I Cowling. Determinization in monte-carlo tree search for the card game dou di zhu. *Proc. Artif. Intell. Simul. Behav*, pages 17–24, 2011.

[Rubinstein, 2018] Aviad Rubinstein. Inapproximability of nash equilibrium. *SIAM Journal on Computing*, 47(3):917–959, 2018.

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[Sandholm, 2015] Tuomas Sandholm. Abstraction for solving large incomplete-information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[Whitehouse *et al.*, 2011] Daniel Whitehouse, Edward J Powley, and Peter I Cowling. Determinization and information set monte carlo tree search for the card game dou di zhu. In *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pages 87–94. IEEE, 2011.

[You *et al.*, 2019] Yang You, Liangwei Li, Baisong Guo, Weiming Wang, and Cewu Lu. Combinational q-learning for dou di zhu. *arXiv preprint arXiv:1901.08925*, 2019.

[Zinkevich *et al.*, 2008] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.