

Learning to Play *Koi-Koi* Hanafuda Card Games with Transformers

Sanghai Guan, *Member, IEEE*, Jingjing Wang, *Senior Member, IEEE*, Ruijie Zhu, *Member, IEEE*, Junhui Qian, *Member, IEEE*, and Zhongxiang Wei, *Member, IEEE*

Abstract—Card games are regarded as an idealized model for many real-world problems for their rich hidden information and strategic decision-making process. It provides a fertile environment for artificial intelligence (AI), especially reinforcement learning algorithms. With the boom of deep neural networks, increasing breakthroughs have been made in this challenging domain. *Koi-Koi* is a traditional two-player imperfect-information playing card game. However, due to its unique deck and complex rules, related researches are mostly based on handcrafted features and the custom network architecture. In this paper, we design a more general AI framework, relying a Transformer encoder as the network backbone with tokenized card state input, which is trained by Monte-Carlo reinforcement learning with phased round reward. Experimental results show that our AI achieves a winning rate of 53% and +2.02 average difference point versus experienced human players in multi-round *Koi-Koi* games. Moreover, with the aid of attention mechanism, we provide a novel view for analyzing the playing strategy. Such framework design can be applied to various card games. This project is available at <https://github.com/guansanghai/KoiKoi-AI>.

Impact Statement—After the great success of AlphaGo and AlphaZero in board games, AI for card games have also started receiving increasing interest. With more complicated rules and imperfect information, they have higher strategic and uncertain decision-making process, and have become a emergent frontier of AI research. In this paper, we focus on a classic playing card game *Koi-Koi*, which has representative characteristics including multi-round, multiple action types, complex decks and winning hands. In order to overcome these challenges, we introduce the popular Transformer architecture to our work. Through reinforcement learning from zero, our trained *Koi-Koi* AI reaches the level of experienced human players, which is the best performance at state-of-the-art. Moreover, our designed framework provides better transportability and interpretability for various games.

Index Terms—Game AI, card games, Transformer, deep reinforcement learning, DQN.

This work was supported in part by the Fundamental Research Funds for the Central Universities, in part by the NSFC under Grant 62101384, in part by Chongqing Key Laboratory of Mobile Communication Technology under Grant cqupt-mct-202101, and in part by Shanghai Automotive Industry Science and Technology Development Foundation under Grant 2207. (*Corresponding author: Jingjing Wang*)

Sanghai Guan is with iFLYTEK Research, iFLYTEK Co., Ltd., Hefei 230088, China (e-mail: shguan3@iflytek.com).

Jingjing Wang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: drwangjj@buaa.edu.cn).

Ruijie Zhu is with the School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China (e-mail: zhuruijie@zzu.edu.cn).

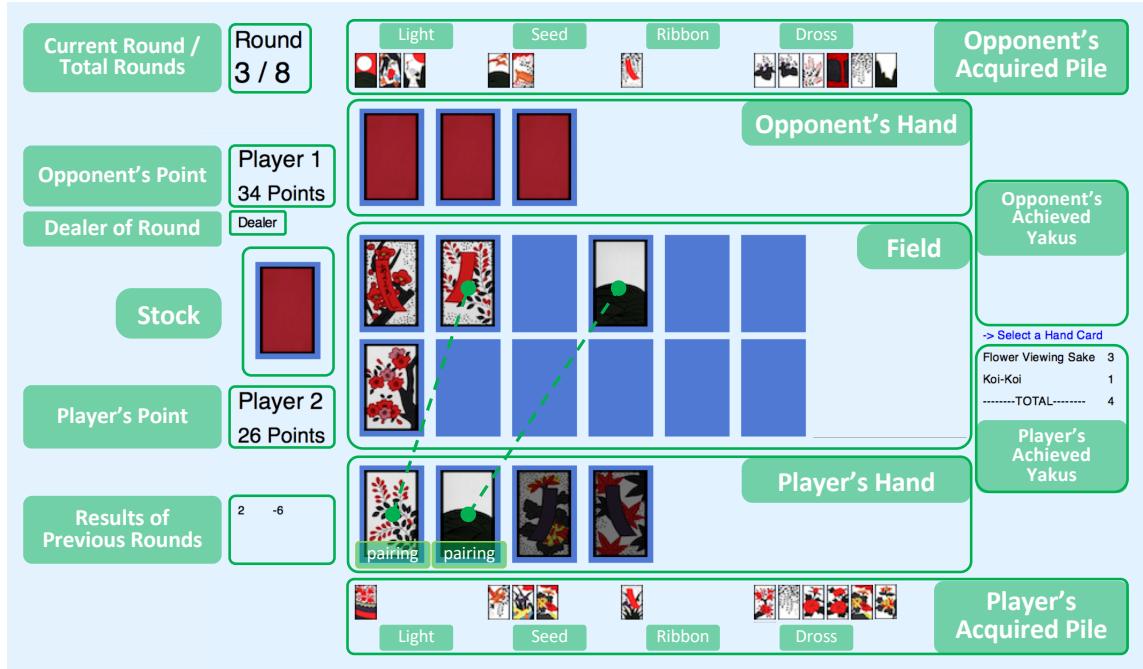
Junhui Qian is with the School of Microelectronic and Communication Engineering, Chongqing University, Chongqing 400044, China (e-mail: junhuiq@cqu.edu.cn).

Zhongxiang Wei is with the College of Electronic and Information Engineering, Tongji University, Shanghai 201804, China (e-mail: z_wei@tongji.edu.cn).

I. INTRODUCTION

WITH the rapid development of artificial intelligence (AI), games have once again attracted the attention of researchers for various rules and deep strategy, which provide a fertile environment for models and algorithms [1]–[3]. As a member of traditional games, card games have been absorbed by players around the world for hundreds of years. Recently, with the aid of deep reinforcement learning [4]–[7], increasing human or even superhuman level card game AI have been proposed, including *Texas Hold'em* [8], *Mahjong* [9], *Dou-Dizhu* [10], and so on. However, there are still some problems limit the participation and application of relevant research. Firstly, most models and algorithms are proposed only for solving specific card games. The design of neural networks relies on the expert experience, while complex handcrafted or look-ahead features are utilized. It is difficult to understand the details, and hard to provide guidelines for other card games. Secondly, card games usually has huge state and action space. For improving performance, some works have to cost much computation resources to conduct real-time look-ahead search, which makes them difficult to deploy. Thirdly, though great contributions have been made in mastering various games, the interpretability of the decision-making logic of AI agents is still a challenge [11]. Giving insight into its playing strategy will motivate both AI research and game study.

In this paper, we focus on *Koi-Koi*, a traditional matching-based card game. In each round of *Koi-Koi* game, both players take turns to pair and collect the cards, and win the points by combining winning hands with their collected cards. It has many representative characteristics of card games, including multiple rounds, multiple action types, complex decks and scoring rules, which bring challenges to realize a high-level AI. Our objective is to design a entire neural network based AI that reaches a high level in *Koi-Koi* games, while achieves simple feature design, easy deployment, and better interpretability. Therefore, a highly general neural network architecture for card games is urgently needed. As one of the most influential achievements in recent years, Transformer [12], which is based on entirely self-attention mechanism, has been widely applied in the fields of natural language processing [13]–[15] and computer vision [16]–[18]. We believe that most of its advantages are also applicable to card game AI. Furthermore, by properly designing the tokenized input and output, it can be easily deployed for different card decks and rules. Because the dependencies between cards and the overall game state can be naturally established, complex handcrafted features



(a) An observation of the player in *Koi-Koi* game.

(b) Examples of the input encoding of the player's observation above.

Fig. 1. A game interface of *Koi-Koi* and corresponding examples of the input encoding. The card token based encoding contains the following information: (1) Card State: current position of cards (hand, field, and both acquired piles), historical actions (including discard, pick, draw, claim koi-koi in each turn); (2) Card Attribute: suit and category of cards so that the network can distinguish each card; (3) Game Progress: common game state information (the dealer, round index, points of players, and so on).

with experts experience can be avoided to the greatest extent. Lastly, the multi-head self-attention mechanism is also more explainable [19], which is conducive to the analysis of game tactics.

In this work, we realize a human-level *Koi-Koi* AI. It takes an entire neural network framework and trained with deep reinforcement learning. We use a Transformer encoder architecture, and adopt “deck tokens + special tokens” input encoding, which only contains basic observed information. With Mote-Carlo learning from zero and phased round reward, the agents reaches state-of-the-art performance. In case study, playing strategy of AI is further clarified with the aid of attention weighting visualization [20]. In addition, open-source game environment, trained models and datasets are provided.

The remaining content is arranged as follows. In Section II, we briefly introduce the rules and the decision flow of *Koi-Koi* games. Then we summarize the existing researches on

card games in Section III. Our proposed neural network architecture and machine learning algorithms are discussed in Section IV. The performance and case analysis of our trained agents in experiments are given in Section V, followed by our conclusions and future works in Section VI. In addition, a detailed introduction of *Koi-Koi* games is supplemented in Appendix.

II. BACKGROUND

Koi-Koi is a multi-round two-player matching-based card game played with a hanafuda deck of 48 cards. An example of game interface is shown in Fig. 1a. In each round of *Koi-Koi*, both players are dealt with 8 private hand cards, while another 8 cards are turned face up on the field, and the remaining 24 cards compose the stock. Then, the dealer plays first, and two players discard from hand and draw from stock in turn. The decision flow of a turn is summarized in Fig. 2. In brief,

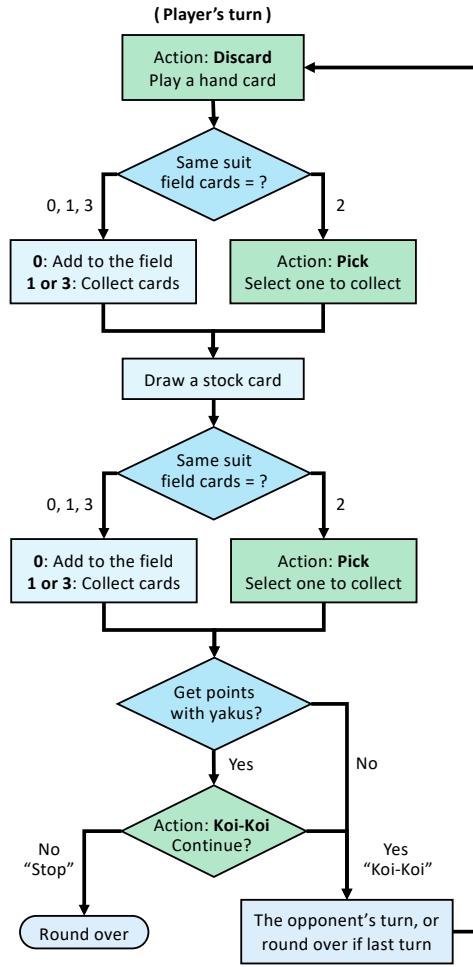


Fig. 2. The decision flow of a round of *Koi-Koi* game. The decision nodes in player’s turn include: (1) Discard a card from hand; (2) Choose a filed card paired with discarded card; (3) Choose a filed card paired with drawn card; (4) Determine whether to “koi-koi”.

if there is a field card with the same suit of the player’s discarded or drawn card, he collects them into his acquired pile. The player’s goal is to make the cards in acquired pile form specific winning hands called “yaku”. Different yaku are with different points according to the difficulty of achieving. When a player achieves any yaku in his turn, he can continue this round for more yaku by claiming “koi-koi”¹, or he can claim “stop”, becoming the winner of this round and the dealer of the next round. In this way, the loser pays the winner the total points of winner’s all achieved yaku. In the last round, the one with the higher point wins the game. Detailed rules and the list of yaku *Koi-Koi* game are attached in the Appendix. In this paper, the total number of rounds is set as 8, and both players’ initial points are 30. If the points of anyone are less than or equal to zero, he loses the game directly.

In our opinion, *Koi-Koi* is a suitable card game environment for deep reinforcement learning. Firstly, it is a two-player game with imperfect information. It also provides random

¹It should be noted that, “koi-koi” means not only the name of the game, but also a type of action in this game, which can be claimed when a player achieves yaku in order to continue this round. In this paper, we distinguish the game *Koi-Koi* and action type koi-koi in different fonts.

hand and stock cards. Moreover, its decision-making process includes card-playing actions including discarding and picking, as well as special actions of claiming koi-koi. These characteristics are representative in most general playing card games. Secondly, its complexity of action and state space is moderate. It has $\sim 10^{49}$ information sets with the average size of $\sim 10^7$. Because each round contains up to 16 turns, and each turn contains up to 4 steps of decision-making, the length of the state transition chain is also appropriate. Thirdly, it has rich playing tactics to explore. For example, should the agent aim at the yaku easy to achieve with fewer points, or the yaku difficult to achieve with higher points; should the agent just focus on the cards he needs, or pair and snatch the opponent’s desired cards for defense; when achieving yaku, should the agent take a radical strategy of claiming “koi-koi” to expand yaku and points with risk, or a conservative strategy of “stop” to ensuring the obtained point and dealer’s advantage of the next round. All of these are determined according to specific situation, including game progress, players’ cards, opponent’s style and so on. Finally, the rules of “pairing by suits” and “forming yaku with cards” establish complex and diverse correlations between cards, hence the powerful feature extraction ability of deep neural networks can be utilized.

III. RELATED WORKS

A. Overview of Card Game AI

Most classic card games are two-player or multi-player zero-sum games with imperfect information [21]. However, these games adopts different decks and rules, which impacts the nature of their game state and action space. As a result, we roughly divide them into the following three categories, where the common applied AI framework are also different:

- **Gambling Poker Games:** The decision-making is focus on the strategy of betting or bidding. Most games use standard French decks, and the winner is determine by the combination of hand cards, such as *Leduc Hold’em*, limit and no-limit *Texas Hold’em*. Such games usually have a large number of information sets but the size of each information set is relatively small [22], and high level game AIs rely on counterfactual regret minimization (CFR) algorithms [23]. In recent years, deep neural network are widely adopted to improve the efficiency of tree search. Typical works include DeepStack [8], Libratus [24], and Pluribus [25] for no-limit *Texas Hold’em*.
- **General Playing Card Games:** The decision-making is focus on the tactics of playing the cards, such as discard or select. This category covers a wide range of decks and rules, which can be further divided into matching games (*Mahjong*, *Koi-Koi*), hedging games (*Rummy*, *Uno*, *Dou-Dizhu*), trick-taking games (*Bridge*), cooperate games (*Hanabi*) [26], [27] and so on. These games usually have a large size of each information set, i.e., much hidden information. Recently, increasing high level AI tend to adopt entire neural network framework without look-ahead tree search. Typical works include Suphx for *Mahjong* [9] and DouZero for *Dou-Dizhu* [10].

- **Collectable / Trading Card Games:** This kind of card games has a huge deck, such as *Magic: The Gathering*, *Hearthstone* and *Yu-Gi-Oh*. The decision-making is focus on selecting cards to build the private stock and formulate corresponding playing tactics. The rules of such games are very complex, and it is difficult to build testing environment and obtain training data. Therefore, most of applied AI are still manual rule-based. However, some progress has been made on heuristic algorithm, supervised learning and card embeddings [28]–[30].

B. Deep Reinforcement Learning in Card Game AI

After the series of successes of AlphaGo, AlphaGo Zero, and AlphaZero [31], [32] in the field of board games including *Go*, *chess* and *shogi*, deep reinforcement learning is increasingly adopted in card game AI.

Firstly, the roles of deep reinforcement learning in AI frameworks are different. In some works, the neural networks exists as an auxiliary component. AlphaGo is a representative work, the policy network and value network of which are used for more efficient Monte-Carlo tree search (MCTS) [33]. Similarly, DeepStack for *Texas Hold'em* utilizes neural networks to fit counterfactual value to optimize the look-forward tree search. In *Bridge*, some work particularly uses reinforcement learning to handle the bidding phase [34]. In contrast, other works apply a entire neural network framework, such as Suphx for *Mahjong* and Douzero for *Dou-Dizhu*, which take actions directly based on the output of the neural network. Such design holds the advantages of fast decision-making and easy deployment. But it is generally believed that neural networks are not good at logical reasoning. As a result, its decision-making accuracy is usually lower than search methods.

Secondly, the reinforcement learning algorithms and network architectures are also various. For example, in DeepStack, a deep full-connected network is designed to predict the counterfactual value. In Suphx, a deep residual convolutional network (CNN) with tiny convolution kernels [35], and the policy gradient algorithm are used to predict the probability of playing each tile. In DouZero, the authors build a mixed long short-term memory (LSTM) [36] and full-connected hybrid network, and use Monte-Carlo learning algorithm to predict the winning probability of each action. Therefore, the rules and characteristics of the game should be seriously considered when designing AI frameworks.

C. Previous Koi-Koi Game AI

At present, existing works of *Koi-Koi* game AI are relatively few, and these works mostly just take single-round *Koi-Koi* game as the test environment. In [37], the authors tried the UCT (Upper Confidence bounds applies to Tree) based MCTS algorithm [38] to build a *Koi-Koi* AI. Due to insufficient performance experiments, this work did not provide a baseline. In [39], the authors presented an adversarial search framework, which utilized an modified expected min-max pruning algorithm. In the experiment, the proposed AI was tested with other two baselines using random search and heuristic greedy search. The results shows that the proposed AI had 56.7%

winning rate versus random search, and 50.8% winning rate versus greedy search. In [40], the authors introduced the deep reinforcement learning AI with a three-layer fully-connected neural network. Two AI agents are trained by policy gradient and Q-learning, respectively. In the test, a rule-based AI is adopted as baseline, which wins +2.5 points per round versus a fully random agent. The results show that the policy gradient AI achieves +0.5 points per round, while the Q-learning AI achieves -0.3 points per round versus the rule-based baseline. Generally, the performance of existing works are only slightly better than heuristic or rule-based algorithms, and are still far behind the level of human players.

IV. METHODOLOGY

A. Neural Network Architecture

Our card-playing model is a modified Transformer encoder, where we adjust the input and output layers to adapt card games, and the overall structure is shown in Fig. 3. In this paper, we use three models with the same network structure but independent weight parameters for three different tasks, i.e., discard, pick, koi-koi. In practice, using a single model or sharing parameters of bottom layers to reduce the size of parameters is also feasible.

Similar to the “word tokens + special tokens” input as Transformer processes sentences, our model adopts a “deck tokens + special tokens” mode to represent the overall observation. As shown in Fig. 1b, the input encoding of 48 columns represent the 48 cards composing a deck. Specifically, each column means the observation of the corresponding card, and this vector for about 300 dimensions includes the following three parts:

- **Card State:** This part records the current position of cards (player’s hand, field, both players’ acquired pile, and not seen) and historical actions in each turn (discard from hand, draw from stock, pair, collect into acquired pile and claim koi-koi), which implicates complete state information of this round, and are encoded as 0-1 values. As the example in Fig. 1b, in the row of **Position = Hand**, cards in player’s hand are marked as 1. While in the row of **Act = Turn 1, Discard**, the card played in the first turn is marked as 1.
- **Card Attribute:** This part reveals the inherent attribute to distinguish each card, including suit, category, and whether it can form each yaku or not, which are all encoded as 0-1 values. As shown in Fig. 1b, in the rows from **Suit = Jan.** to **Category = Seed**, cards with corresponding attributes are marked as 1.
- **Game Progress:** This part is the public information of game progress. These dimensions are the same for all the columns. Some are encoded as continuous values (the players’ points, the number of cards in each position, the number of collected cards of each attribute), and the others are encoded as 0-1 values (the dealer, round index, turn index). For example, the **Dealer = False** row in Fig. 1b is all marked as 1, because the player is not the dealer.

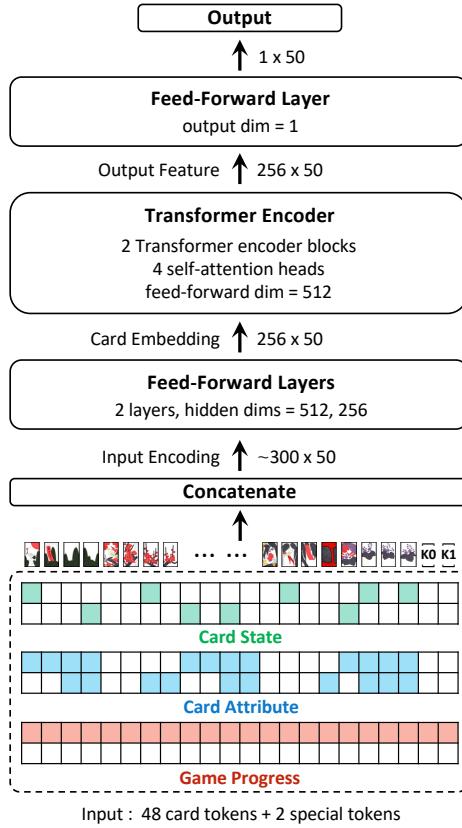


Fig. 3. The illustration of neural network architecture.

All of these features are directly obtained without any lookahead searching or handcrafted design. These three parts are concatenated as the input of our model. Then we pass it through two position-wise feed-forward layers with ReLU activations, and obtain the preliminary processed card embedding. In addition, in order to deal with koi-koi tasks, two columns of special tokens named [K0] and [K1] with unique one-hot card attribute encoding are added. Similar to the [CLS] token in language models, they extract the overall state to judge whether to continue (claiming koi-koi) or stop when successfully forming yakus.

For the backbone networks, we adopt a Transformer encoder, which is composed of multiple encoder blocks. Each block is characterized by a multi-head self-attention layer and a two-layered position-wise feed-forward network. After each module, residual connection [41] and layer normalization [42] are performed. The residual structure enhances the convergence performance and reduces the training difficulty, especially for reinforcement learning tasks. Here we adopt the original layer normalization method, i.e., post-LN.

For the output layer, we directly pass the processed card features through a position-wise linear layer, and take the one-dimension final output. By conducting only on specific card columns, it can naturally mask the output of illegal or irrelevant actions. The order of each column (card token) can also be adjusted freely. These characteristics are particularly conducive to reinforcement learning. For supervised learning and reinforcement learning, the post-processing of outputs is different, which will be furtherly described later.

We believe that such “deck tokens + special tokens” encoding is suitable for various card games. First of all, as the basic element of card games, card encoding carries game state information in a lossless and organized way, and establish correlations with each other to extract features under the attention mechanism. In most card games, the size of a deck is from dozens to over a hundred, so the input length is also within the processing capacity of the Transformer. Then, the special tokens can be flexibly adopted to extract the overall state feature for non-card-playing tasks, such as betting in *Poker*, bidding in *Bridge*, chow, pung, kong in *Mahjong*, as well as for card-playing tasks under some rules which allow to discard multiple cards such as pairs or chains.

B. Supervised Learning Verification

Although our work adopts reinforcement learning from zero, in order to prove the feasibility of the network architecture and provide a reliable baseline, we first trained an agent by supervised learning with human playing records. Here we adopt the model as a multi-classifier. Specifically, for the discard model and pick model, there are 48 classes corresponding to the 48 cards, and for the koi-koi model, it is a binary classifier. We use a small dataset of 200 eight-round *Koi-Koi* games for training. The sample distribution and result are summarized in Table I, where we use 5-fold cross validation, and each card playing model. It can be found that the model fits the choices of human players. Generally speaking, it is difficult for a supervised learning based card game AI to obtain high performance. In addition, considering the size of the dataset is small, it may face overfitting problems. Therefore, in the following, we will introduce our reinforcement learning framework to train a high-level AI from scratch.

TABLE I
ACCURACY OF SUPERVISED LEARNING MODELS

Action Type	# of Samples	Accuracy
Discard	17 821	0.740
Pick	1846	0.903
Koi-Koi	1768	0.802
Overall	21 435	0.759

C. Reinforcement Learning Framework

We adopt the value-based method for deep reinforcement learning (RL) [43]. The card playing model is utilized as a deep Q network (DQN) [44] and the agents are trained with self-play Monte-Carlo learning. We take each round of a game as an episode, and establish a phased round reward model. The pipeline of our reinforcement learning framework is illustrated as Fig. 4.

1) *Round Reward Model*: Since the goal of agents is to get more points than the opponent to win the game, we can give a reward of 1 to the winner and 0 to the loser at the end of the game. However, *Koi-Koi* is a multi-round game. In a new round, only point and dealer state are inherited from the

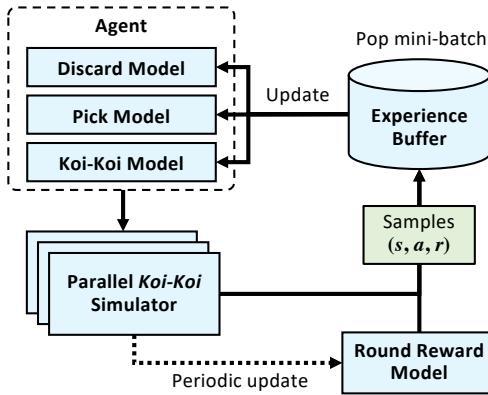


Fig. 4. The reinforcement learning training pipeline.

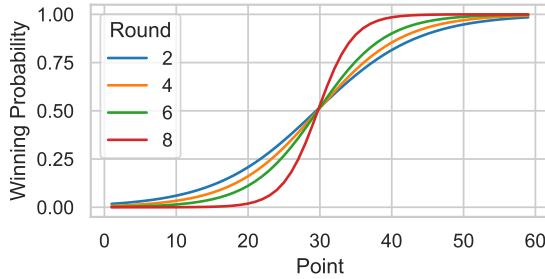


Fig. 5. An example of the dealer's predicted winning probability under different rounds and points, obtained by fitting self-play records with logistic regression.

previous round. Therefore, we choose the trajectory of each round as an episode, and provide a phased reward [45].

In this work, two kinds of round reward are tried, and the corresponding agents are named RL-Point and RL-WP, respectively. For RL-Point agent, we directly adopt the points obtained or lost in this round as the reward. For RL-WP agent, according to the points and dealer after this round, we estimate the probability of winning this game as the reward. Hence, we maintain a winning probability predictor, composed of the logistic regression models for each round. The agent self-plays with the optimal (greedy) policy to generate samples and train the predictor. Because the agent's performance improves during the training process, the predictor is updated periodically. Fig. 5 provides an example of winning probability prediction. It can be found that the closer to the last round, the faster the winning probability changes with the point, and the greater the advantage of the leader.

2) *DQN with Monte-Carlo Learning*: We take the network model as a DQN, and its prediction output $Q(s, a)$ represents the expected reward after taking action a at player-observed state s . Our adopted learning method is Monte-Carlo learning, which is also called deep Monte-Carlo introduced in [10]. It can also be regarded as Q-learning with infinite-step reward and discount factor of $\lambda = 1$. This method is easy to implement and has good convergence performance, but can be only applied to the environment where the state transition ends in finite steps. Compared with Q-learning, it avoids the overestimation caused by the next state reward

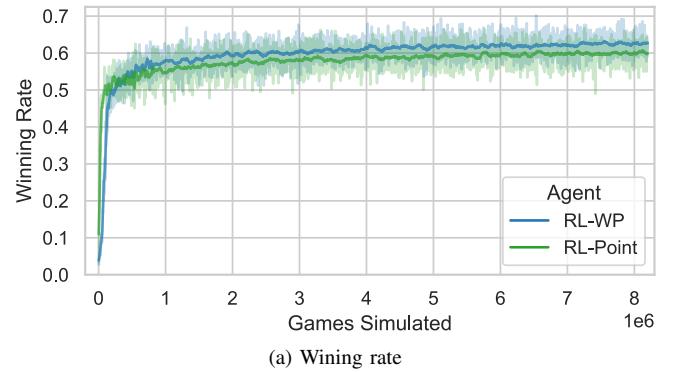


Fig. 6. The performance of the reinforcement learning agents against the supervised learning baseline in eight-round *Koi-Koi* games during the training process (RL-WP: trained with winning probability reward, RL-Point: trained with point reward).

prediction [46]. However, its disadvantages are high variance and low sample utilization rate. Because samples generated under the current policy are invalid after policy update, more computation resources are consumed for trajectory simulation.

Specifically, as the training loop shown in Fig. 4, in the parallel simulator, the agent self-plays with ϵ -greedy policy to generate episode trajectories, where the probability of taking random action ϵ gradually decreases in the training process from 0.15 to 0.02. The trajectory of each round contains multiple state-action pairs $(s_1, a_1), \dots, (s_n, a_n)$. The round reward model provides the reward r according to the end state, and forms sample triplets $(s_1, a_1, r), \dots, (s_n, a_n, r)$ that pushed into the experience buffer. The samples are stored separately by three types of action, i.e., discard, pick and koi-koi, for the three models. After sufficient samples are generated, in the last phase of the training loop, the buffer randomly pops samples as mini-batches to perform a step of optimization until exhausted. We take the mean square error (MSE) as the loss function, representing the estimated error of $Q(s, a)$ between the round reward model and the DQN.

In our implementation, when assembling sample (s, a, r) , we directly move the encoding vector of agent-taken action a to the first column of s , and simplified as (s, r) . Hence, the output layer can directly conduct on the first column of mini-batched output feature.

3) *Training Process*: The training process of agents includes two phases. In the first phase, in order to speed up, the agent plays against the previously introduced supervised

learning agent until overwhelms it. In the second phase, we let the agent self-play to improve the performance. Our hardware environment is made up of a 12-core Intel Xeon Platinum CPU (for parallel game simulation) and an NVIDIA Tesla P100 GPU (for network optimizing), training for over 10 million games in one week. In each loop of the training pipeline, the parallel game simulator generates trajectories of 2500 games. The batch size is set as 256, and we adopt the Adam optimizer with leaning rate of $1e^{-4}$.

Fig. 6 shows the performance of agents against the supervised learning baseline during the training process, including the winning rate and the average point difference relative to the initial point (for example, if the game ends with 32 points, it is counted as +2.0). We can find that RL-Point agent learns faster at the beginning, but RL-WP agent reaches a higher winning rate. RL-WP agent beats the supervised learning baseline after 500,000 games playing against it. In the second phase, its performance continues to gradually improve through self-play.

V. EXPERIMENTS

A. Design and Results

In order to test the performance of our trained agents, we conducted dueling between them. We calculated the average winning rate and the point difference relative to the initial point, and summarized in Table II, of which each grid represents the result of the row agent against the column agent. Our test includes the following agents:

- **Author:** In order to provide a rough baseline of experienced players, the authors played 50 games with RL-WP and RL-Point agents, respectively.
- **RL-WP:** Reinforcement learning agent trained with winning probability reward, introduced in Section IV-C.
- **RL-Point:** Reinforcement learning agent trained with round point reward, introduced in Section IV-C.
- **SL:** Supervised learning agent trained with human game record, introduced in Section IV-B.
- **Random:** Weak agent randomly discards and picks cards, and koi-koi with the probability of 0.5.

TABLE II
WINNING RATE AND POINT DIFFERENCE BETWEEN AGENTS

	RL-WP	RL-Point	SL	Random
Author	0.52 - 1.04	0.47 - 2.02	-	-
RL-WP¹	-	0.515 - 2.88	0.636 + 3.44	0.992 + 27.41
RL-Point²	-	-	0.616 + 6.13	0.982 + 31.01
SL³	-	-	-	0.982 + 20.05

* The average results of 50 games for human player vs. AI agent, 2000 games for AI agent vs. AI agent.

¹ RL-WP: reinforcement learning with winning probability reward.

² RL-Point: reinforcement learning with point reward.

³ SL: supervised learning.

First of all, RL-Point agent achieves 53% winning rate and +2.02 average difference point in the 50 games played against human, which achieves the best performance at state-of-the-art, while RL-WP agent achieves 47% winning rate and +1.04 average difference point. As introduced in the related works, this is the first AI that reaches the level of experienced human players in multi-round *Koi-Koi* games. Our open-source trained models and test environments also provide a reliable baseline for future research. Besides, for the results between AI agents, the reinforcement agents play better than the supervised learning agent, and all better than the random agent. In the performance comparison between the RL-WP and RL-Point agent, the RL-WP agent always gets a higher winning rate and the RL-Point agent always obtains more points. This shows that different reward settings have a significant impact on the playing style and strategy of trained AI, which will be further analyzed in the following.

B. Case Analysis and Discussion

In this part, we would like to share two specific cases. In the first case shown as Fig. 7a, the player has a good beginning in the first round as the dealer, and needs to select a hand card to discard. There are multiple light cards in hand, and the *Sake Cup* card has been collected into the acquired pile. We input the state encoding into RL-WP and RL-Point agents and predict rewards of playing each card. Both agents make an optimistic estimate of the game. RL-WP agent tends to directly play the *Curtain* card to form *Flower Viewing Sake* yaku and claim koi-koi. While, RL-Point agent chooses to play and collect the *ribbed Wisteria* card, which remains broader space for taking different tactics in the next turns. For remaining actions, the *ribbed Cherry Blossom* is not a good choice because it will make the *Curtain* card unable to pair and collect. While for the other cards, the player can keep them in hand and wait for the opportunity, aiming at yakus including *Three Lights*, *Moon Viewing Sake* and so on.

In Fig 7b, we discuss a more complex situation. The game has reached the seventh round, the opponent is the dealer and the player only leads with a few points. At this time, most high-value cards have not appeared, which also increases the uncertainty in the following turns. In this case, RL-Point agent tends to pair and collect the *Moon* card, waiting for the light and *Sake Cup* cards to appear, and get the maximum expected points for this round. However, for RL-WP agent, the optimum action is to pair and collect *Wild Goose*, which aims at collecting five seed cards and forming the *Tane* yaku to end this round in the fastest way. Such a tactic gives up high-point yakus, but avoids potential risks. As a result, the player can maintain the lead of points and obtain the dealer advantage in the last round, which maximizes the global winning probability of the game. It can be concluded that RL-WP agent has more flexible attack-defense judgment and playing tactics, especially in complex situations. Beyond that, experienced players usually estimates the opponent's hand and tactics according to his historical actions. In our test, AI agents can also adjust playing strategy like that.

As aforementioned, the attention mechanism of the model can be utilized for analyzing the decision-making logic of

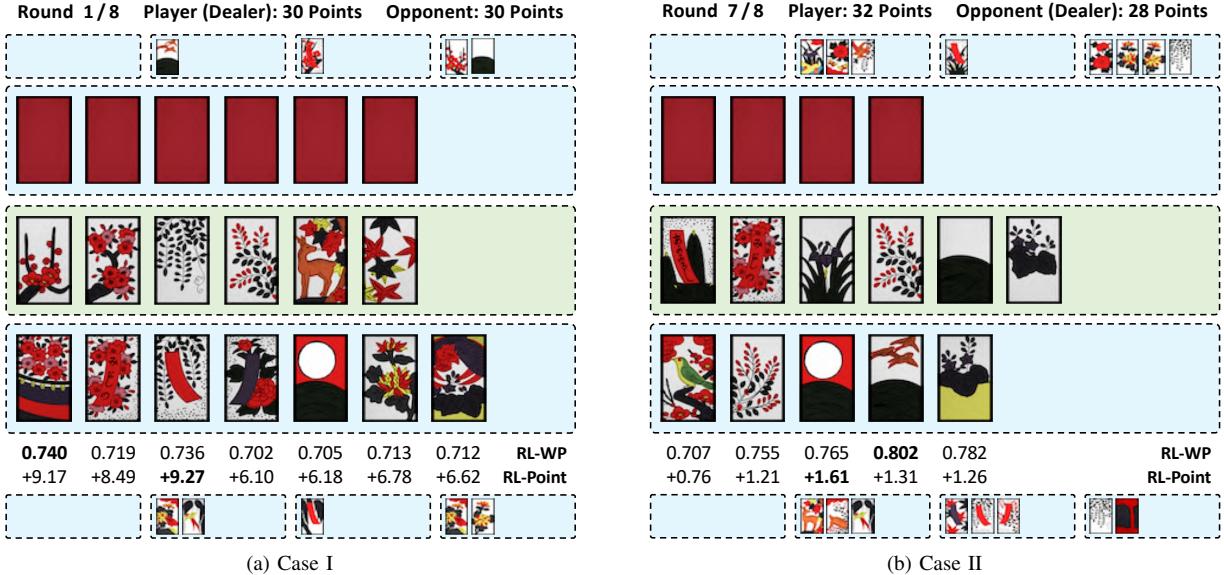


Fig. 7. The predicted reward of reinforcement learning agents (RL-WP: trained with winning probability reward, RL-Point: trained with point reward).

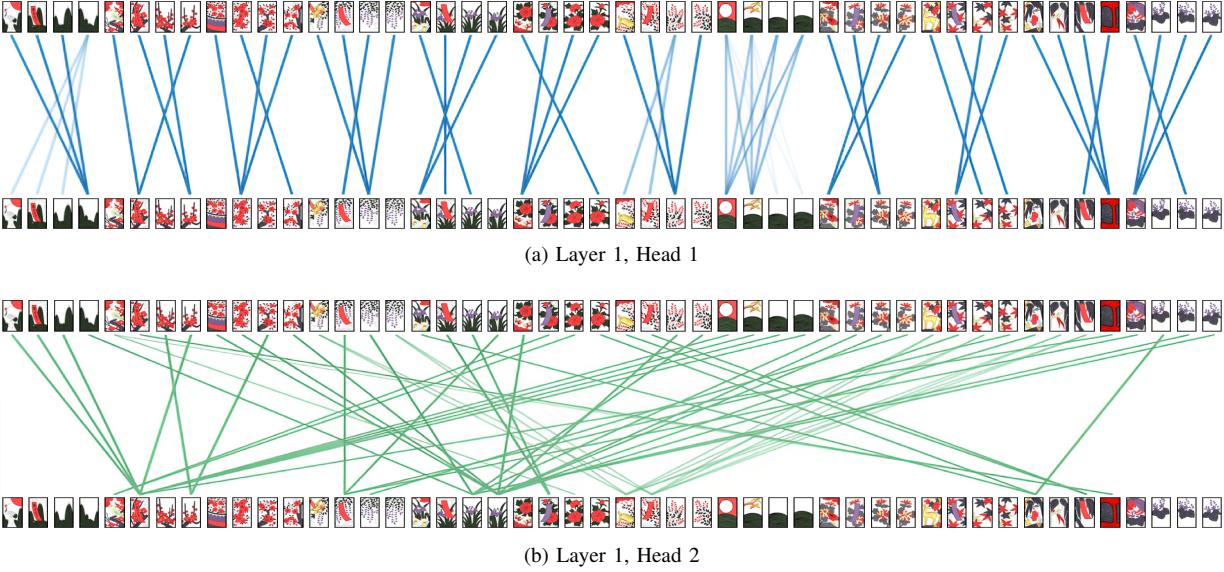


Fig. 8. Partial attention weights between card tokens in the discard model of the reinforcement learning agent with winning probability reward under Case I.

AI agents. Fig. 8 illustrate the weights of some self-attention heads extracting the relationships between cards. Specifically, it is obvious that the first attention head focus on the paring of cards with the same suit. While the other attention heads make connections between other cards as a supplement, in order to obtain both players' progress on yakus, historical actions and so on. It also proves that the Transformer architecture is able to adapt to card token input. The model recognizes the attributes and states of cards, discovers the correlation between cards, and extracts corresponding card features to integrate the overall game state.

VI. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a Transformer based neural network model and trained AI for *Koi-Koi* card games. The multi-head self-attention mechanism is able to learn the complex

dependencies between cards and extract global state features from “deck tokens + special tokens” based input encoding. Trained by Monte-Carlo reinforcement learning with phased round reward, human-level *Koi-Koi* AI is realized. However, there are still some limitations in current work. Firstly, the the round reward model still need optimization. In the winning rate prediction, we used the logistic regression models to fit the self-play data. Due to the limited fitting ability and not considering the opponent's playing style, it is not accurate enough and affected the performance of the RL-WP agent. Secondly, the training speed and sample efficiency need to be further improved due to the use of multi-layer attention architecture and Monte-Carlo learning algorithm. Finally, in the experiment, we only conducted the dueling between agents to roughly compare the performance. If an open test platform can be established for the community and introduces

reliable rating score systems, it will be more conducive to determine the level and also motivate the participation of relevant research. In the future, we intend to carry out further research on the following aspects:

- Optimize the Transformer architecture for card games, including input encoding and embedding methods, multi-head attention mechanisms, post-processing of output layer, and other training tricks.
- Explore effective reinforcement learning algorithms, including advanced value-based or policy-based methods, and reward shaping skills.
- Expand the application to other card games to further prove the generality and adaptability of our method.

The authors hope that this work will inspire more research and interest in card games.

APPENDIX

DETAILED RULES OF KOI-KOI GAMES

A. Hanafuda Cards

Hanafuda², translated as “flower cards”, is a kind of traditional Japanese playing cards. There are many popular hanafuda-based card games including “*Hana-Awase*”, “*Koi-Koi*”, “*Go-Stop*”. As summarized in Table III, a hanafuda deck contains 48 cards divided by 12 suits that correspond to 12 months and represented by a plant on the card. In *Koi-Koi* games, the cards are matched based on their suit. The cards are also divided into four rank-like categories, i.e., light, seed, ribbon and dross (called “*hikari*”, “*tane*”, “*tan*” and “*kasu*”). Light cards are the rarest and most valuable cards printed with bright scenery, while seed cards are featured by animals or crafts on them. The ribbon cards are printed with colorful ribbons. There are three special cards with red poetry ribbons, and three special cards with blue ribbons among them. The dross cards, which only contain plain plant patterns, are the common cards with the lowest value. In particular, the “*Sake Cup*” card belongs to both categories of seed and dross in *Koi-Koi* games.

B. Rules of Koi-Koi

*Koi-Koi*³ is a popular matching-based two-player hanafuda card game. The goal of *Koi-Koi* is to collect cards by matching the cards by suit, and forming specific winning hands called “yaku” from the acquired pile to earn points from the opponent. The name “koi-koi” means “continue” which is claimed when a player forms a yaku and wants to continue the game to earn more points.

A *Koi-Koi* game consists of multiple rounds, usually four to twelve. Both players start with equal points. At the beginning of a game, each player draws a card from the deck, and the player with the earlier month card becomes the dealer of the first round. In each round, each player is dealt with 8 cards from a shuffled deck as his hand, then the next 8 cards are turned face up on the field, and the remaining 24 cards compose the stock. If there are four cards with the same suit

on the field or in a player’s hand, the cards will be re-dealt. Then, the dealer plays first and two players discard, draw and try to match and collect cards in turn. The process of a turn can be summarized as following:

- 1) *Play a Card from the Hand*: The player plays a card from his hand. If there is a field card with the same suit, he collects two cards and adds them into his acquired pile. When there are two field cards with the same suit as the discarded card, the player has to select one. In particular, three cards with the same suit may be dealt on the field at the beginning of a game. In this case, the player takes all three cards along with the discarded card into his acquired pile. However, if there is no field card with the same suit, the discarded card will remain on the field.
- 2) *Draw a Card from the Stock*: Then, this player turns a card face up from the top of the stock, tries to match this card with field cards and collects them according to the same rule of the previous step.
- 3) *Check Yaku of the Acquired Pile*: According to the list of yaku⁴ summarized in Table IV, if the acquired pile is able to form a new yaku, or add points to already formed yaku (including Tane, Tan and Kasu), the player needs to choose “koi-koi” or “stop” (unless it is his last turn of this round). If he claims “stop”, he wins this round and receives points from the opponent based on the yaku formed by his acquired pile. The winner will become the dealer of the next round. If he claims koi-koi, or there are no new yaku or additional point, his turn is over and the round continues. The opponent will repeat the above process.

Koi-Koi game obeys the “winner takes all” rule, where the loser has to give points to the winner according to the winner’s yaku, regardless of his own acquired pile or formed yaku. Hence, claiming “koi-koi” brings potential benefits and risks at the same time. If both players have finished all their turns with all hand cards exhausted and neither of them claims “stop”, this round ends in a stalemate. The dealer will benefit from the rule of “Dealer’s Priority” by receiving one point from the opponent, and remains the dealer of the next round.

After the last round, the player with more points is the winner of the game. If both players are with the same points, the game is tied. In particular, once a player has lost all his points after any round, he loses the game directly.

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] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [3] 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.

⁴We adopt the same rules and yaku list as the PC game “*Koi-Koi Japan – Hanafuda Playing Cards*” on Steam, available at https://store.steampowered.com/app/364930/KoiKoi_Japan_Hanafuda_playing_cards.

²<https://en.wikipedia.org/wiki/Hanafuda>

³<https://en.wikipedia.org/wiki/Koi-Koi>

TABLE III
A DECK OF HANAFUDA CARDS

Suit (Pattern)	Category				Suit (Pattern)	Category			
	Light (Hikari)	Seed (Tane)	Ribbon (Tan)	Dross (Kasu)		Light (Hikari)	Seed (Tane)	Ribbon (Tan)	Dross (Kasu)
Jan. (Mutsuki) Pine (Matsu)					Feb. (Kisaragi) Plum Blossom (Ume)				
Mar. (Yayoi) Cherry Blossom (Sakura)					Apr. (Uzuki) Wisteria (Huji)				
May. (Satsuki) Iris (Ayame)					Jun. (Minazuki) Peony (Botan)				
Jul. (Fumizuki) Bush Clover (Hagi)					Aug. (Hazuki) Grass (Susuki)				
Sep. (Nagatsuki) Chrysanthemum (Kiku)					Oct. (Kannazuki) Maple (Momiji)				
Nov. (Shimotsuki) Willow (Yanagi)					Dec. (Shiwatsu) Paulownia (Kiri)				

* The “Sake Cup” is with dual categories, which is also a dross card and is able to compose the “Kasu” yaku.

TABLE IV
LIST OF YAKUS (WINNING HANDS) IN KOI-KOI GAME

Yaku (Winning Hand)*	Points	Description
Five Lights	10	Acquire all five light cards (incompatible with Four Lights, Rainy Four Lights and Three Lights).
Four Lights	8	Acquire four light cards excluding “Rainman” (incompatible with Three Lights).
Rainy Four Lights	7	Acquire four light cards including “Rainman” (incompatible with Three Lights).
Three Lights	5	Acquire three light cards excluding “Rainman”.
Boar-Deer-Butterfly	5	Acquire “Boar”, “Deer” and “Butterfly”.
Flower Viewing Sake	3 or 1	Acquire “Curtain” and “Sake Cup” (Combining “koi-koi” makes it 3 points, otherwise 1 point).
Moon Viewing Sake	3 or 1	Acquire “Moon” and “Sake Cup” (Combining “koi-koi” makes it 3 points, otherwise 1 point).
Tane “the Seed Cards”	1 (+1)	Acquire five or more seed cards (+1 point for every additional card).
Red & Blue Ribbons	10	Acquire all six red poetry and blue ribbon cards (compatible with Red Ribbons and Blue Ribbons).
Red Ribbons	5	Acquire all three red poetry ribbon cards.
Blue Ribbons	5	Acquire all three blue ribbon cards.
Tan “the Ribbon Cards”	1 (+1)	Acquire five or more ribbon cards (+1 point for every additional card).
Kasu “the Dross Cards”	1 (+1)	Acquire ten or more dross cards (+1 point for every additional card).
Koi-Koi Bonus	1/2/3/ ×2–5	Bonus for claiming “koi-koi” (1 point, 2 points, 3 points if a player claims “koi-koi” once, twice, three times in a round, while total points $\times 2$, $\times 3$, $\times 4$, $\times 5$ for 4, 5, 6, 7 times, respectively).
Dealer’s Priority**	1	The dealer gets 1 point if the round ends with all hand cards exhausted.

* When calculating the total points, except the Lights series where only the one with the highest points is counted, all the yakus are compatible.

** For Dealer’s Priority, the dealer receives only 1 point from the opponent even if he has collected yakus but claimed “koi-koi”. In other words, a player’s collected yakus are only taken into account when he properly claims “stop” to end the round.

- [4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
- [5] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *International Conference on Machine Learning*. PMLR, 2014, pp. 387–395.
- [6] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, “Dueling network architectures for deep reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2016, pp. 1995–2003.
- [7] 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*. PMLR, 2016, pp. 1928–1937.
- [8] M. Moravčík, M. Schmid, N. Burch, V. Lisý, 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] 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.
- [10] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, and J. Liu, “DouZero: Mastering DouDizhu with self-play deep reinforcement learning,” in *International Conference on Machine Learning*, 2021, pp. 12333–12344.
- [11] A. Rawal, J. McCoy, D. B. Rawat, B. Sadler, and R. Amant, “Recent advances in trustworthy explainable artificial intelligence: Status, challenges and perspectives,” *IEEE Transactions on Artificial Intelligence*, pp. 1–1, 2021.
- [12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional Transformers for language understanding,” *arXiv preprint, arXiv:1810.04805*, 2018.
- [14] A. Rogers, O. Kovaleva, and A. Rumshisky, “A primer in BERTology: What we know about how BERT works,” *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2020.
- [15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
- [16] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, “A survey on visual Transformer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2022.
- [17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [18] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16000–16009.
- [19] F. K. Došilović, M. Brčić, and N. Hlupić, “Explainable artificial intelligence: A survey,” in *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, pp. 210–215.
- [20] H. Chefer, S. Gur, and L. Wolf, “Transformer interpretability beyond attention visualization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 782–791.
- [21] D. Fudenberg and J. Tirole, *Game Theory*. MIT press, 1991.
- [22] D. Zha, K.-H. Lai, S. Huang, Y. Cao, K. Reddy, J. Vargas, A. Nguyen, R. Wei, J. Guo, and X. Hu, “RLCard: a platform for reinforcement learning in card games,” in *International Joint Conferences on Artificial Intelligence*, 2021, pp. 5264–5266.
- [23] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, “Regret minimization in games with incomplete information,” *Advances in Neural Information Processing Systems*, vol. 20, 2007.
- [24] 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.
- [25] ———, “Superhuman AI for multiplayer poker,” *Science*, vol. 365, no. 6456, pp. 885–890, 2019.
- [26] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes *et al.*, “The hanabi challenge: A new frontier for ai research,” *Artificial Intelligence*, vol. 280, p. 103216, 2020.
- [27] M. Eger, C. Martens, P. S. Chacón, M. A. Córdoba, and J. H. Cespedes, “Operationalizing intentionality to play hanabi with human players,” *IEEE Transactions on Games*, 2020.
- [28] A. R. Da Silva and L. F. W. Goes, “Hearthbot: An autonomous agent based on fuzzy art adaptive neural networks for the digital collectible card game HearthStone,” *IEEE Transactions on Games*, vol. 10, no. 2, pp. 170–181, 2017.
- [29] M. Świechowski, T. Tajmajer, and A. Janusz, “Improving hearthstone AI by combining MCTS and supervised learning algorithms,” in *IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.
- [30] T. Bertram, J. Fürnkranz, and M. Müller, “Predicting human card selection in Magic: the Gathering with contextual preference ranking,” in *IEEE Conference on Games (CoG)*. IEEE, 2021, pp. 1–8.
- [31] 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.
- [32] 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.
- [33] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, “A survey of Monte Carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.
- [34] C.-K. Yeh, C.-Y. Hsieh, and H.-T. Lin, “Automatic bridge bidding using deep reinforcement learning,” *IEEE Transactions on Games*, vol. 10, no. 4, pp. 365–377, 2018.
- [35] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, “Recent advances in convolutional neural networks,” *Pattern recognition*, vol. 77, pp. 354–377, 2018.
- [36] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [37] Y. Takaoka, T. Kawakami, and R. Ooe, “A study on strategy acquisition on imperfect information game by UCT search,” in *IEEE/SICE International Symposium on System Integration (SII)*, 2017, pp. 881–886.
- [38] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.
- [39] A. B. Cruz, L. Preuss, J. Quadros, U. Souza, S. Serique, A. Ogasawara, E. Bezerra, and E. Ogasawara, “Amé: an environment to learn and analyze adversarial search algorithms using stochastic card games,” in *30th Annual ACM Symposium on Applied Computing*, 2015, pp. 208–213.
- [40] N. Sato and K. Ikeda, “Applying policy gradient method and neural fitted Q iteration for hanafuda Koi-Koi game player,” in *22nd Game Programming Workshop*. Information Processing Society of Japan, 2017, pp. 64–71.
- [41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [42] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, “On layer normalization in the Transformer architecture,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 10524–10533.
- [43] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.
- [44] A. Ramaswamy and E. Hüllermeier, “Deep Q-learning: Theoretical insights from an asymptotic analysis,” *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 139–151, 2021.
- [45] M. Grześ, “Reward shaping in episodic reinforcement learning,” in *16th Conference on Autonomous Agents and MultiAgent Systems*, 2017, pp. 565–573.
- [46] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.



Sanghai Guan (Member, IEEE) received the B.S. degree in electronic engineering from Dalian University of Technology, Liaoning, China in 2017, and the M.S. degree in information and electronic engineering from Tsinghua University, Beijing, China in 2020. He is currently a junior researcher with iFLYTEK Research, iFLYTEK Co., Ltd., Anhui, China. His research interests include machine learning and deep learning, complexity and network science, as well as heuristic optimization.



Jingjing Wang (Senior Member, IEEE) received his B.S. degree in Electronic Information Engineering from Dalian University of Technology, Liaoning, China in 2014 and the Ph.D. degree in Information and Communication Engineering from Tsinghua University, Beijing, China in 2019, both with the highest honors. From 2017 to 2018, he visited the Next Generation Wireless Group chaired by Prof. Lajos Hanzo, University of Southampton, UK. Dr. Wang is currently a full professor at School of Cyber Science and Technology, Beihang University. His

research interests include AI enhanced next-generation wireless networks, UAV swarm intelligence and confrontation. He has published over 100 IEEE Journal/Conference papers. Dr. Wang was a recipient of the Best Journal Paper Award of IEEE ComSoc Technical Committee on Green Communications & Computing in 2018, the Best Paper Award of IEEE ICC and IWCMC in 2019.



Ruijie Zhu (Member, IEEE) received the Ph.D. degree from the State Key Laboratory of Information Photonics and Optical Communication, Beijing University of Posts and Telecommunications, Beijing, China, in 2017. He is currently an Associate Professor with Zhengzhou University, Zhengzhou, China. He was a visiting scholar with the University of Texas at Dallas, Dallas, TX, USA, under the supervision of Prof. J. P. Jue. His research interests include reinforcement learning, network virtualization, and machine learning.



Junhui Qian (Member, IEEE) received the Ph.D. degree in signal and information processing from the University of Electronic Science and Technology of China, Chengdu, China, in 2018. From 2016 to 2017, he was a Visiting Graduate Researcher with the Electrical Engineering Department, Columbia University, New York, NY, USA. He is currently an associate professor with Chongqing University, Chongqing, China. His current research interests include signal processing for MIMO radar and communicate on systems, artificial olfaction electronic nose, and biomedical and modern signal processing technology.



Zhongxiang Wei (Member, IEEE) received his Ph.D. degree from the University of Liverpool in 2017. From March 2016 to March 2017, he was with the Institute for Infocomm Research, Agency for Science, Technology, and Research (A*STAR), Singapore, as a research assistant. He is an associate professor of electronic and information engineering at Tongji University, Shanghai, China. He has served as a technical program committee chair/member for various international flagship conferences. His research interests include multiple-input and multiple-output systems, physical layer security, and anonymous communication designs.