Journal of the Korean Physical Society (2022) 81:1189–1197
https://doi.org/10.1007/s40042-022-00622-8

Online ISSN 1976-8524
Print ISSN 0374-4884

ORIGINAL PAPER - GENERAL, MATHEMATICAL AND STATISTICAL PHYSICS

A statistical approach for detecting AI-assisted cheating in the game of Go

Junyeong Park[1,5] · Jungyu Im[2] · Sojin On[3,5] · Sung Jong Lee[4] · Jooyoung Lee[2,5]

Received: 24 June 2022 / Revised: 2 August 2022 / Accepted: 22 August 2022 / Published online: 31 October 2022

Abstract
Since the AlphaGo Zero paper was published, many superhuman Go AIs followed, some with open sources and easy-to-use interfaces. Currently, it is impossible even for the best human Go player to beat such an AI on an even game. One of the drawbacks of the freely available and easy-to-use superhuman Go AI is that one can be easily tempted to become an invincible Go player by copying AI moves in a secretive manner, especially in an online game. In reality, such AI-assisted cheating has been reported in a multiple cases causing a new kind of social problem. By comparing the calculated similarity against expected similarity distribution obtained from face-to-face games among professional Go players, we obtained the probability that a game record is solely the result of a human play unaided by any AI tools. Rather than simply estimating the AI-likeness rate of a play, we provide a numerical basis for AI-assisted cheating by calculating the probability that a certain value of AI-likeness rate will occur in a professional game. We apply our method to a recent alleged game of AI-assisted cheating, and find that the AI-likeness rate is quite high and the chance of happening is as low as one in 1,000,000 games, if it were played by a professional player.

Keywords Baduk · AI-cheating · Statistics · reliability

1 Introduction

The game of Go, also known as Baduk or Weiqi, had been a long-standing daunting challenge for an artificial intelligence (AI) to compete against a human professional player until the historical game between Lee Sedol and AlphaGo in 2016 [1]. Prior to 2016, many approaches contributed to the improvement of the AI Go machine, and they include Monte Carlo tree search (MCTS) [2] and pattern recognition to name a few [3]. However, these algorithms alone were not good enough to compete against professional Go players without a considerable amount of handicaps. AlphaGo [4], which combined deep neural networks and MCTS, served as the first AI Go machine to beat human professional players under the equal condition. Subsequently, more powerful versions of AlphaGo followed, namely, AlphaGo Zero [5] and AlphaZero [6]. Currently, multiple open source AI Go machines [7–9] developed in a similar fashion are freely available to the general public. These superhuman variants of AI Go machines caught the attention of many people, especially because they come with an easy-to-use graphic user interface such as Lizzie [10], Sabaki [11], and LizzieYZY [12].

The quantitative estimate of the superhuman nature of a strong AI Go machine over a top-ranked professional Go player can be an arguable subject in the Go community mainly due to the lack of systematic experiments. One arguable estimate indicates that the gap between an AI and a professional player can be at least as large as two-stone handicap without komi, equivalent to about 19 territorial points [13]. This huge gap, along with the availability of multiple easy-to-use AI Go machines, prompted many human Go players under the temptation of cheating a Go

✉ Jooyoung Lee
jlee@kias.re.kr

[1] Department of Computer Science, Hanyang University, Seoul 04763, Republic of Korea

[2] Good Intelligence Co., Ltd., Seoul 04701, Republic of Korea

[3] Department of Baduk Studies, Myongji University, Gyonggi-do 17058, Republic of Korea

[4] Research Institute for Basic Sciences, Changwon National University, Changwon-Si 51140, Republic of Korea

[5] School of Computational Sciences, Korea Institute for Advanced Study, Seoul 02455, Republic of Korea

game (especially on an on-line game) against the opponent by copying moves suggested by an AI. Multiple reported cases of AI-assisted cheating in the on-line as well as off-line game of Go [14, 15] caused a social problem, disturbing trust between game players. Generally, it is hard to obtain direct evidence of cheating especially from on-line games, and, for this reason, there exist some needs to develop a method for detecting AI-assisted cheating in the game of Go. One such method would be to install surveillance instruments to make an on-line game effectively a face-to-face off-line game. But currently this kind of approach is not feasible for a large number of competition games happening simultaneously. Here, we propose a statistical approach for detecting AI-assisted cheating by reviewing a game record, which, we hope, will discourage people from the temptation of AI-assisted cheating.

In this paper, we describe the AI-assisted cheating detection method in which numerical information about the possibility of cheating is calculated through statistical estimation. Basically, we check the similarity between the play of a recorded game and the suggested moves according to our Go playing program, BaduGI. Then, the calculated similarity is examined against the expected similarity distribution of top-ranked professional Go players. Here, we name the similarity as the AI-likeness rate (AILR), and we assume that AILR of professional players follows a normal distribution. We made a statistical model of AILR from 398 games of 2019–2020 KB Baduk league [16]. Because the 2019–2020 KB Baduk league was an on-site event, we expect that thus-generated statistical model will represent AILR of professional Go players playing without any assistance of AI. The p-value of a given AILR value can be calculated from the statistical model, which represents the probability of happening by chance.

This paper is organized as follows: In Sect. 2, we describe how we analyzed Go games and detected AI-assisted cheating. In Sect. 3, we describe the results of analysis applied to actual game records. Conclusions and discussions are made in Sect. 4.

2 Methods

2.1 BaduGI

BaduGI is developed based on the AlphaGo Zero [5] framework that combines deep neural network with MCTS. In BaduGI, there are some modifications in the neural network architecture and the tree search algorithm. BaduGI uses squeeze-and-excitation network (SENet) [17] instead of ResNet [18] used in AlphaGo Zero. The neural network of BaduGI predicts policy and value as in AlphaGo Zero, and additionally it predicts the territorial ownership of each of the 19×19 lattice points, and the final territory difference. The policy is a conditional probability of an action a for a given state s. The value is the expectation of the game result, win or loss. The territory difference prediction refers to the final territorial difference between White and Black.

In AlphaGo Zero, each tree node (s, a) stores a set of statistics, $\{N(s,a), W(s,a), P(s,a)\}$, where $N(s, a)$ is the visit count; the number of investigations of action a in state s, $W(s, a)$ is the total action value, and $P(s, a)$ is the prior probability of selecting action a in state s. $P(s, a)$ is the probability of selecting an action a for a given state s and it represents the frequency of the action a during training. The average action value, $Q(s, a)$, can be calculated by $Q(s,a) = W(s,a)/N(s,a)$.

In BaduGI, each tree node (s, a) stores one additional value: the balanced value $B(s, a)$. The balanced value is designed to quickly reflect the feedback of the value change obtained during the last stage of simulation to the tree search. The update rule of the balanced value is as follows:

$$B(s,a) \leftarrow \frac{N_{\text{bal}} - 1}{N_{\text{bal}}} \times B(s,a) + \frac{1}{N_{\text{bal}}} \times v, \tag{1}$$

where v is the new value obtained from the last simulation event and N_{bal} is a parameter controlling the reflection speed. In our experiments, N_{bal} is set to 1999. When $N(s,a) < N_{\text{bal}}$, $B(s, a)$ is set to $Q(s, a)$ to minimize unintended fluctuation.

After the simulation is finished, BaduGI selects an action a to play in the root state s_0 that maximizes the score $S(s_0, a)$,

$$S(s_0,a) = (1-\lambda)Q(s_0,a) + \lambda B(s_0,a) + \gamma \log N(s_0,a), \tag{2}$$

where λ is a weighting parameter that mixes the average action value and the balanced value, and γ is a parameter reflecting the visit count to the score. In our experiments, λ is set to 0.2 and γ is set to 0.05. We define the best move, a^*, as the action a that maximizes Eq. 2.

2.2 AI moves versus human moves

We analyzed a given Go game either by executing a total of 10,000 simulation counts for each move or by the maximum visit count reaching 10,000 for a visited move, whichever was met first. For each s_0, we collected $P(s_0,a)$, $Q(s_0,a)$, and $S(s_0,a)$ for all top-scoring moves of BaduGI and the actual move played by the human player.

We performed the analysis by running 5 separate BaduGI simulations for each game using different random numbers. To properly discriminate AI-assisted moves against true human moves, first, we need to understand potential variation in AI-assisted moves arising from the choice of an AI engine, the total simulation count per move, stochastic nature of MCTS, and the multiplicity of good AI moves for a given board situation.

Indeed, from 5 BaduGI simulation runs, we observed variation in a^* for about 28% of the board positions examined. Therefore, we expanded the definition of an AI-like move to include all top-scoring moves satisfying Eq. 3 below. In this study, we assume that a move a in state s_0 is AI-like, or potentially copied by a cheating human player from a set of AI's recommended moves if and only if

$$S(s_0, a^*) - S(s_0, a) \leq S_{\mathrm{diff}}, \tag{3}$$

where the parameter S_{diff} is large enough to accommodate the variation arising from various AI engines, the total simulation count, and the board position. In addition, S_{diff} should be small enough not to include too many alternative moves. To determine an appropriate value of S_{diff}, we examined the variation of a^* in five BaduGI runs as follows.

In Fig. 1a, we show the score difference between a^* and the N-th best move for a given s_0 of a given BaduGI run, where N is the number of different identities of a^* from 5 BaduGI runs. As mentioned above, in 72% of the cases $N = 1$, and we excluded these board positions in Fig. 1a. For example, from 5 simulation results for a given s_0, there may exist two quite efficient moves, a_1 and a_2, with highly competing relative scores to each other, where, in three cases, $S(s_0, a_1) > S(s_0, a_2)$, and in the other two cases $S(s_0, a_1) < S(s_0, a_2)$. For this case, $N = 2$, and both a_1 and a_2 should be classified as AI-like moves, and this can be achieved if S_{diff} is set to be larger than $|S(s_0, a_1) - S(s_0, a_2)|$ for all 5 simulations.

In Fig. 1b, we show the score difference between a^* and the $(N + 1)$-th best move. For the example above with $N = 2$, the 3rd best move a_3 should be discriminated by S_{diff} in such a way that $S(s_0, a_{1,2}) - S(s_0, a_3) > S_{\mathrm{diff}}$ for all 5 simulations. Ideally, we want to set the value of S_{diff} to maximize the cases shown as A in Fig. 1a while minimizing the cases shown as α in Fig. 1b. Here, for the calculation of AI-likeness, assuming all a^* observed in 5 BaduGI runs are AI-like moves, the precision for AI-likeness can be referred to as $P = A/(A + \alpha)$, and the recall for AI-likeness as $R = A/(A + B)$. For a proper balance between P and R, we used F_β score with more emphasis on P with $\beta = 0.5$ as follows.

$$F_\beta = \frac{(1 + \beta^2)PR}{\beta^2 P + R}. \tag{4}$$

The rationale for $\beta = 0.5$, less than unity, is to minimize the case of misclassification of a non AI-like move to be included as an AI-like move. The $F_{0.5}$ score obtained using Fig. 1a, b is shown in Fig. 2 where maximum is located at $score_{\mathrm{diff}} = 0.04125$. In this study, for each BaduGI simulation, a^* and additional action a satisfying Eq. 3 are defined as AI-like moves.

Some of AI-like moves agree well with the play of many top-ranked professional Go players, especially for a board position where deviation from a one-way sequence

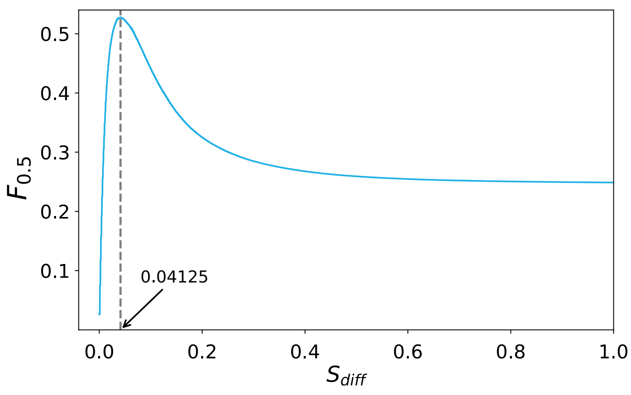

Fig. 2 $F_{0.5}$ is plotted as the function of S_{diff}. We set the value of $S_{\mathrm{diff}} = 0.04125$ to maximize $F_{0.5}$

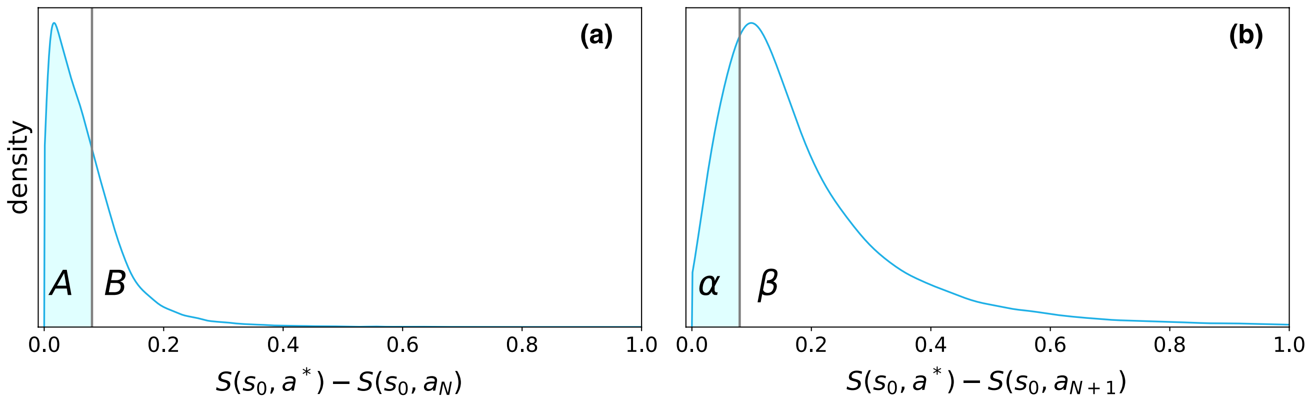

Fig. 1 (a) Distribution of the score difference between $S(s_0, a^*)$ and $S(s_0, a_N)$ is shown. Here, N is the number of different identities of a^* from 5 BaduGI runs. (b) Distribution of $S(s_0, a^*)$ and $S(s_0, a_{N+1})$ is shown. A and α represents the area under each curve corresponding between 0 and S_{diff}. We want to set the value of S_{diff} to maximize A while minimizing α (see Eq. 4)

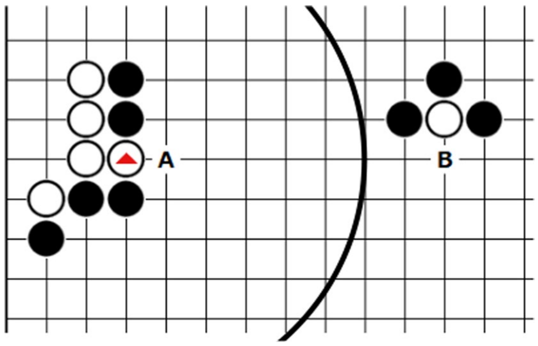

Fig. 3 The white stone marked by a triangle is the last move played. All positions enclosed by and on the circle of radius of 6 lattice units represent the vicinity of the marked move. Here, p_{thres} for move A and B are set to 0.4 and 0.8, respectively

of moves can lead to an outright defeat of a game. In order to screen out these situations, we excluded the board position s where $P(s, a^*) > p_{\text{thres}}$. The parameter p_{thres} is set to 0.4, if a^* is located in the vicinity of the last opponent's move. Otherwise, p_{thres} is set to 0.8. The vicinity is defined as the Euclidean distance between two positions being 6 lattice units or less (Fig. 3).

When the winning probability is either too high or too low, there exist many actions of similar winning probability values, and contribution of a particular move is not so much discriminating compared to other moves, having little effect on the game result. Therefore, we excluded all moves with $Q(s_0, a) > 0.95$ or $Q(s_0, a) < 0.05$ where a is the actual move played in our analysis. Finally, we excluded the first 20 moves of a game in our analysis. This is due to the fact that, currently, many professional Go players are quite familiar with the opening moves of AIs, and the similarity between human moves and AI-like moves is usually quite high making the discrimination of true human play from AI-assisted human play difficult to detect in terms of statistics.

We divided the game into three stages. The early stage is from the 21st to the 50th move. The middle stage is from the 51st to the $(T-50)$th move, and the last stage is from the $(T-49)$th to the Tth move, where T is the total number of moves. Statistics for AI-likeness were collected separately for each stage as well as the early and middle stages. Let the number of moves included in the analysis in the early, middle stage, and the late (last) stages be N_{c1}, N_{c2}, and N_{c3}, respectively. And let the number of AI-like moves in the early, middle stage, and the late stages be N_{b1}, N_{b2}, and N_{b3}, respectively. Then, the number of moves included in the analysis is $N_c = \sum_{i=1}^{2} N_{ci}$, and the number of AI-like moves is $N_b = \sum_{i=1}^{2} N_{bi}$. Finally, the AI-likeness in the early, middle, and the late stages are $R_i = N_{bi}/N_{ci}$ for $i = 1, 2, 3$, and the overall AI-likeness rate is $R_t = N_b/N_c$.

To calculate the average AI-likeness rate of ranked professional Go players, we used a total of 398 games of 2019–2020 KB Baduk league, organized by Korea Baduk Association. The 398 games include all 360 regular season games and 38 post season games. To calculate the AI-likeness rate of a game, we performed five independent BaduGI simulations resulting in five values. Hence, we obtained a total of 3,980 values of the AI-likeness rate. The distribution of the AI-likeness rate is assumed to be a normal distribution as shown in Fig. 4. Assuming the normal distribution for the AI-likeness rate of the early and middle stages, one can calculate the average value of the AI-likeness rate, μ, and the standard deviation, σ. The z-score Z is defined as the deviation of a value from the mean μ in the unit of σ as,

$$Z = \frac{R_t - \mu}{\sigma}. \tag{5}$$

One can calculate the probability of observing a particular value of the AI-likeness rate with z-score Z as,

$$p\text{−value} = 1 - \Phi(|Z|), \tag{6}$$

where $\Phi(x)$ is the cumulative density function of normal distribution.

3 Results

3.1 Analysis of the 2019–2020 KB Baduk league

In Fig. 5, we show the distribution of the AI-likeness rate for the early and middle stages as well as 3 separate stages of the $796 = 398 \times 2$ game plays. Each AI-likeness rate is the average of 5 BaduGI simulations. The mean value of AI-likeness rate of the early and middle stages is about 0.286, and the

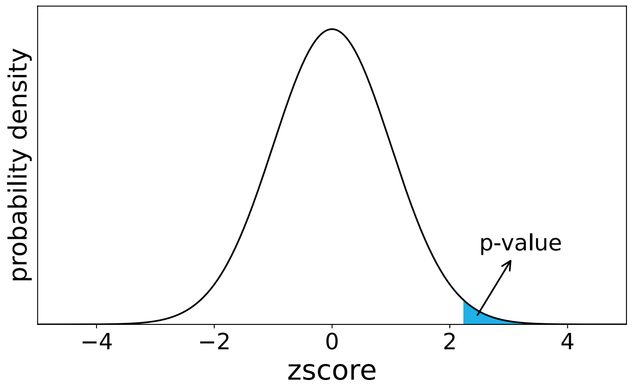

Fig. 4 The probability of AILR = 0.50 (corresponding to z-score = 2.239) happening by chance is calculated by the p-value shown in the figure. The calculated p-value = 0.013 represents that this game play is likely to happen in one out of about 80 games in the 2019–2020 KB Baduk league

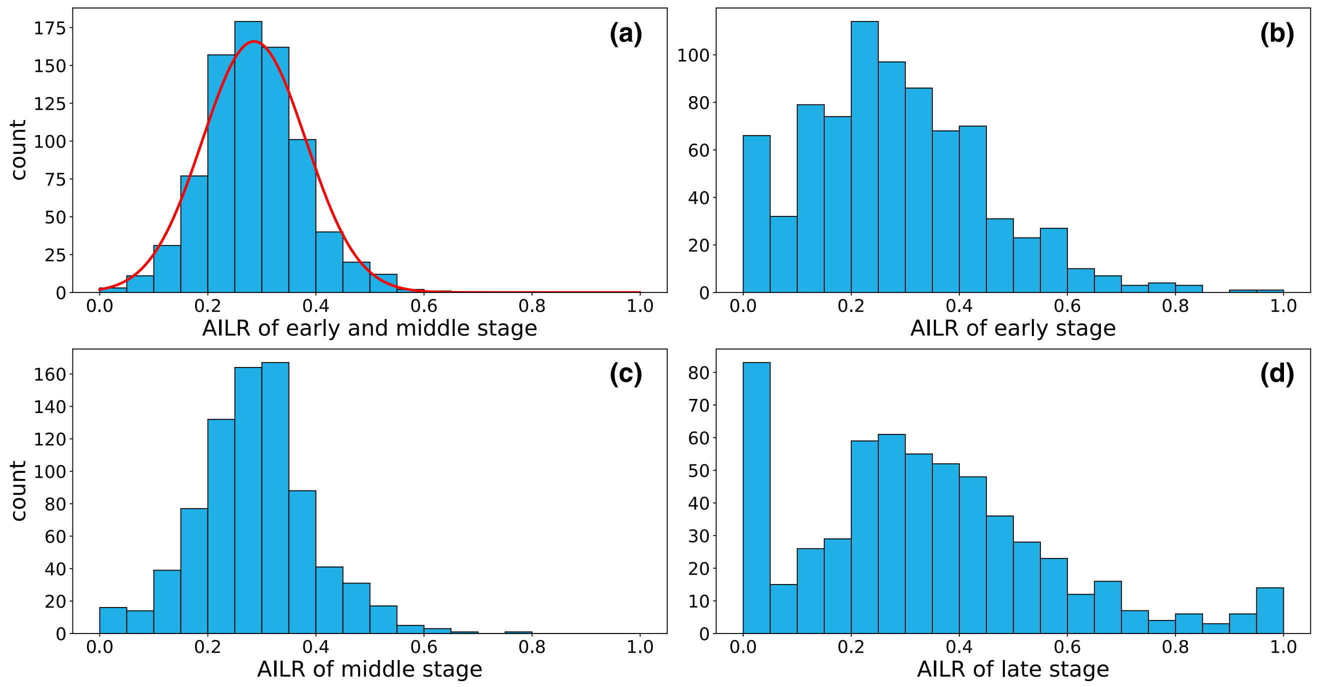

Fig. 5 The histogram of AI-likeness rate (AILR) is shown for the early and middle stages (a), the early stage (b), the middle stage (c), and the late stage (d). We observe that the distribution of AILR of the early and middle stages is not far from a normal distribution shown in (a). We used the AILR of early and middle stages as the criterion to detect the AI-assisted cheating. The cases of very small and large values of AILR in the late stage are the artifact of small number of evaluation cases

standard deviation is about 0.096. We observe relatively high AILR values from the late stage as shown in Fig. 5d, and we decided to use the AILR value of the combined stage of early and middle as the criterion to detect the AI-assisted cheating. From here on, we use the terminology of AILR as the AILR of the early and middle stages.

The highest value of AI-likeness rate is 0.603 played by Shin Jinseo in his second game of the playoff round in the postseason against Kang Seungmin. The z-score is about 3.314, and p-value is about 0.00046, representing the probability of happening once in about 2200 plays of the league. Considering that there are a total of 796 plays in the 2019–2020 KB Baduk league, it is not so much unexpected that this low value of p-value is actually observed. We note that the lowest value of AI-likeness rate is 0.00.

We observe a strong correlation between the AI-likeness rate and the game result. In Fig. 6, we show the game result of the top 100 AI-likeness rate plays ($\text{AILR} \geq 0.384$), where 82/18 games are won/lost. On the other hand, when we examine the bottom 100 AI-likeness rate plays ($\text{AILR} \leq 0.185$), where 33/67 games are won/lost. In Fig. 7 and Table 1, we show the statistics of 45 professional players who played at least 10 games in the 2019–2020 KB Baduk league. We observe that the pearson correlation coefficient between the winning rate and the AI-likeness rate is about 0.674, representing that, on average, the player with a higher

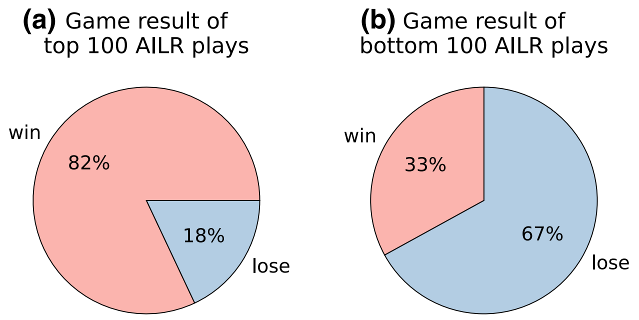

Fig. 6 Game results are analyzed according to the AI-likeness rate (AILR) of each play, by White and by Black. Out of $2 \times 398 = 796$ game plays from the 398 2019–2020 KB Baduk league games, top (a)/bottom (b) 100 AILR plays are examined for their game results, win or lose. Here, each game play represents a set of moves played by a player. We observed that higher AILR game plays tend to result in more wins on average

AILR is more likely to end up with a higher winning rate. However, some players, like Lee Wondo, Park Junghwan, Choi Jeong, Park Seunghwa and Lee Changho have relatively low AI-likeness rates, compared to their winning rates. These players may improve their winning rates by attempting to increase their AILRs. Taken together, the AI-likeness rate is highly related to the winning rate, but it is not an absolute measure to decide the game result.

Table 1 The results of game plays are summarized in terms of 45 ranked Korean Go players who have played at least 10 games in the 2019–2020 KB Baduk league

Name	Winning rate	AI-likeness rate	Name	Winning rate	AI-likeness rate
Shin Jinseo	0.957	0.376	Paek Hongseok	0.667	0.283
Byun Sangil	0.789	0.370	Choi Jaeyoung	0.500	0.282
Jin Siyoung	0.667	0.326	Lee Yeongkyu	0.529	0.282
Seol Hyunjun	0.533	0.313	Na Hyun	0.375	0.281
Kim Jiseok	0.625	0.312	Moon Yubin	0.500	0.279
Lee Donghoon	0.769	0.311	Park Yeonghun	0.667	0.278
Lee Jihyun	0.625	0.310	Yun Chanhee	0.214	0.278
Choi Cheolhan	0.722	0.308	Park Sangjin	0.438	0.275
Shin Minjun	0.789	0.304	An Jungki	0.313	0.271
Yun Junsang	0.647	0.304	Han Seungjoo	0.529	0.271
Kim Myounghoon	0.500	0.303	Song Jihoon	0.438	0.270
Jeong Seojun	0.333	0.301	Park Jinsol	0.375	0.269
Weon Seongjin	0.438	0.293	Kang Seungmin	0.444	0.268
Han Taehee	0.500	0.293	Han Sanghoon	0.176	0.264
Heo Yongho	0.588	0.292	Song Gyusang	0.200	0.262
Kang Dongyun	0.765	0.291	Lee Hoseong	0.538	0.258
Park Geunho	0.429	0.290	Song Taekon	0.100	0.255
Lee Changseok	0.350	0.290	Park Seunghwa	0.563	0.249
Cho Hanseung	0.571	0.289	Choi Jeong	0.476	0.244
Hong Seongji	0.563	0.288	Park Jonghoon	0.357	0.242
Park Hamin	0.579	0.288	Lee Wondo	0.267	0.228
Sim Jaeik	0.357	0.288	Lee Changho	0.273	0.226
Park Junghwan	0.813	0.285			

The winning rate is the ratio of the games won and the total games played. The AI-likeness rate of an individual player is first calculated using the entire game play for each game, and then averaged for the total number of games played by the player. The scatter plot of winning rate vs. the AI-likeness rate of an individual player can be found in Fig. 7

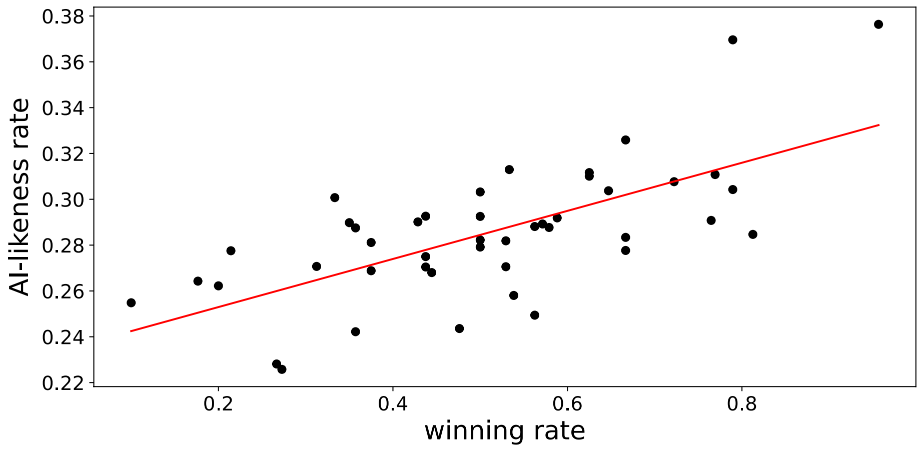

Fig. 7 AI-likeness rate vs. the winning rate calculated for each player who has played more than 10 games are shown. The line shows the linear regression with the pearson correlation 0.674. Each data point is also shown in Table 1. We observe a positive correlation between AILR and the winning rate

3.2 Analysis of an alleged game suspected of AI intervention

Using our method, we analyzed a disputed game played in an on-line tournament [14]. The result is summarized in Table 2.

Two players played a total of 129 moves in total in the on-line tournament. In this game, the early stage is from the 21st to the 50th move, the middle stage is from the 51st to the 79th move, and the last stage is from the 80th to the 129th move. On average, 10.8 moves of Player B in the early stage, no moves in the middle stage, and 0.8 moves in the last stage were analyzed to estimate the AI-likeness rate. Similarly, 8.6 moves of Player A in the early stage, no moves in the middle stage, and 1.2 moves in the last stage were analyzed.

Table 2 The AI-likeness rate analysis of the alleged match with suspected AI-assisted cheating is shown

Player	Result	$R_t(N_c/N_t)$	$R_1(N_{c1}/N_{t1})$	$R_2\ (N_{c2}/N_{t2})$	$R_3(N_{c3}/N_{t3})$	z-score	p-value
Player A (W)	Lose	0.292 (8.6/29)	0.292 (8.6/15)	– (0.0/14)	0.3 (1.2/25)	0.063	0.475
Player B (B)	Win	0.740 (10.4/30)	0.740 (10.4/15)	– (0.0/15)	0.2 (0.8/25)	4.746	1.04×10^{-6}

The game is played in the on-line tournament. B and W shown in parentheses next to each player's name represents the color of the player's stone. The number of moves in each stage is N_{ti} (i = 1, 2, 3) and the number of analyzed moves in each stage is N_{ci} (i = 1, 2, 3). (N_{ci}/N_{ti}) means that, there are N_{ti} moves and N_{ci} moves are investigated for stage i. The AI-likeness rate for each stage, R_i (i = 1, 2, 3) and the early and middle stages, R_t are also shown

The R_t of Player B are much higher than the average performance of the professional Go player investigated in this study. The average AI-likeness rate of Player B in this game is much higher than that of average player of the 2019–2020 KB Baduk league. The z-score is about 4.746, which is larger than 3.314, the z-score of the highest AI-likeness rate in the 2019–2020 KB Baduk league. Based on the ranking system, the level of Player B's Go playing skill can be considered to be relatively lower than that of the average player of the 2019–2020 KB Baduk league. Even after assuming that Player B's skill is on a par with the average players of the 2019–2020 KB Baduk league, the value of the AILR = 0.740 with z-score = 4.746 represents that this high value of AILR can only happen in one out of about 1,000,000 games. In contrast, Player A's value of AILR = 0.292 with z-score = 0.063 indicates that this game play can happen in one out of two games. Based on this, we can conclude that it is highly unlikely that Player B played the game solely on his or her own skills (without any assistance of AI).

3.3 Analysis of the AI-likeness rate for games played in online

Usually, the level of human Go players is known to vary depending on whether they play in on-line or on-site. We performed the analysis of the games played in on-line to confirm that the results in Table 2 are out of distribution regardless of the environment (Table 3).

The final match of 25th LG Cup was held in an on-line environment. However, using monitoring equipment, AI-assisted playing was virtually impossible. Since it is the final round of the world competition, and each player was given more than three hours of thinking time, the three games would have been the highest level of play. Nevertheless, the average AILR of Shin Minjun is 0.312, and the average of AILR of Ke Jie is 0.277, significantly lower than 0.740 of Player B in Table 2.

3.4 Analysis of the AI-likeness rate for games between an AI and a human

To test our approach to other examples, we applied our method to games played between an AI and a professional player. In the Future of Go Summit [19] held in 2017, Ke Jie (first-ranked Chinese Go player at the time), played 3 games against AlphaGo Master [5], losing all three games. The analysis result is shown in Table 4.

Ke Jie and AlphaGo Master played 289 moves in total in Game 1 of Future of Go Summit. In this game, the early stage is from 21st move to the 50th, the middle stage from the 51st to the 239th, and the last stage from the 240th to the 289th move.

In Game 2, they played 155 moves in total. In this game, the early stage is from 21st move to the 50th move, the middle stage from the 51st to the 105th move, and the last stage from the 106th to the 155th move. According to Demis Hassabis who is the main developer of the AlphaGo project, Ke Jie's play in Game 2 was evaluated highly by AlphaGo Master, making proper response of AlphaGo Master hard to find easily. We observe that Ke Jie's play in Game 2 was not highly AI-like. The fact that his play made AlphaGo Master hard to find proper response seems to be consistent with the

Table 3 AI-likeness rate analysis of the three games of 25th LG Cup final round between Shin Minjun and Ke Jie is shown

	Player	Result	$R_t(N_c/N_t)$	$R_1(N_{c1}/N_{t1})$	$R_2\ (N_{c2}/N_{t2})$	$R_3(N_{c3}/N_{t3})$	z-score	p-value
Game 1 (184)	Ke Jie (W)	Win	0.247(30.0/57)	0.244(9.0/15)	0.248(21.0/42)	0.209 (12.2/25)	-0.407	0.342
	Shin Minjun (B)	Lose	0.234(35.0/57)	0.157(6.2/15)	0.250(28.8/42)	0.371(14.0/25)	-0.539	0.295
Game 2 (198)	Shin Minjun (W)	Win	0.367(47.4/64)	0.444 (10.4/15)	0.346 (37.0/49)	1.0(1.0/25)	0.852	0.197
	Ke Jie (B)	Lose	0.291(38.6/64)	0.524(8.6/15)	0.227(30.0/49)	0.0(3.2/25)	0.053	0.479
Game 3 (302)	Shin Minjun (W)	Win	0.335(50.2/116)	0.489(8.2/15)	0.305(42.0/101)	– (0.0/25)	0.515	0.303
	Ke Jie (B)	Lose	0.293 (47.8/116)	0.389(8.2/15)	0.273(39.6/101)	–(0.0/25)	0.079	0.469

The number shown in parentheses after the game ID is the numbers of the total moves played. See the description of the notations in Table 2

Table 4 AI-likeness rate analysis of the three games of Future of Go Summit between AlphaGo Master and Ke Jie is shown

	Player	Result	$R_t(N_c/N_t)$	$R_1(N_{c1}/N_{t1})$	$R_2(N_{c2}/N_{t2})$	$R_3(N_{c3}/N_{t3})$	z-score	p-value
Game 1 (289)	AlphaGo (W)	Win	0.638(20.4/109)	0.607(7.6/15)	0.627(12.8/94)	–(0.0/25)	3.682	0.0001
	Ke Jie (B)	Lose	0.350(20.0/110)	0.250(8.0/15)	0.402(12.0/95)	–(0.0/25)	0.676	0.250
Game 2 (155)	Ke Jie (W)	Lose	0.114(21.0/42)	0.111(9.0/15)	0.117(12.0/27)	0.218(11.0/25)	-1.790	0.037
	AlphaGo (B)	Win	0.525(25.2/43)	0.393(9.2/15)	0.582(16.2/28)	0.689(9.0/25)	2.496	0.006
Game 3 (209)	Ke Jie (W)	Lose	0.323(12.0/69)	0.217(12.0/15)	0.600(4.8/54)	0.800(1.0/25)	0.385	0.350
	AlphaGo (B)	Win	0.618(11.0/70)	0.696(7.2/15)	0.450(4.0/55)	0.000(0.4/25)	3.474	0.0003

The number shown in parentheses after the game ID is the numbers of the total moves played. See the description of the notations in Table 2

resulting relatively low value of AILR = 0.525 with z-score = 2.496 from the AlphaGo Master side.

They played 209 moves in total in Game 3 of Future of Go Summit. In this game, the early stage represents moves from 21st to the 50th, the middle stage from the 51st to the 159th, and the last stage from the 160th to the 209th. On average, 12 moves of Ke Jie in the early stage, 4.8 moves in the middle stage, and one move in the last stage were investigated in 5 BaduGI simulation runs. 7.2 moves of AlphaGo Master in the early stage, 4.0 moves in the middle stage, and 0.4 moves in the last stage were examined in 5 BaduGI simulation runs.

The average value of AILR = 0.594 of AlphaGo Master from the three plays is relatively low, and we believe that this is due to the fact that BaduGI was developed following the spirit of AlphaGo Zero [5] and that there may exist subtle differences between AlphaGo Zero and AlphaGo Master.

4 Conclusions and discussion

In this paper, we proposed a statistical method to detect the AI-assisted cheating in the game of Go. Using this model, it is possible to detect AI-assisted cheating by calculating the probability that AI-assisted moves come from a professional player's game. To establish a standard measure for the similarity of move-selection between a strong AI developed in the spirit of AlphaGo Zero and an average tournament professional Go player, we generated the distribution of the AI-likeness rate (AILR) from 398 games played among 65 top-ranked Korean professional Go players in the 2019–2020 KB Baduk league. From this distribution, we found that the mean value of AILR $\simeq 0.286$ and the standard deviation $\simeq 0.096$. When AILR of a human Go player is significantly higher than this estimate in terms of z-score, it is unlikely that the player achieved this high level of playing by chance. When we apply this logic to the game play in the alleged game of AI-assisted cheating, we find that AILR = 0.740 with z-score = 4.746 and, its probability of happening by chance, $p = 1.04 \times 10^{-6}$. This means that, assuming he or she has the average Go playing skill of 2019–2020 KB Baduk league, the game play of AILR = 0.740 can happen in one out of 1,000,000 games.

To invalidate our approach, one may attempt a more sophisticated way of AI-assisted cheating, for example, by using an AI in an irregular fashion. We leave this issue to the Go community and subsequent follow-up studies.

Acknowledgements This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017R1E1A1A01077717 and NRF-2018R1D1A1B07049312). We thank KIAS Center for Advanced Computation for providing computing resources.

References

1. S.H. Choe, *Google's computer program beats Lee Se-dol in go tournament*. The New York Times. https://www.nytimes.com/2016/03/16/world/asia/korea-alphago-vs-lee-sedol-go.html. Accessed 10 June 2022
2. P. Baudiš, J.L. Gailly, Pachi: state of the art open source Go program. Adv. Comput. Games Prog. **24–38** (2011)
3. C. Clark, A. Storkey, Teaching deep convolutional neural networks to play go. In *Proceedings of 32nd International Conference on Machine Learning 2015*, pp. 1766–1744 (2015)
4. D. Silver, et al. Mastering the game of go with deep neural networks and tree search. Nature 484–489 (2016)
5. D. Silver, et al. Mastering the game of go without human knowledge. Nature **354–359** (2017)
6. D. Silver, et al., A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **1140–1144** (2018)
7. F. Morandin, et al. Sai: a sensible artificial intelligence that plays go. In *2019 international joint conference on neural networks* (2019)
8. B. Lee, et al., Minigo: a case study in reproducing reinforcement learning research. RML@ICLR (2019)
9. D.J. Wu, Accelerating self-play learning in go. arXiv:1902.10565 (2019)
10. Lizzie-leela zero interface, Github. https://github.com/featurecat/lizzie. Accessed 20 June 2022
11. An elegant go board and sgf editor for a more civilized age, Github. https://github.com/SabakiHQ/Sabaki. Accessed 20 June 2022
12. LizzieYzy-gui for game of go, Github. https://github.com/yzyray/lizzieyzy. Accessed 20 June 2022

13. Youtube. https://youtube.com/watch?v=ciGSqouYN6E. Accessed 10 June 2022
14. H. Lee, A genius go girl who couldn't resist ai temptation... Kim Eunji, cheating, qualification suspended for one year, Chosun Ilbo. https://www.chosun.com/national/2020/11/20/2Y3WOD4S4JFNZEA43YQ3LIL7FA/. Accessed 24 June 2022
15. U. Jang, 1 year imprisonment for a game player who received ai 'hiddenly cheating' at the professional competition, Yonhap News. https://www.yna.co.kr/view/AKR20200715161300004. Accessed 10 June 2022
16. KB baduk league. http://kbleague.baduk.or.kr. Accessed 22 June 2022
17. J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141 (2018)
18. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
19. The future of go summit, Google. https://events.google.com/alphago2017. Accessed 20 June 2022