Applied Intelligence
https://doi.org/10.1007/s10489-022-04050-w

Behavior imitation of individual board game players

Chao-Fan Pan[1] · Xue-Yang Min[2] · Heng-Ru Zhang[1] · Guojie Song[2] · Fan Min[1]

Accepted: 27 July 2022

Abstract
Modeling and predicting player behavior is of the utmost importance in game development and matchmaking. A variety of methods have been proposed to build artificial intelligence (AI), human-like players. However, these human-like players have a limited ability to imitate the behavior of individual players. In this paper, we propose a player behavior imitation method using imitation learning under the framework of meta-learning. A generic behavior model of game players was learned from historical records using adversarial imitation learning. Then, we personalized the policy by imitating the behavior of each individual player. Convolutional neural networks were used to construct the feature extractor of game board states. The experiments were conducted using the Reversi game, and 18,000 game records of different players were used to train the generic behavior model. The behavior of each new player was learned using only hundreds of records. The results demonstrate that our method can be utilized to imitate individual behavior in terms of action similarity well.

Keywords Behavior imitation · Gaming · Imitation learning · Meta-learning

1 Introduction

Artificial intelligence imitation agents (AI^{2}As) aim to pass the Turing test [1], deceiving human experts into believing that they are human players. In the short term, these agents can increase the attractiveness of board games such as chess [2, 3], Go [4, 5], Reversi [6, 7], Hex [8, 9], Poker [10, 11], Spades [12, 13] and Mahjong [14, 15]. In the long run, they can provide support for social computing, such as film script design [16], company operation simulation [17], and government policy preview [18]. Compared with AI agents striving to reach the highest level in games (see, e.g., [19, 20]), the study on AI^{2}As is far from sufficient (see e.g., [2, 21]).

Currently, AI^{2}As face the following challenges. The first challenge is how to imitate the diverse behavior styles of different players. The goals of players are diverse and include more than pursuing the optimal solution. Therefore, players also have preferred behavioral styles. Most existing approaches only use supervised learning to replicate the actions of players. Hence, the diversity is low. The second challenge is how to imitate players with a particular behavioral style. In games, extremely strong performance is often achieved by aggregating data from many players. When playing against particular players, a model that can still accurately capture the player's actions is of clear use for automating different forms of interaction with them. The ultimate realization of AI^{2}As is the ability to emulate players at the individual level. The third challenge is how to imitate players with limited records. Most existing imitation methods require many demonstrations, even in simple games. However, individual players often have a small number of game records that can be imitated. The fourth challenge is how to handle games with great strategic depth. With the increase of strategic depth, the behavioral simulation becomes more difficult due to the complexity of players' behavior. Currently, there is no general approach to handle this situation.

✉ Fan Min
minfan@swpu.edu.cn

Chao-Fan Pan
pan.chaofan@foxmail.com

Xue-Yang Min
mitchellemin@163.com

Heng-Ru Zhang
zhanghr@swpu.edu.cn

Guojie Song
songgj@swpu.edu.cn

[1] School of Computer Science, Southwest Petroleum University, Chengdu 610500, People's Republic of China

[2] School of Sciences, Southwest Petroleum University, Chengdu 610500, People's Republic of China

Published online: 07 September 2022

In this paper, a behavior imitation method is proposed for individual board game players to address some of these challenges using three techniques: generic policy learning, individual player imitation and board state feature extraction. From the algorithmic viewpoint, we use adversarial imitation learning rather than supervised learning to learn more diverse behaviors from gameplay records. As a result, the intentions of players may be better captured. From the architectural viewpoint, meta-learning is adopted to construct an adaptive framework and improve the performance of behavior imitation for individual players by using other players' records. From the data viewpoint, our feature extractor preprocesses the state of the game board. With this technique, the learning effect of a given player with limited records is enhanced.

Note that behavior imitation in this paper has a different optimization objective from that of imitation learning. In imitation learning, the only aim is to learn a skill to handle a task. It is usually assumed that the demonstrations are approximately optimal, thus the optimal policy is learned. For example, an imitation learning agent of Reversi aims to win as much as possible. In contrast, in behavior imitation, the aim is to learn the player's behavior. The policy that is learned is usually not the best for the environment. For example, an imitation agent aims to deceive people in believing that it is human.

Figure 1 depicts the model structure of our method. The leftmost part represents inputs, including the states and actions of players' records. The output is a personalized policy for a given player. In the upper-middle part, the generic policy is learned. The player behavior models are trained using each player's records by adversarial imitation learning. Policies are used as generators to produce a series of behavior records. Discriminators provide rewards for the update of policies to maximize the similarity of generated records. The generic player behavior model, which represents the commonality of multiple players' behavior, is established through optimization-based meta-learning.

The lower-middle part represents the individual player behavior model, which uses adversarial imitation learning to imitate individual behavior. By initializing the parameters, only a small number of gameplay records are required. The CNN-based feature extractor treats the board state as a noise-free 2D image. It provides more expressive inputs for adversarial imitation learning. Since the game board is fixed, the feature extractor does not need to be retrained when imitating individual behavior. As a result, the learned generic player behavior model can quickly adapt to new players.

The contributions of this paper are three-fold:

1. We propose employing adversarial imitation learning to model the behavior of game players. With our approach, more diverse behaviors can be learned.
2. We use imitation learning for individual player imitation under the framework of meta-learning. By learning from multiple players' records, a generic player behavior model can be established. This improves the data utilization of the behavior imitation method.
3. We build a CNN-based board state feature extractor for Reversi that accelerates the behavior imitation process of individual players by preprocessing input states.

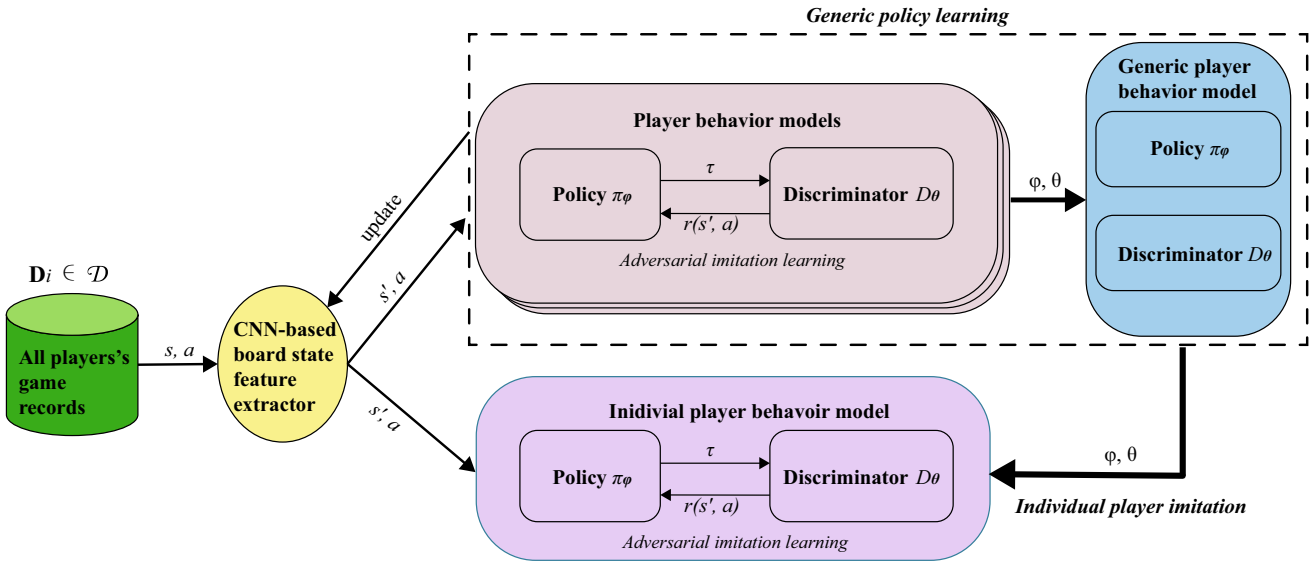


Fig. 1 The proposed player behavior imitation method. The board state feature extractor is used to process the input for training. Under the framework of meta-learning, a generic player behavior model is learned that separately imitates the records of each player to build an individual player behavior model. Notations are defined in Section 2

Experiments were performed using Reversi to evaluate the effectiveness of our method. We selected six different bots as players to be imitated, and their gameplay records were used as datasets. Our method was compared with other popular methods on datasets of different sizes. The results show that our method imitates individual behavior in terms of action similarity well. The source code is available at https://github.com/Dumail/Behavior-imitation, and the data can be downloaded from https://www.botzone.org.cn/downloadmatches.

The remainder of this paper is given as follows: In Section 2, the preliminaries are introduced. In Section 3, the method that we have proposed in this paper is described. The details of the experiments and results are presented in Sections 4 and 5. Some discussions are made in Section 6. Finally, we conclude this study and present future research directions in Section 7.

2 Related works

In this section, the game of Reversi is first introduced. Then, some basic definitions and theorems of the Markov decision process and imitation learning are presented. Table 1 lists some important notations used throughout this paper.

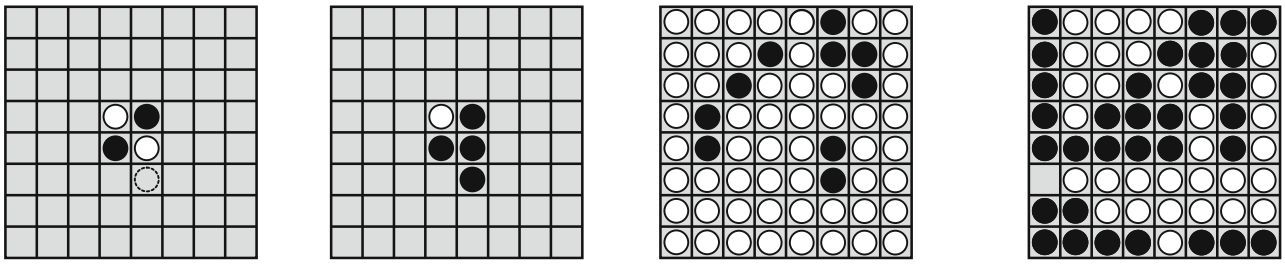
2.1 Reversi

Reversi is a perfect information, zero-sum, two-player strategy game played on a board. There is only one kind of game piece, which is white on one side and black on the other. Figure 2 shows an illustration of some states of a typical 8×8 Reversi board. The game begins with four pieces placed in a square in the center of the board, two facing white-side-up and two facing black-side-up. These pieces are arranged in such a way that the same-colored pieces are on a diagonal. Conventionally, the black-side-up pieces are to the north-east and south-west (Fig. 2a). The player with the black-side-up pieces moves first. Players take turns placing pieces on the board with their assigned color facing up. A move is legal if the newly placed piece is adjacent to an opponent's piece and causes one or more of the opponent's pieces to become enclosed from both sides on a horizontal, vertical, or diagonal line. The enclosed pieces are then flipped to the opposite color. If a black colored piece (dotted circle in Fig. 2a) is placed below the rightmost, white colored piece, the board state will be changed to the one depicted in Fig. 2b. If a player cannot make a legal move, their opponent moves again. The game ends when neither player can make a move. This occurs when the board is filled up (Fig. 2c) or there are no remaining legal moves (Fig. 2d). At this point, the player with more of their pieces on the board wins.

Despite its simple rules, the game of Reversi is far from trivial. The number of legal position sequences is very large (at most 10^{28}). This game also has a game-tree complexity of approximately 10^{58}. Therefore, it is not easy to solve this game using a search algorithm. Reversi can also be characterized by its high temporal volatility: many pieces can be flipped in a single move, dramatically changing the board state. This makes an evaluation of different moves much more difficult. It is essential to capture spatial patterns of the players' and opponents' pieces [7]. This challenge is usually handled by a learning system capable of analyzing decision-making processes.

Table 1 Notations

Notation	Meaning	Notation	Meaning
$\mathscr{M}$	Markov decision process	ψ	Distance function
$\mathscr{S}$	State space	$\mathscr{D}_{\mathrm{JS}}$	Jensen-Shannon distance
s	State	λ	Policy regularization parameter
$\mathscr{A}$	Action space	$\mathscr{H}$	Causal entropy
a	Action	D	Discriminator
r	Reward function	f_θ	Discriminator value function
$\mathscr{P}$	Transition distribution	n	Number of players
$\mathscr{P}_0$	Initial state distribution	$\mathbf{D}_i$	Dataset of player i
γ	Discount factor	$\mathscr{D} = \cup_{i=1}^{n} \mathbf{D}_i$	Total dataset
$T \in \mathbb{N}$	Episode horizon	$\mathscr{D}_r \subset \mathscr{D}$	Training dataset
π	Policy	V	Value of policy
π^*	Optimal policy	L_D	Discriminator loss
π_{E}	Expert policy	ϕ	Policy parameter
τ	Trajectory	M, K, H	Iterations
d	Occupancy distribution	$\mathscr{S}_{\mathrm{act}}$	Similarly of actions
N	Number of players used for the training datasets	$\mathscr{S}_{\mathrm{ran}}$	Random action similarly



(a) Initial board state (b) After first move (c) Board has been filled (d) No remaining legal moves

Fig. 2 Reversi board in different states

2.2 Markov decision process

The Markov decision process (MDP) [22, 23], which is a fundamental model of reinforcement learning, is popular in building game AI agents. Consider the standard MDP $\mathscr{M} = \langle \mathscr{S}, \mathscr{A}, r, \gamma, \mathscr{P}, \mathscr{P}_0, T \rangle$. $\mathscr{S}$ and $\mathscr{A}$ are the state space and action space, respectively. Successor states are given by the transition distribution $\mathscr{P}(s'|s,a) \in [0,1]$, and the initial state s_0 is drawn from $\mathscr{P}_0(s) \in [0,1]$. Transitions are rewarded by the reward function $r : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$. The discount factor is $\gamma \in [0,1]$. The episode horizon is $T \in \mathbb{N}$, where $T < \infty$ for $\gamma = 1$. A stationary stochastic policy $\pi : \mathscr{S} \to \Delta(\mathscr{A})$ produces a trajectory $\tau = (s_0, a_0, s_1, \cdots, s_{T-1}, a_{T-1}, s_T)$ when executed on $\mathscr{M}$.

The probability of trajectory τ under policy π is [22]

$$P_\pi(\tau) := \mathscr{P}_0(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t)\, \mathscr{P}(s_{t+1}|s_t, a_t).$$

The corresponding marginals are defined as $d_{t,\pi}(s) := \sum_{\tau:s_t=s} P_\pi(\tau)$ and $d_{t,\pi}(s,a) := \sum_{\tau:s_t=s,a_t=a} P_\pi(\tau) = d_{t,\pi}(s)\pi(a|s)$. The expected sum of discounted rewards can be expressed in terms of the occupancy measures as follows [23]:

$$V_\pi := \mathbb{E}_{\tau\sim P_\pi}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right].$$

The standard reinforcement learning (RL) objective is to obtain an optimal policy that maximizes the expected sum of the discounted rewards:

$$\pi^* := \arg\max_\pi (V_\pi).$$

In the case of Reversi, the interaction between the player and the game environment can be modeled as a specific MDP. The key components are defined as follows.

- **State space.** A state s is defined as the current position of the pieces on the board. The set of all states constitutes the state space $\mathscr{S}$. The initial state s_0 of all games is the same because the initial board state of Reversi is constant (Fig. 2a). In our implementation, we also include the legal moves of the next turn in the state. Therefore, s and s_0 can be represented as three matrices (Fig. 4).
- **Action space.** The action a indicates the position in which a player places a piece on the board during a move, which must be legal for the current board state. The set of all actions constitutes the action space $\mathscr{A}$. $|\mathscr{A}| = 60$ for an 8×8 board because there are only 60 different positions in which to place pieces.
- **Reward function.** After a legal move is completed, the Reversi game environment returns a reward to the player in the form of reward function $r(s,a)$. Normally, r is closely related to the result of the game. However, in our method, r is built upon the outcome of the discriminator.
- **Transition.** Transition $\mathscr{P}(s'|s,a)$ defines the state transition from s to s' when a player completes action a. In Reversi, the change of the board is determined after the player completes an action; therefore, $\mathscr{P}(s'|s,a) \in \{0,1\}$. For example, when the player with the black colored piece (dotted circle in Fig. 2a) places their piece below the rightmost, white colored piece, the next board state must be the one depicted in Fig. 2b.

2.3 Imitation learning

Imitation learning (IL) [24–26] refers to an agent's acquisition of skills by observing a teacher demonstrating a given task. In this problem, it is assumed that the reward function of the MDP is unknown. However, demonstrations of the expert's policy π_E are provided. The objective of IL is to search for a policy π that is most similar to π_E by minimizing the difference between their expected returns: $\arg\max_\pi \left(V_{\pi_\mathrm{E}} - V_\pi\right)$. When the state-action occupancy measure of policy π is close to that of π_E, the difference between their expected returns is small. Therefore, in most of the existing methods, the IL problem is transforms into a distribution matching problem: $\arg\max_\pi \psi\left(d_{\pi_\mathrm{E}}, d_\pi\right)$, where ψ is a distance function.

A well-known imitation learning algorithm for this type of problem is generative adversarial imitation learning (GAIL). In GAIL, the Jensen-Shannon distance $\mathscr{D}_{\mathrm{JS}}$ is used to match the state-action occupancy measures. Therefore, in the maximum-entropy RL framework [27, 28], the optimization objective is:

$$\min_{\pi} \mathscr{D}_{\mathrm{JS}}\left(d_{\pi}, d_{\pi_{\mathrm{E}}}\right) - \lambda \mathscr{H}(\pi), \tag{1}$$

where $\mathscr{H}(\pi) := \mathbb{E}_{\pi}\left[-\log \pi(a|s)\right]$ is the γ-discounted causal entropy of policy π. It is treated as a policy regularizer controlled by $\lambda \geq 0$. This optimization objective draws a connection between imitation learning and generative adversarial networks (GANs) [29]. Therefore, GAIL handles this problem by finding a saddle point (π, D) of the expression:

$$\mathbb{E}_{\pi}\left[\log(D(s,a))\right] + \mathbb{E}_{\pi_{\mathrm{E}}}\left[\log(1 - D(s,a))\right] - \lambda \mathscr{H}(\pi),$$

where D is a discriminator, which is used to distinguish between the occupancy measure generated by π and the expert's occupancy measure.

While GAIL is a general framework for directly extracting a policy from data, it is not an inverse reinforcement learning (IRL) algorithm that seeks to recover reward functions. Therefore, the policy is a less transferable representation than the reward function. Adversarial inverse reinforcement learning (AIRL) [30] builds on the maximum causal entropy IRL framework [27]. Therefore, AIRL can be utilized to recover reward functions that are robust to changes in dynamics.

The optimization in AIRL is similar to that in GAIL, where the discriminator and generator are updated in an adversarial manner. The main difference between AIRL and GAIL is that the discriminator in AIRL has a particular form:

$$D_{\theta}(s,a) = \frac{\exp\left(f_{\theta}(s,a)\right)}{\exp\left(f_{\theta}(s,a)\right) + \pi(a|s)}, \tag{2}$$

where π is the learned policy, θ is the discriminator's parameter, and $f_{\theta}(s,a)$ is a learned function of the discriminator value.

2.4 Meta-adversarial inverse reinforcement learning

To enhance the adaptability the model, meta-AIRL [31] is implemented, in which meta-learning and adversarial inverse reinforcement learning are integrated to adapt to new tasks that have limited data samples. Meta-AIRL is composed of a discriminator and a generator. The discriminator takes single state-action pairs (s,a) as input at individual time steps, which has been proven to provide more stable learning. The generator represents the action policy of the agent whose optimization is based on a reward function. Therefore, the generator can generate trajectories that are as similar to the expert demonstrations as possible by maximizing the total rewards accumulated in an episode or up to the horizon. Meta-AIRL has a meta-training stage, an outer loop, and an inner loop. For the update in the inner loop, meta-AIRL uses an ADAM optimizer for the discriminator and trust region policy optimization (TRPO) [32] for the generator, which is consistent with AIRL and GAIL. The discriminator is used to provide reward signals to the generator. It is beneficial to use a high frequency for the discriminator and a low frequency for the generator. In the outer loop, meta-AIRL updates the model parameters with the average update information of all tasks. In addition, meta-AIRL leverages REPTILE [33] to train the generator and discriminator because it does not calculate any second derivatives.

We adopt this model for two reasons. On the one hand, imitation learning is generally data hungry. However, there are only a small amount of game records for each player. Therefore, we should more effectively learn the behavior of individual players. On the other hand, one thousand players have one thousand behavioral styles. This model can capture the commonality and variability of behaviors to adapt to new players.

3 Proposed model

In this section, we first present the problem statement, then elaborate on the related techniques described in Fig. 1. To quickly imitate the behavior of an individual player with limited gameplay records, we consider using meta-adversarial imitation learning. The imitation process of player behavior is divided into two stages: generic policy learning and individual player imitation. In each stage, the adversarial imitation learning method is used for model training on different data. These stages are combined through a meta-learning framework. Figure 3 shows the main flow of our method. There are two main steps: data processing and model training. Some details are described in the remainder of this section.

3.1 Problem statement

We consider the following problem:

Problem 1 Behavior imitation for an individual player.
Input: Total dataset $\mathscr{D}$.
Output: Imitation policy π_l for target player l.
Optimization objective: Maximize the similarity between π_l and the target player's real policy.

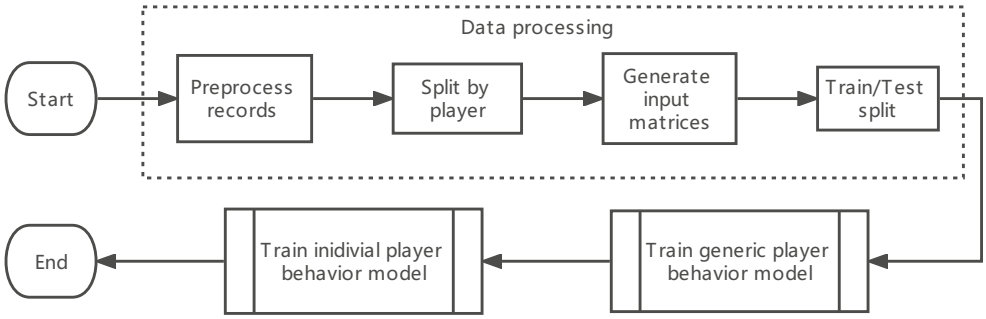


Fig. 3 The flowchart of our method. The training subprocesses are presented in Algorithm 1

Additional details of this problem are described below.

1. There are n players in Reversi. Each player's records constitute the total dataset $\mathbf{D}_i, i \in [1, 2, \cdots, n]$. The total dataset $\mathscr{D} = \cup_{i=1}^{n} \mathbf{D}_i$ consists of the game records of all players.
2. The dataset $\mathbf{D}_i$ is given by access to m-many trajectories of player i w.r.t the MDP, following a policy π_i.
3. We suppose that the policy of each player is relatively fixed.

3.2 Generic policy learning

In many games, we observe that players usually have a generic policy that is related to the environmental structure of the game. For a given game, players are usually reluctant to drastically change the generic policy to complete a task; instead, they minimally alter the generic policy to generate a specific policy with a personalized style to make a move. For example, in a racing game, the generic policy is to analyze the track situation and maintain the highest speed to achieve the highest ranking; in Reversi, the generic policy is to evaluate the positions of the game board and occupy the most favorable positions to obtain the most pieces.

Based on this observation, we use adversarial imitation learning to learn the generic policy π_{m} of multiple players in Reversi. The objective of the generator is to generate trajectories similar to multiple players' records. The discriminator attempts to distinguish between the generated trajectories and the actual trajectories. Similar to AIRL, we input single state-action pairs (s, a) at individual time steps to the discriminator. The cross-entropy loss and ADAM optimizer are used to update the discriminator.

Assuming the generic policy is parameterized by ϕ and the discriminator is parameterized by θ, the discriminator loss is given by:

$$L_D(\theta) = \mathbb{E}_{(s,a)\sim\pi_{\mathrm{m}}}\left[-\log D_\theta(s,a)\right] + \mathbb{E}_{(s,a)\sim\pi_\phi}\left[-\log\left(1 - D_\theta(s,a)\right)\right]. \quad (3)$$

Similarly, the reward function is built upon the outcome of the discriminator:

$$r(s,a) = \log\left(D_\theta(s,a)\right) - \log\left(1 - D_\theta(s,a)\right). \quad (4)$$

The discriminator and generator are connected. By scoring high values for the generated trajectories, the discriminator is "fooled". In this setting, the objective of the generator is equivalent to maximizing the total expected rewards in an episode or up to a horizon T:

$$\begin{aligned} &\arg\max_\phi \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T}\left(\log\left(D_\theta(s_t,a_t)\right) - \log\left(1 - D_\theta(s_t,a_t)\right)\right)\right] \\ &= \arg\max_\phi \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T}\left(f_\theta(s_t,a_t) - \log\pi_\phi(a_t|s_t)\right)\right]. \end{aligned} \quad (5)$$

Note that the optimizer of the generator in our method is a dueling double DQN (D3QN) [34] with prioritized experience replay [35], which is different from the AIRL or GAIL methods. D3QN is more suitable for Reversi games, where the behavior space is discrete. The training data are randomly sampled from the records of multiple players $\mathscr{D}$, thus improving the generality of the generic policy π_{m}.

3.3 Individual player imitation

Individual player imitation aims to learn players' personalized behaviors. Since different Reversi players have a similar policy, which is learned by the above process, we can quickly imitate individual behaviors. The behavior pattern of player i can also be expressed by policy π_i. Therefore, as with generic policy learning, we use adversarial imitation learning on the dataset of each player $\mathbf{D}_i$. The difference is that the parameters of the generator and discriminator are initialized to the parameters of the generic player behavior model.

Input: Datasets of n players $\mathscr{D} = \{\mathbf{D}_i\}_{i=1}^{n}$
Output: Policy parameter ϕ_l of players l // Target player for behavior imitation
1: Initiate generic policy parameter ϕ and meta discriminator parameter θ;
2: **for** iteration1 = 1, $\cdots$, M **do**
3: Sample training datasets $\mathscr{D}_r \subset \mathscr{D} \setminus \{\mathbf{D}_l\}$ st. $|\mathscr{D}_r| = N$ // Does not include the target player's dataset
4: **for** $\mathbf{D}_j \in \mathscr{D}_r$ **do**
5: **for** iteration2 = 1, 2, $\cdots$, K **do**
6: Calculate ϕ_j and θ_j using the adversarial imitation learning method;
7: **end for**
8: **end for**
9: Linearly decrease outer learning rates α_ϕ and α_θ;
10: $\phi \leftarrow \phi + \alpha_\phi \frac{1}{N-1} \sum_{i=1}^{N-1} (\phi_i - \phi)$; // Update policy
11: $\theta \leftarrow \theta + \alpha_\theta \frac{1}{N-1} \sum_{i=1}^{N-1} (\theta_i - \theta)$; // Update discriminator
12: **end for**
13: $\phi_l \leftarrow \phi$; // For the target player
14: $\theta_l \leftarrow \theta$;
15: **for** iteration1 = 1, 2, $\cdots$, H **do**
16: Update ϕ_l and θ_l using the adversarial imitation learning method on $\mathbf{D}_l$;
17: **end for**

Algorithm 1 Player behavior imitation

Algorithm 1 shows the main steps of player behavior imitation for Reversi. An initialization operation for the generic policy and meta discriminator is given on line 1. Lines 2 through 12 represent the stage of generic policy learning. From line 3, N players' datasets in the total dataset $\mathscr{D}$ are sampled for training. The generalization of the learned policy in this stage increases with the increase of N. However, the difficulty of learning also increases. To demonstrate the performance of this method, we exclude the target player's dataset from the training datasets. However, this rule is not a requirement. Lines 4 through 8 represent the inner loop of training. The adversarial imitation learning method is performed K times on each player's dataset. From line 9, the learning rates of the generic policy and the meta discriminator are decreased. The parameters of the model are updated, as shown in lines 10 and 11. The above training process is repeated M times.

Lines 13 through 17 represent the stage of individual player imitation. After generic policy learning, parameters of the generic policy and meta discriminator are independent of individual players. Instead, they are the initial parameters of the target player's model. The rest of the steps are similar to those of the inner loop of the previous stage. From line 16, the adversarial imitation learning method is used on the target player's dataset. This process repeated H times to eventually imitate the target player's behavior.

Table 2 shows the time complexity of Algorithm 1. Let the number of parameters in the network model be W. The time complexity of parameter initialization is $O(W)$. Accordingly, the time complexity of parameter updating is $O(NW)$. The time required by the adversarial imitation learning method depends on the optimizer of the generator. We denote the time complexity of one round of learning as $O(Q)$. Since $O(N) \ll O(Q)$ and $O(W) \ll O(Q)$, the total time complexity of this algorithm is $O(MNKQ + HQ)$.

The space requirement of our method mainly consists of two parts:

1. Storing the state-action pairs of all players. Suppose that each player has m state-action pairs and each state-action pair requires b bits to store. The space complexity of this part is $O(bmn)$.
2. Storing each $(s, a, r(s, a), s')$ in experience replay memory. Let the size of the memory be R. The space complexity of this part is $O(bR)$.

Thus, the total space complexity is $O(bmn + bR)$.

Table 3 shows the comparison of the complexity and actual consumption in terms of time and space required

Table 2 Time complexity of Algorithm 1

Line	Time complexity	Description
Line 1	$O(W)$	Initialize parameters
Line 3	$O(N)$	Sample training datasets
Line 6	$O(Q)$	Adversarial imitation learning
Line 9	$O(1)$	Decrease learning rates
Line 10-11	$O(NW)$	Update parameters
Line 13-14	$O(W)$	Parameter assignment
Line 16	$O(NW + Q)$	Learning and updating
Line 2-12	$O(MNKQ)$	
Line 13-17	$O(HQ)$	
Total	$O(MNKQ + HQ)$	

Table 3 Complexity and actual consumption time and space required to imitate a player

Method	Complexity		Consumption		
	Time	Space	Training time	Testing time	Space
Our method	$O(MNKQ + HQ)$	$O(bmn + bR)$	$\approx$ 3000s	$\approx$ 0.003s	$<$ 5GB
Neural network-based [36]	$O(lmW)$	$O(bm)$	$\approx$ 30s	$\approx$ 0.001s	$<$ 1GB
Random forest-based [21]	$O(kbm \log m)$	$O(bm)$	$\approx$ 3s	$\approx$ 2s	$<$ 1GB

to imitate a player using three methods. Let the number of training iterations be l. The time complexity of the neural network-based method is $O(lmW)$. Let the number of decision trees be k. The time complexity of the random forest-based method is $O(kbm \log m)$. In our experiments, the random forest-based method requires the least amount of training time, while our method requires the longest amount of training time. The main reason is that our method uses an adversarial imitation learning method, which requires more time to interact with the environment. To imitate a player, all methods need to store the player's state-action pairs. However, our method contains experience replay memory. Therefore, the state-action pairs of other players must also be stored. Therefore, the space complexity of our method is higher than others. This is not a major drawback since this space consumption is acceptable even for a personal computer.

3.4 Board state feature extraction

The board state of Reversi is relatively complex and changeable. Therefore, we use convolutional neural networks (CNNs) as the board feature extractor. Currently, CNNs are the leading machine learning tool for image analysis [37]. In our study, we customize the architecture of CNNs for Reversi.

Figure 4 shows the structure and function of the board feature extractor for Reversi. It takes the representation of the current board state, which is encoded in three 8x8 binary matrices, as input. Following recent research conducted on Reversi [6], we add a third matrix that marks the next legal moves to facilitate the selection of valid actions during game play. The state representation is retrieved by the feature maps in the layers of the CNN. Each unit in the feature map only receives data from its neighbors. Units within the same feature map share their weights. Then, the same local feature is calculated, albeit from a different part. This technology is suitable for extracting the spatial features of data. Hence, the board state features composed of the relations between pieces can be extracted layer by layer. This structure not only reduces the number of parameters but also makes the extracted features equivalent.

Here, more details of our network structure are presented. The network is composed of four convolutional layers and two fully connected (FC) layers. All feature maps in all layers have 8×8 units. Since the input is small, we pad it with zeros on the grid border to preserve its spatial resolution after convolution. For the same reason, we do not use any form of pooling. We use rectified linear units (ReLUs) to process convolution outcomes. The hidden layer consists of 128 units with ReLU activations. The output is a feature vector of size 64.

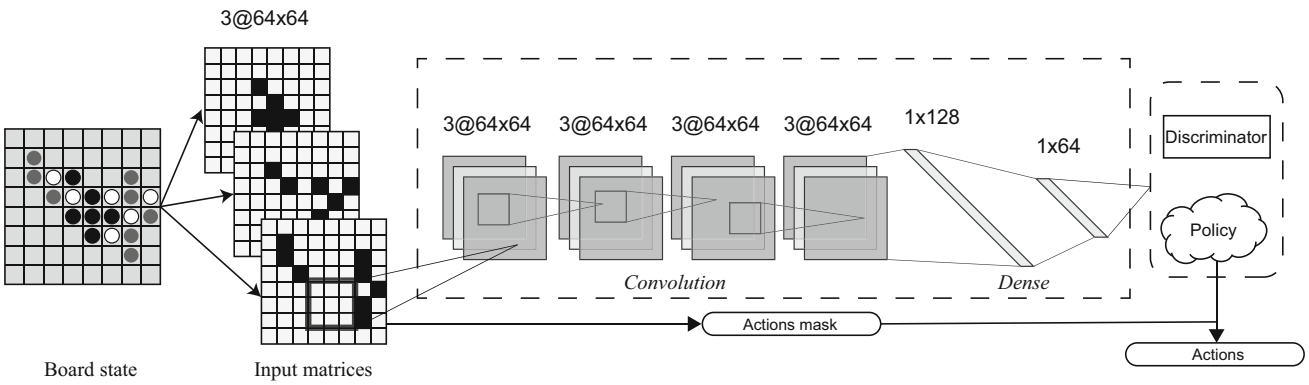


Fig. 4 The structure and function of the board feature extractor for Reversi. The board size is 8×8. The input of the board feature extractor is three 8×8 binary matrices. The first two matrices represent the piece locations of the player and their opponent. The legal moves mask is coded as the third matrix. The feature vector is obtained through four convolutional layers and two fully connected layers. This vector is the input of the generators (policies) and discriminators. When the feature vector is used to select actions, the third matrix multiples the generator's output to ensure the legality of actions

Our network cooperates with the generator and the discriminator in the player behavior imitation method [38, 39]. The generator and discriminator provide board state features for the policy function and the discriminant function, respectively [40, 41]. In the process of generic policy learning [42, 43], the feature extractors update. Their parameters are preserved in the process of individual player imitation, making the imitation process robust.

4 Experimental procedure

In this section, we evaluated our method on the game of Reversi. Experiments were conducted in AI vs. AI settings to learn the behavior of individual AI players. The behavioral styles of human players are susceptible to many factors, while the behavior preferences of AI players are relatively stable. Thus, we can generate large numbers of game records for method evaluation. The rest of this section details the evaluation measure and procedure of our experiments.

4.1 Measure of similarity

Currently, the research on behavior imitation is mainly focused on imitating human groups, with the purpose of making AI agents behave similarly to humans. The common evaluation indices of such research are believability and error [24]. The former is hard to quantify. The latter cannot be utilized to steadily evaluate the performance of individual player imitation. Therefore, we need a new measure to evaluate the similarity between the behavior of the imitation agent and the player from an individual perspective.

We propose using the learned imitation agents as behavior predictors to calculate the similarity of actions in the same state sets. Let $\mathbf{S} = \{s_1, s_2, \cdots, s_n\}$ and $\mathbf{A} = \{a_1, a_2, \cdots, a_n\}$ be the state set and action set of a player, respectively. Action similarity is defined as:

$$\mathscr{S}_{\text{act}} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(a_i, \pi(s_i)),$$
$$\mathbb{I}(a_i, \pi(s_i)) = \begin{cases} 1, & a_i = \pi(s_i); \\ 0, & a_i \neq \pi(s_i), \end{cases} \tag{6}$$

where π is the policy of the learned imitation agent.

We considered two issues while designing this measure. On the one hand, compared with believability, action similarity is more feasible and objective. On the other hand, the above equation can be used to calculate the policy similarity in identical states. These states are closely related to an individual's behavior. Therefore, this measure can steadily reflect the imitation effect of the imitation agent on an individual player.

4.2 AI player imitation

The primary focus of the experiments on AI players is to demonstrate that our method can learn robust individual player behavior models with limited records. These experiments are conducted in the game of Reversi using Botzone [44]. Botzone is a platform designed to evaluate implementations of different game AI agents by automatically scheduling matches for them using the ELO rating system [45]. This execution environment is limited with respect to memory and time, forcing users to write time-efficient and space-efficient programs to balance exploration and exploitation. There are a total of 453 Reversi bots. We select six different bots as players to be imitated.

Table 4 lists the information on the bots that we selected. These bots have different ranks due to the different algorithm frameworks implemented. The original data were generated based on almost twenty months of gameplay on Botzone. For each bot, 3,000 matches, each of which contains an average of 30 actions, were randomly selected, totaling approximately 90,000 actions. We generated the state-action pair datasets for each bot through simulations on playing Reversi. The data format is the three-dimensional matrix of $8 \times 8 \times 3$. The first matrix represents the position of the player's pieces, while the position of the opponent's pieces is stored in the second matrix. The action is represented as a vector with a length of 64. The mask of legal moves is the same size as the action to limit the range of actions.

The number of legal moves of Reversi depends on the state. To objectively evaluate the performance of our method, it is necessary to calculate the random accuracies

Table 4 Information on the Reversi bots and datasets

Bot name	Rank	Algorithm framework	Number of (s, a)	$\mathscr{S}_{\text{ran}}$
CAB	27	AlphaBeta	87539	17.12%
Shazam (SHA)	50	Monte Carlo tree search (MCTS)	87535	19.47%
Random (RAM)	58	Random search	87451	21.13%
ReversiBot_V1 (RVB)	61	Reinforcement learning	68640	17.54%
LearningBot (LEB)	65	Self-play with AlphaBeta pruning	86924	18.39%
ReversiTest (RVT)	70	AlphaBeta	87625	19.84%

used as the baseline of our experiments. Suppose that there are l_k legal moves in state s_k, the random action similarity of a dataset is calculated as follows:

$$\mathscr{S}_{\text{ran}} = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{l_k}, \tag{7}$$

where N is the number of state action pairs in the dataset. Table 4 shows the random action similarity of each bot on their corresponding dataset.

The datasets of selected bots were divided into training sets and test sets. To obtain imitation agents, we used Algorithm 1 on the training sets. The initial outer learning rates of discriminators and generators were set to 0.5 and 0.25, respectively. In addition, the frequency of the discriminator in each update was 20 times that of the generator. For generic policy learning, 9,000 iterations were completed. In the process of individual player imitation, we trained the individual player behavior model with 5,000 iterations. Finally, the similarities of the actions between imitation agents and bots on the testing sets were calculated.

5 Results

As shown in Table 5, our method has the ability to imitate different bots. The similarity of actions was evaluated using 5-fold cross-validation. As expected, very poor performance was achieved using the random baseline since random guessing is unlikely to imitate a player. The behavior of bot SHA and bot RVT are the most difficult to imitate. On the one hand, the imitation performance of these bots is related to their ranking. High-ranking bots have more complex behavior policies. On the other hand, imitation difficulty may be related to the MCTS algorithm used by bot SHA. MCTS is a type of heuristic random search algorithm with great uncertainty, resulting in a bot that is difficult to imitate.

The above experiment was conducted with a large number of player game records. We conducted another experiment to evaluate the imitation performance of our method for individual players with limited gameplay records. The experimental parameters and settings were the same as in the previous experiment.

Figure 5 displays the imitation results of different methods for different bots. Our method is compared with the baseline (dotted line) and two supervised methods [21, 36]. The supervised methods predict players' actions based on a neural network and random forest, respectively. For fair comparison, the neural network-based method is pre-trained before comparison. This operation can ensure that the total number of iterations is the same as the total number of iterations in our method. For each bot, we used 800 gameplay records as the test set to obtain the actions similarity of each method. From the results, we observe that:

1. There is a positive correlation between the similarity of actions and the number of gameplay records used for imitation. For these methods, a larger number of gameplay records results in player behavior characteristics that are easier to be captured.
2. The imitation effect of our method on these bots is different because the algorithms used by them are diverse. The action similarity for bot RVB is higher than that of the other five bots. On the one hand, the ranking of this bot is not high. On the other hand, its algorithm may be deterministic. Only tens of gameplay records need to be imitated for this bot. The behavior imitation for bot SHA is the most difficult. This phenomenon may be due to the lack of fixed behavior patterns.
3. When the number of gameplay records is small, our method performs much better than the supervised methods. However, the performance gap narrows as the number of records increases. In other words, supervised methods are more dependent on the amount of data for an individual player. Note that supervised methods only need tens of gameplay records to imitate bot RVB's behavior. This indicates that the policy of this bot is relatively simple. In contrast, even if 200 gameplay records are used, the behavior of bot SHA cannot be imitated using the supervised methods.
4. For all bots, better action similarity is achieved using our method than the baseline and supervised methods. Therefore, our method can faithfully imitate the player behavior with limited records for a challenging game environment.

Table 5 Similarity of actions by imitating each bot. Difference performances were obtained by our method to imitate different bots

Bot name	Rank	Baseline	Our method	Enhancement
CAB	27	17.12%	42.05%	24.93%
SHA	50	19.47%	35.44%	15.97%
RAM	58	21.13%	43.80%	22.67%
RVB	61	17.54%	59.68%	42.14%
LEB	65	19.39%	41.38%	21.99%
RVT	70	19.84%	33.20%	13.36%

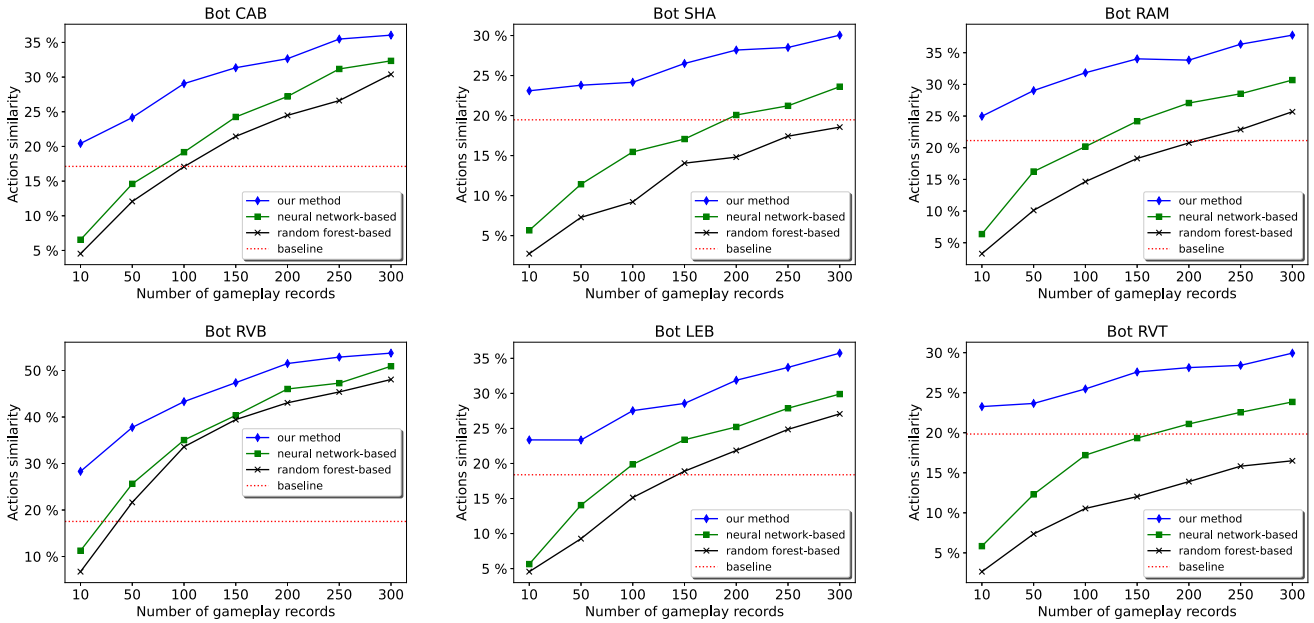


Fig. 5 Action similarity for each bot. We used different numbers of gameplay records to imitate the individual bot's behavior

Figure 6 shows the action similarity for each bot with different numbers of iterations. The abscissa represents the number of training iterations on the gameplay records for each bot. The number of iterations of generic policy learning is not included. In this experiment, 250 gameplay records were used to imitate each bot. From these results, we observe that:

1. Better action similarity of a bot can be achieved using our method without training on its gameplay records. The reason is that the generic player behavior model has been learned from other bots' records. This further supports the idea that players usually have a generic policy.
2. The action similarity value for our method increase with an increase of the number of iterations. This means that the generic policy is gradually transformed into the individual policy. Although the neural network-based methods can achieve high action similarity with very few iterations, more iterations may degrade the performance due to overfitting.

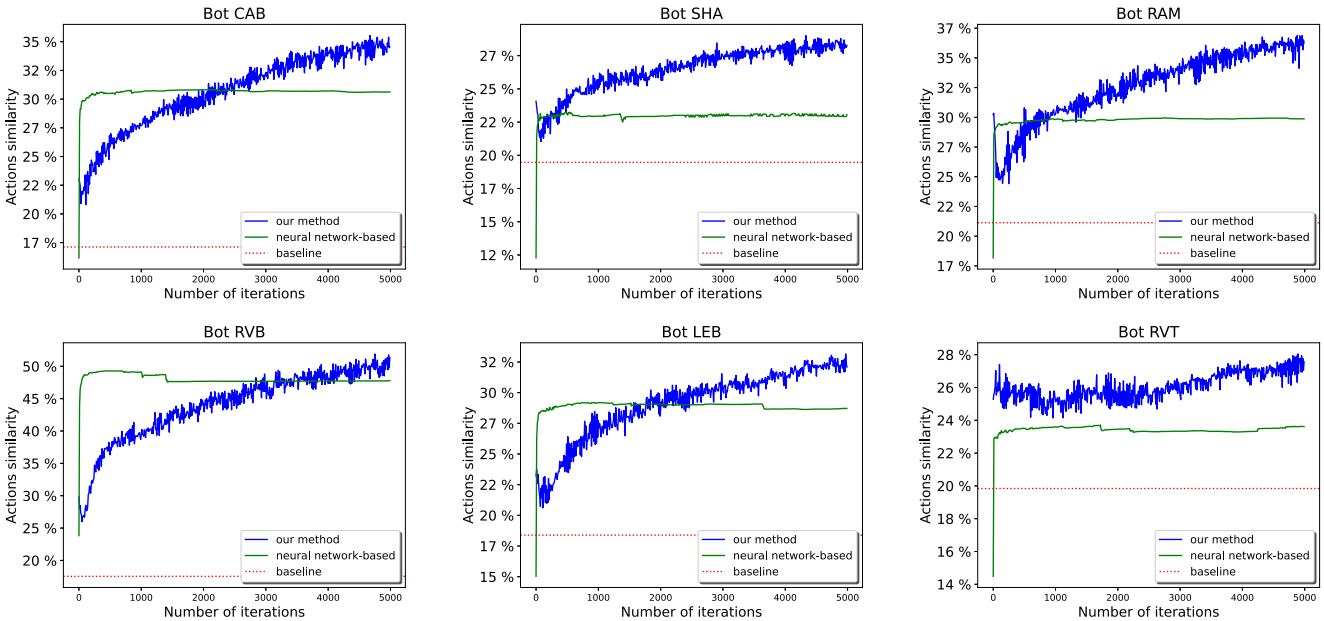


Fig. 6 Action similarity for each bot. Different iterations were run to imitate the individual bot's behavior

Table 6 Behavior imitation method for game players

Method	Game	Individual	Core technology	Number of records
Our method	Reversi	✓	Imitation learning	> 100
Player imitation [21]	StarCraft	✓	Recurrent neural networks	> 100
Maia [2]	Chess	✗	Convolutional neural networks	$> 10,000,000$
AI imitator [36]	Snake	✗	Convolutional neural networks	$> 1,000$
Move Prediction [8]	Hex	✗	Convolutional neural networks	$> 10,000$
RHEAOM [46]	Fighting game	✗	Evolution algorithm	–
Zoom model [47]	Zoom Poker	✗	Features design	$> 10,000$
Imitation for Shogi [48]	Shogi	✓	Generative adversarial networks	$> 1,000$
Hybrid model [49]	Gomoku	✗	Convolutional neural networks	$> 100,000$

3. When the number of iterations is within a given range, the neural network-based methods may achieve better action similarity than our method for some bots (e.g., bots CAB, RVB and LEB). However, our method eventually performs better. This phenomenon reflects the fact that our method is relatively slow, but has a strong ability to imitate individual players.

6 Discussions

Table 6 shows a comparison of behavior imitation methods for game players in the past five years. These methods are designed for different games. Most of these methods imitate the behavior of the players' group, rather than the behavior of individual players. Our method can imitate the behaviors of an individual player with hundreds of records, which is fewer than the requirement of most methods.

The experiments on the Reversi bots have provided some preliminary empirical evidence. Compared with the state-of-the-art methods, our method does not depend on a large amount of data. It performs better than other methods when the records of an individual player are limited. Although our method is relatively slow, time consumption is acceptable when the dataset is not very large. In fact, most players do not have a large number of records. Therefore, our method is more suitable for practical problems.

The ultimate goal of this research is to provide a behavior imitation method that is suitable for all players. Unfortunately, our method has a limited ability to imitate some bots. For instance, when a player's behavior is rather random, the imitation method tends to imitate his average behavior. In addition, the amount of data required for this method grows exponentially with the number of random factors [50], which significantly increases the difficulty of behavior imitation.

Even though there are many similarities in behavior patterns between human players and bots, the former is relatively more diverse. Our research on behavior imitation of human players is still insufficient. One of the main challenges of this research is how to create high-quality datasets of different level players. For instance, how should records of a player's abnormal behavior be discarded? In this context, it may be necessary to rely on supervised learning methods for classification. Therefore, as a first step in the next phase of this research, a data filter will be created. If the results of our method on the filtered datasets prove unsatisfactory (e.g., insufficient data/similarity), unused additional features of human players will be analyzed. For instance, we can pretrain board state feature extraction using the player's historical matchmaking information.

7 Conclusion

In this paper, we propose a method to integrate meta-learning and imitation learning to imitate the behaviors of individual players in a board game. This method processes the states of the game board through a CNN-based feature extractor. By using the generative adversarial approach to learn a generic policy on the gameplay records of multiple players, the individual player's behavior can be imitated with limited records. The effectiveness of the proposed model has been proven for the task of imitating the behaviors of Reversi AI players. Similarly, the evaluation shows that the imitation agent indeed learns the inherent features of different behaviors instead of just learning the average behavior of players.

There are some limitations of this work, which could be addressed in the future:

1. Lack of analyses of behavioral styles. It is necessary to understand to what extent different behavioral styles affect imitation and whether this influence can be increased in nonmeta-learning settings.
2. Lack of experiences using human players. We need to create high-quality datasets of different levels of human players and evaluate our method on them.

3. Lack of experiences on other games. Our experiments are conducted only on Reversi. However, our method can be extended to other similar games, such as Gobang, Go, and even real-time strategy games.
4. Insufficient study on the board state feature extractors. We will explore whether learning performance can be improved by other architectures for the board state feature extraction.

The imitation learning method under the framework of meta-learning will open new doors to AI design and training aids. Using Reversi as a model system, we have shown that imitating behavior at an individual level is possible. We hope this inspires the advancement of human imitation and AI collaboration systems in a variety of domains.

Acknowledgements This work is in part supported by the Central Government Funds of Guiding Local Scientific and Technological Development (No. 2021ZYD0003), the National Natural Science Foundation of China (No. 62006200), the Sichuan Province Youth Science and Technology Innovation Team (No. 2019JDTD0017), and the Nanchong Municipal Government-Universities Scientific Cooperation Project (No. SXHZ045).

References

1. Levesque HJ (2017) Common sense, the turing test, and the quest for real AI. mit press
2. McIlroy-Young R, Sen S, Kleinberg J, Anderson A (2020) Aligning superhuman AI with human behavior: chess as a model system. In: SIGKDD, pp 1677–1687
3. McIlroy-Young R, Wang Y, Sen S, Kleinberg J, Anderson A (2021) Detecting individual decision-making style: Exploring behavioral stylometry in chess. Adv Neural Inf Process Syst 34:24482–24497
4. Tian Y, Ma J, Gong Q, Sengupta S, Chen Z, Pinkerton J, Zitnick L (2019) ELF OpenGo: an analysis and open reimplementation of AlphaZero. In: PMLR, vol 97. pp 6244–6253
5. Lee CS, Tsai YL, Wang MH, Kuan WK, Ciou ZH, Kubota N (2020) AI-FML agent for robotic game of Go and AIoT real-world co-learning applications. In: FUZZ-IEEE, pp 1–8
6. Liskowski P, Jaśkowski W, Krawiec K (2018) Learning to play Othello with deep neural networks. IEEE Trans Games 10(4):354–364
7. Norelli A, Panconesi A (2022) OLIVAW: mastering Othello without human knowledge, nor a penny. IEEE Transactions on Games :1–1
8. Gao C, Hayward R, Müller M (2017) Move prediction using deep convolutional neural networks in Hex. IEEE Trans Games 10(4):336–343
9. Gao C, Müller M, Hayward R (2018) Three-head neural network architecture for Monte Carlo tree search. In: IJCAI, pp 3762–3768
10. Brown N, Sandholm T (2019) Superhuman AI for multiplayer poker. Science 365(6456):885–890
11. Li X, Miikkulainen R (2017) Evolving adaptive poker players for effective opponent exploitation. In: AAAI Workshops
12. Powley E, Cowling P, Whitehouse D (2017) Memory bounded Monte Carlo tree search. In: AIIDE, vol 13. pp 94–100
13. Baier H, Sattaur A, Powley EJ, Devlin S, Rollason J, Cowling PI (2019) Emulating human play in a leading mobile card game. IEEE Trans Games 11(4):386–395
14. Kurita M, Hoki K (2021) Method for constructing artificial intelligence player with abstractions to Markov decision processes in multiplayer game of Mahjong. IEEE Trans Games 13(1):99–110
15. Zheng Y, Li S (2020) A review of Mahjong AI research. In: ICRIC, pp 345–349
16. Smith JR, Joshi D, Huet B, Hsu WH, Cota J (2017) Harnessing A.I. for augmenting creativity: application to movie trailer creation. In: ACM-MM, pp 1799–1808
17. Ghimire A, Thapa S, Jha AK, Adhikari S, Kumar A (2020) Accelerating business growth with big data and artificial intelligence. In: I-SMAC, pp 441–448
18. Valle-Cruz D, Alejandro Ruvalcaba-Gomez E, Sandoval-Almazan R, Ignacio Criado J (2019) A review of artificial intelligence in government and its potential from a public policy perspective. In: DG.O, pp 91–99
19. Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, et al. (2020) Mastering Atari, Go, chess and shogi by planning with a learned model. Nature 588(7839):604–609
20. Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P, et al. (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575(7782):350–354
21. Partlan N, Madkour A, Jemmali C, Miller JA, Holmgård C, El-Nasr MS (2019) Player imitation for build actions in a real-time strategy game. In: AIIDE
22. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT press
23. Boucherie RJ, Van Dijk NM (2017) Markov decision processes in practice. Springer
24. Hussein A, Gaber MM, Elyan E, Jayne C (2017) Imitation learning: a survey of learning methods. ACM Comput Surv 50(2):1–35
25. Osa T, Pajarinen J, Neumann G, Bagnell JA, Abbeel P, Peters J, et al. (2018) An algorithmic perspective on imitation learning. Found Trends Robot 7(1-2):1–179
26. Fang B, Jia S, Guo D, Xu M, Wen S, Sun F (2019) Survey of imitation learning for robotic manipulation. Int J Intell Robot Appl 3(4):362–369
27. Wu Z, Sun L, Zhan W, Yang C, Tomizuka M (2020) Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. IEEE Robot Autom Lett 5(4):5355–5362
28. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: PMLR, vol 80. pp 1861–1870
29. Song J, Ren H, Sadigh D, Ermon S (2018) Multi-agent generative adversarial imitation learning. Adv Neural Inf Process Syst :31
30. Yu L, Song J, Ermon S (2019) Multi-agent adversarial inverse reinforcement learning. In: PMLR, vol 97. pp 7194–7201
31. Zhou A, Jang E, Kappler D, Herzog A, Khansari M, Wohlhart P, Bai Y, Kalakrishnan M, Levine S, Finn C (2019) Watch, try, learn: meta-learning from demonstrations and rewards. In: ICLR
32. Shani L, Efroni Y, Mannor S (2020) Adaptive trust region policy optimization: global convergence and faster rates for regularized mdps. In: AAAI, vol 34. pp 5668–5675
33. Huisman M, Van Rijn JN, Plaat A (2021) A survey of deep meta-learning. Artif Intell Rev 54(6):4483–4541
34. Sewak M (2019) Deep Q Network (DQN), Double DQN, and Dueling DQN, Springer, pp 95–108
35. Tao X, Hafid AS (2020) Deepsensing: a novel mobile crowdsensing framework with double deep q-network and prioritized experience replay. IEEE Internet Things J 7(12):11547–11558

36. Zhou Y, Li W (2020) Discovering of game AIs' characters using a neural network based AI imitator for AI clustering. In: IEEE CIG. vol 2020-Augus. pp 198–205
37. Aloysius N, Geetha M (2017) A review on deep convolutional neural networks. In: ICCSP, pp 0588–0592
38. Irfan A, Zafar A, Hassan S (2019) Evolving levels for general games using deep convolutional generative adversarial networks. In: CEEC, pp 96–101
39. Pfau J, Liapis A, Volkmar G, Yannakakis GN, Malaka R (2020) Dungeons & replicants: automated game balancing via deep player behavior modeling. In: IEEE CoG, pp 431–438
40. Wang K, Gou C, Duan Y, Lin Y, Zheng X, Wang FY (2017) Generative adversarial networks: introduction and outlook. IEEE/CAA J Autom Sin 4(4):588–598
41. Aggarwal A, Mittal M, Battineni G (2021) Generative adversarial network: an overview of theory and applications. Int J Inf Manag Data Insights 1(1):100004–0
42. Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML, vol 70. pp 1126–1135
43. Finn C, Yu T, Zhang T, Abbeel P, Levine S (2017) One-shot visual imitation learning via meta-learning. In: CoRL, vol 78. pp 357–368
44. Zhou H, Zhang H, Zhou Y, Wang X, Li W (2018) Botzone: an online multi-agent competitive platform for AI education. In: ITiCSE, pp 33–38
45. Kovalchik S (2020) Extension of the elo rating system to margin of victory. Int J Forecast 36(4):1329–1341
46. Tang Z, Zhu Y, Zhao D, Lucas SM (2020) Enhanced rolling horizon evolution algorithm with opponent model learning. IEEE Trans Games :1–1
47. Carneiro MG, De Lisboa GA (2018) What's the next move? Learning player strategies in Zoom Poker games. In: CEC, pp 1–8
48. Wan S, Kaneko T (2017) Imitation learning for playing Shogi based on generative adversarial networks. In: TAAI, pp 92–95
49. Yan P, Feng Y (2018) A hybrid gomoku deep learning artificial intelligence. In: AICCC, pp 48–52
50. Laskin M, Lee K, Stooke A, Pinto L, Abbeel P, Srinivas A (2020) Reinforcement learning with augmented data. Adv Neural Inf Process Syst 33:19884–19895

Chao-Fan Pan received the B.S. degree from the School of Information, Southwest Petroleum University, Chengdu, China, in 2020. He is currently pursuing the postgraduate degree with the School of Computer Science, Southwest Petroleum University, Chengdu, China. His current research interests include reinforcement learning and deep learning.

Xue-Yang Min is a graduate student with School of Sciences, Southwest Petroleum University, Chengdu, China. Her current research interests include multi-label learning and cost-sensitive learning.

Heng-Ru Zhang received the M.S. degree from the School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2002, and the Ph.D. degree from the School of Sciences, Southwest Petroleum University, Chengdu, in 2019. He is currently a professor with Southwest Petroleum University, Chengdu. He has published more than 30 refereed papers in various journals and conferences, including Applied Soft Computing, Information Sciences, and Knowledge-Based Systems. His current research interests include recommender systems, label distribution learning and granular computing.

Guojie Song received the M.S. and Ph.D. degrees from the Department of mathematical science, Tsinghua University, Beijing, China, in 2008 and 2011, respectively. He is currently a professor with Southwest Petroleum University, Chengdu. He has published more than 50 refereed papers in various journals and conferences, including the Geophysics, BSSA, et. al. His current research interests include forward modeling and inversion of seismic wave, deep learning.

Fan Min (Member, EEE) received the M.S. and Ph.D. degrees from the School of Computer Science and Engineering, University of Electronics Science and Technology of China, Chengdu, China, in 2000 and 2003, respectively. He visited the University of Vermont, Burlington, Vermont, from 2008 to 2009. He is currently a Professor with Southwest Petroleum University, Chengdu. He has published more than 100 refereed papers in various journals and conferences, including the IEEE Transactions on Systems, Man, and Cybernetics: Systems, Information Sciences, International Journal of Approximate Reasoning, and Knowledge-Based Systems. His current research interests include recommender systems, active learning, granular computing and multiple instance learning.