Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2025)

Policies of Multiple Skill Levels for Better Strength Estimation in Games

Kyota Kuboki[1], **Tatsuyoshi Ogawa**[1], **Chu-Hsuan Hsueh**[1], **Shi-Jim Yen**[2], **Kokolo Ikeda**[1]

[1]Japan Advanced Institute of Science and Technology, Japan
[2]National Dong Hwa University, Taiwan
s2460002@jaist.ac.jp, ogawa.tatsuyoshi@jaist.ac.jp, hsuehch@jaist.ac.jp, sjyen@gms.ndhu.edu.tw, kokolo@jaist.ac.jp

Abstract

In many online competitive games, matchmaking is typically based on ratings systems, such as Elo or Glicko-2. However, these rating systems generally require sufficient matches to achieve accurate estimates, resulting in mismatches with opponents of different skill levels, especially during the early stages. To address this issue, methods for estimating strength from a small number of matches are essential. We use "strength" to refer to a player's overall ability, which can be quantified as a skill level such as a rating tier. In a previous state-of-the-art study, researchers estimated player's skill levels using strength scores learned from human match data. In this paper, we further incorporate policies (i.e., probability distributions of moves) of different skill levels from neural networks trained to imitate human players' gameplay. Namely, we combine features from policies and the strength scores to estimate skill levels. We targeted Go and chess, where abundant data is available. Experiments in Go show that our method achieved 80% accuracy in strength estimation when given 10 matches, increasing to 92% when given 20 matches. Compared to the previous state-of-the-art method's 71% and 84%, our approach yields 8% improvements. Similar improvements were observed in chess. Since our method requires no game-specific features, it can be applied to other games or real-world problems.

Code — https://github.com/kyotakuboki/go-chess-strength-estimation

1 Introduction

In recent years, artificial intelligence (AI) technology has been actively researched in various fields and has achieved remarkable results. Especially in the field of games, AI players have surpassed the strength of top human professionals. Examples of this include AlphaGo for the game of Go (Silver et al. 2016) and AlphaStar for StarCraft II (Vinyals et al. 2019). Beyond research focused on enhancing strength, there is a growing interest in utilizing these powerful AI for applications such as dynamic difficulty adjustment (Fujita 2022) and teaching human players (Hsueh and Ikeda 2024). One of the critical components for realizing these applications is the ability to assess human players' strength.

In various competitive games, well-known rating systems such as Elo (Elo 2008) and Glicko-2 (Glickman 2012) have been widely used to estimate players' strength. In this paper, we use the term "strength" to refer to a player's overall ability, which can be represented by a skill level such as a rating tier. These rating systems rely solely on match outcomes, requiring a sufficient number of matches to provide accurate estimations. For example, Neubauer reported that in Go, the rating deviation under Glicko-2 was about 350 after one match but decreased to around 73 after 50 matches (Neubauer 2024), highlighting the importance of match volume in stabilizing ratings.

However, in practice, there are cases where it is desirable to estimate strength based on a small number of matches. For example, in online games, when a new player joins, traditional rating systems require them to play many matches against opponents of different skill levels, which may lead to a decrease in motivation. Previous studies have shown that mismatched games can reduce players' motivation and increase churn (Kang, Suh, and Kim 2024). A similar issue arises in amateur Go tournaments, which often involve players from different online platforms or local Go clubs, each of which uses its own independent rating system. As a result, the ratings are not directly comparable, making it difficult to use them for fair matchmaking. To address such issues, it is desirable to accurately estimate players' strength with as few matches as possible.

Researchers have proposed various approaches to estimate strength based on a small number of matches. A notable recent study by Chen, Shih, and Wu (2025) introduced a strength estimator based on the Bradley-Terry model. This estimator determines a strength score (s-score) for a given game state and move. The s-scores from multiple moves in one or more matches can be aggregated to estimate players' skill levels. In their experiments, they achieved over 80% accuracy in predicting skill levels within 15 matches in Go and 26 matches in chess, setting a state-of-the-art result.

While the strength estimator achieved remarkable results, we identified areas for improvement. Specifically, a player's overall strength is usually influenced by multiple factors that may not be adequately captured by a single s-score. For example, in Go and chess, abilities such as positional judgment (e.g., maintaining good shape in Go or controlling the center in chess) and tactical calculation (e.g., accurately reading se-

quences of moves) may be evaluated differently. Therefore, we consider it beneficial to consult with multiple models.

In this paper, we propose a strength estimator that utilizes (a) s-scores, as well as (b) policies (probability distributions for selecting moves) across different skill levels and (c) average advantage losses, which indicate the degree of mistakes made by players when compared to strong AI players. We employ supervised learning to train models that take (a) to (c) as inputs and predict skill levels. In experiments, our method achieved 80% accuracy with 10 matches and 92% with 20 matches in Go, with similar tendencies in chess.

Our main contribution lies in the feature design for strength estimation, where we combine multiple types of information extracted from gameplay. Among these features, we found that policies across different skill levels are particularly important, leading to higher accuracy than the state-of-the-art method (Chen, Shih, and Wu 2025). In addition, our method uses game-independent features and demonstrated effectiveness in both Go and chess, suggesting potential applicability to other games.

2 Related Work

2.1 Strength Estimation

Elo rating system (Elo 2008) is a classical approach to estimating players' relative skill levels in competitive games. Players' ratings are updated after each match based on the outcome–whether they won, drew, or lost. A sufficient number of matches is necessary for an accurate rating. Various approaches have been proposed to overcome this limitation. One approach involves using game-specific features that correlate with player strength. For example, researchers have examined the timing of castling in chess (Tijhuis, Blom, and Spronck 2023) and the number of stones captured per move in Go (Moudřík, Baudiš, and Neruda 2015). Another approach evaluates a player's move quality by comparing their decisions with those of a strong AI player. For example, the *average advantage loss* resulting from the player's moves has been analyzed in chess (Guid and Bratko 2006), Go (Kosaka and Ito 2018), and backgammon (GameSite 2000 Ltd. 2011). Additionally, some methods employ deep neural networks to predict player skill levels using game states, moves, etc., as inputs (Moudřík and Neruda 2016; Omori and Tadepalli 2024).

Recently, Chen, Shih, and Wu (2025) proposed a strength estimator trained using human players' matches. This estimator takes a game state and a move as inputs and outputs a strength score (s-score), the higher the stronger. Assume that skill levels are divided into R ranks, with r_1 being the strongest. For a rank r_i, its s-score $\bar{\beta}_i$ is averaged from those of state-move pairs sampled from the rank. The strength estimator was trained based on the Bradley-Terry model so that the average s-scores of the ranks are in descending order, i.e., $\bar{\beta}_1 > \bar{\beta}_2 > \ldots > \bar{\beta}_R$. In addition, they introduced r_∞, defined as the weakest rank, into the training process, aiming for a more robust estimation. Their experiments showed that this method could predict player ranks with over 80% accuracy in just 15 matches in Go and 26 matches in chess, setting a state-of-the-art result.

2.2 Imitating Human Moves Across Skill Levels

Imitating human players' moves across various skill levels is crucial for developing AI players that can both entertain and teach humans. One effective approach to creating AI players that play human-like moves is supervised learning from human matches. For example, Maia (McIlroy-Young et al. 2020) consists of a set of models in chess that predict policies (i.e., probability distributions of moves) across different skill levels. Each model was independently trained on matches corresponding to a specific player rating range. However, training multiple independent models can lead to inconsistencies among skill levels. To address this, Maia-2 (Tang et al. 2024) introduced a skill-aware attention mechanism, encoding player skill levels within a single unified model. A similar approach was taken for Go, where KataGo HumanSL (lightvector 2024) used a single model to predict moves across different skill levels. The inputs for this model include a game state, both players' ranks, and the match date. Since these models output probability distributions over moves, they can be used not only to select the most likely move for a given skill level but also to evaluate how likely or unlikely a move is for players at that skill level.

3 Method

To achieve a more accurate strength estimation, we propose to create meta-models that integrate various models, including those not originally designed for strength estimation. Figure 1 shows an overview of our method, including feature extraction from multiple models and rank estimation using these features. Specifically, we utilize (a) strength scores (s-scores) from the strength estimators by Chen, Shih, and Wu (2025), (b) imitation models that output policies across different skill levels (lightvector 2024; McIlroy-Young et al. 2020), and (c) game state evaluations based on strong AI players (LeelaChessZero 2025; Wu 2020). Among (a) to (c), only (a) was originally designed for strength estimation.

Given a match, we separate the state-move pairs by the players. For a player in the match, we collect features based on k state-move pairs, represented as $\{(s_i, m_i) | i \in [1, k]\}$. In Section 4, we will discuss the k value and the setting for selecting which k moves to include.

To obtain the first set of features, we input each of the k state-move pairs to the strength estimator, resulting in the corresponding s-scores β_i. We then calculate the arithmetic mean of these s-scores using the formula $(\Sigma_{i=1..k}\beta_i)/k$, which we refer to as $MeanSScore$, along with the median $MedSScore$ and standard deviation $StdSScore$.

Next, assume that there are L imitation models $\pi_1, \pi_2, \ldots, \pi_L$ corresponding to skill levels 1 to L. Note that the L skill levels do not need to be the same as the target rank to predict. For example, in our experiments in chess, our target ranks to predict cover ratings from R1000 to R2599, while the imitation models we employ cover ratings from R1100 to R1999. For each skill level j, we input each state s_i to obtain the move probability distribution at skill level j and extract the selection probability, i.e., prior, of the played move m_i, denoted as $p_{j,i} = \pi_j(m_i|s_i)$. We then calculate the geometric mean of these priors, i.e., likelihood, at skill level j using the

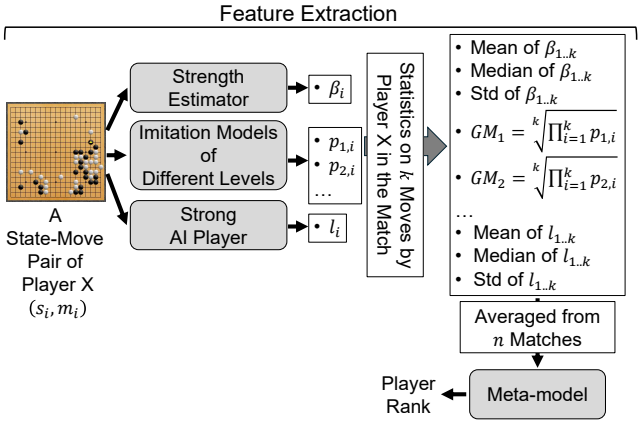


Figure 1: An overview of our method, where β_i denotes s-score of state-move pair (s_i, m_i), $p_{j,i}$ denotes the selection probability of move m_i at state s_i from the imitation model at skill level j, and l_i denotes the loss of move m_i, defined as the difference between the state evaluations before and after executing m_i: $l_i = v_i' - v_i$. Here, v_i is the evaluation of state s_i, and v_i' is the evaluation of the resulting state after playing move m_i, both computed by a strong AI player.

formula $GM_j = \sqrt[k]{\prod_{i=1}^{k} p_{j,i}}$. Namely, the features include L geometric means of the priors for different skill levels.

The main motivation for using imitation models across different skill levels will be explained in more detail with evidence in Subsection 5.3. Briefly speaking, when distinguishing strong players within a narrow skill range (e.g., 9 dan and 8 dan players in Go), priors from models at similarly high skill levels (e.g., 7 dan) often fail to provide sufficient separation. In contrast, models at relatively low skill levels (e.g., 1 dan) can clearly distinguish those strong players by how unlikely the moves are made. Therefore, employing priors from imitation models across different skill levels ultimately help improve the accuracy of strength estimation.

For the final set of features, assume that there is a strong AI player that can provide accurate state evaluations. For each state-move pair (s_i, m_i), we calculate the loss l_i as $v_i' - v_i$, where v_i is the evaluation of s_i, and v_i' is the evaluation of the resulting state after playing m_i. We then obtain the arithmetic mean of these losses, denoted by $MeanLoss$, along with the median $MedLoss$ and standard deviation $StdLoss$. We include losses in the features because we consider that losses can reflect players' ability in reading.

Given a set of matches with the players' ranks being labeled, we first collect the features described earlier. These features, along with players' ranks, are used to train meta-models based on supervised learning for rank estimation.

4 Experiment Settings

4.1 Match Data

For our experiments, we used a Go dataset published on GitHub (featurecat 2019), which contains matches played online on the Fox Go server, and a chess dataset released by the online chess server Lichess (2024). The selection criteria were similar to those of Chen, Shih, and Wu (2025). For Go, target players were those who ranked within 5 kyu (5k), 4 kyu (4k), …, 1 kyu (1k), 1 dan (1d), 2 dan (2d), …, 8 dan (8d), and 9 dan (9d), from weak to strong. The players were separated into 11 rank groups: 3-5k, 1-2k, 1d, 2d, …, 8d, and 9d. The selection criteria for matches were:

- The match must contain at least 50 plies.
- The match was ended after both players passed or when one player resigned, instead of disconnection or timeout.
- Both players must be in the same rank group.

For chess, target players were those whose ratings were from R1000 to R2599 from weak to strong. The players were separeted into 8 rank groups: R1000–R1199, R1200–R1399, …, R2200–R2399, and R2400–R2599.[1] The selection criteria for matches were:

- The match was played in February 2024.
- The match must contain at least 20 plies.
- The time control of the match was blitz.
- Both players must be in the same rank group.

For Go, we collected 5,000 matches for each of the 11 rank groups, using 55,000 matches in total. For chess, we collected 13,000 matches for each of the 8 rank groups, using 104,000 matches in total. From each match, we further separated black and white players into different player instances, resulting in a total of 110,000 player instances for Go and 208,000 player instances for chess. The number of distinct players in each rank group is shown in Table 1.

4.2 Feature Extraction

As explained in Section 3, we utilize strength estimators, imitation models of different skill levels, and strong AI players to estimate strength. In this subsection, we specify the models used in our experiments.

For strength estimators, we employed the models of Go and chess released by Chen, Shih, and Wu (2025), specifically, those with an additional rank r_∞ used in the training process. For imitation models, we employed KataGo HumanSL (lightvector 2024) for Go and Maia (McIlroy-Young et al. 2020) for chess. In more detail, in Go, we obtained policies representing 10 skill levels from KataGo HumanSL by setting the ranks to 10k, 8k, …, 2k, 1d, 3d, …, and 9d, which fully covers the prediction targets 5k to 9d. In chess, we obtained policies representing 9 skill levels from all the released models, Maia-1100, -1200, …, -1800, and -1900, which do not fully cover the prediction targets R1000–R2599. As explained in Section 3, the skill levels of the imitation models do not need to correspond exactly to the rank groups to predict, though it is preferable that the rank groups are fully covered by the skill levels.

For strong AI players, we employed neural networks trained based on AlphaZero from KataGo (Wu 2020) for Go and LeelaChessZero (2025) for chess. More specifically, for

[1] Chen, Shih, and Wu (2025) numbered R ranks $r_1, r_2, \ldots, r_R$ from strong to weak. In this paper, we number rank groups $g_0, g_1, \ldots, g_{R-1}$ from weak to strong.

Rank Group	Train		Test	
	Match #	Player #	Match #	Player #
Go				
3-5k	5,000	8,161	900	886
1-2k	5,000	8,433	900	883
1d	5,000	8,516	900	886
2d	5,000	8,583	900	883
3d	5,000	8,678	900	885
4d	5,000	8,222	900	879
5d	5,000	7,606	900	870
6d	5,000	6,385	900	848
7d	5,000	5,526	900	800
8d	5,000	4,623	900	795
9d	5,000	2,211	900	536
chess				
1000-1199	13,000	11,288	1,200	1,107
1200-1399	13,000	10,444	1,200	1,069
1400-1599	13,000	10,371	1,200	1,096
1600-1799	13,000	9,600	1,200	1,095
1800-1999	13,000	8,332	1,200	1,073
2000-2199	13,000	7,171	1,200	1,013
2200-2399	13,000	6,173	1,200	995
2400-2599	13,000	3,640	1,200	860

Table 1: Statistics on the collected matches in Go and chess used in the training and testing of the main experiments.

state evaluations in Go, we used `scoreLead` predicted by the network, which estimates the number of points the current player is leading in territory. For state evaluations in chess, we first obtained the win rate wr_i predicted by the network for a given state s_i. We then applied a logit transformation to derive the state evaluation v_i by

$$v_i = \log(\frac{wr_i}{1 - wr_i}). \quad (1)$$

The reason for applying the logit transformation is explained as follows. When the winners and losers are clear-cut, the fluctuations in win rates tend to be small. Even if a player makes a terrible mistake that does not alter the match outcome, the resulting change in win rate remains small and fails to adequately reflect the impact of the mistake. By using the logit transformation, we can better capture the mistakes that are made in the endgame phase.

As for k, the number of state-move pairs for obtaining features from a player instance, we experimented with $k \in \{25_{1...25}, 50_{1...50}, \infty\}$. Specifically, $k = 25_{1...25}$ means that we considered the first 25 moves made by the player in the match, while $k = \infty$ means that we considered all moves made by the player. Intuitively, including all moves provides the richest information, leading us to expect to obtain the best results. However, it was interesting to find that the features related to losses did not align with this expectation, which will be further discussed in Subsection 5.3.

4.3 Training of Rank Estimation Meta-models

Assume that there are R rank groups, from the weakest to strongest being $g_0, g_1, \ldots,$ and g_{R-1}. For each group g_j, we collected features from the corresponding player instances, along with j as the rank label, to train regression models. In more detail, we repeated the following process 10,000 times in Go and 26,000 times in chess per rank group g_j: n player instances were sampled, each with k state-move pairs; their features were averaged to form one training sample labeled j. In the experiments, we used $n \in \{1, 5, 10, 15, 20\}$. We employed LightGBM (microsoft 2025) with default settings for regression model training. During training, we further split the collected training samples into training and validation subsets in an 80:20 ratio.

For clarity, we summarize the terms used in this paper:

- A **dataset** is a collection of matches.
- A **match** is played between 2 players.
- A match consists of multiple **state-move pairs** (s_i, m_i), where s_i is a game state and m_i is the move played.
- One player's moves in a match constitute a **player instance**. Thus, one match yields 2 player instances.
- A **training/testing sample** is obtained by averaging features extracted from n player instances with k moves.

5 Main Experiments and the Results

In this section, we present the main results of our rank estimation models and compare them to Chen, Shih, and Wu (2025)'s models.

5.1 Testing Data and Evaluation Procedure

For testing data, we collected matches of Go and chess using the same selection criteria described in Subsection 4.1. For Go, we collected 900 matches for each of the 11 rank groups, using 9,900 matches in total. From each match, we randomly sampled a player and added that player's instance to the testing data, leading to a total of 9,900 player instances. We applied a similar process to chess, with 1,200 matches for each of the 8 rank groups, and obtained 9,600 player instances. The matches collected for training and testing were distinct sets. The number of matches and players for each rank in training and testing data are summarized in Table 1.

Following Chen, Shih, and Wu (2025)'s testing procedure, we evaluated both their models and our meta-models using our testing data. Specifically, for each rank group g_j, both models predicted ranks based on n randomly sampled player instances, where the features were averaged into one testing sample. This process was repeated 500 times to ensure a stable evaluation. In other words, with R rank groups, each model made predictions $500R$ times. In the experiments, we used $n \in \{1, 5, 10, 15, 20\}$.

For our method, the features of the n sampled player instances were averaged and then inputted into the meta-models that were trained with the corresponding n. Since the meta-models are regression models, we rounded the predicted real values to the nearest integers, which then served as the predicted rank groups. In addition, we performed preliminary experiments to select features to prevent overfitting

n	Go		Chess	
	Proposed	Chen et al.	Proposed	Chen et al.
1	**36.2** (**81.4**)	33.7 (74.0)	**32.9** (**79.8**)	31.8 (70.5)
5	**65.9** (**98.7**)	56.3 (96.6)	**61.2** (**98.1**)	51.7 (94.9)
10	**79.5** (**100**)	71.4 (99.7)	**74.9** (**99.8**)	66.5 (99.1)
15	**87.0** (**100**)	78.2 (99.9)	**83.0** (**100**)	73.6 (99.7)
20	**91.7** (**100**)	83.8 (**100**)	**88.2** (**100**)	79.5 (99.9)

Table 2: Rank estimation accuracy with n player instances (accuracy$\pm$1 in parentheses). The boldface indicates the best result between our proposed method and Chen et al. (2025).

due to using too many highly correlated features. As a result, we used the $MeanSScore$, $MedSScore$, $StdSScore$, and the geometric means of priors with $k = \infty$. For features related to loss, we used $MeanLoss$ with $k = 50_{1..50}$ and $MedLoss$ with $k = \infty$ for Go and $MeanLoss$ and $StdLoss$ with $k = 25_{1..25}$ for chess.

We used accuracy as a straightforward evaluation metric for the prediction results, which was defined as the proportion of cases where the predicted rank group was identical to the actual one. Additionally, we considered the metric of *accuracy*$\pm$1 following Chen, Shih, and Wu (2025), which treats predictions of g_{j-1} and g_{j+1} as accurate if the actual rank group is g_j. The accuracy$\pm$1 metric is reasonable, considering that human players' performances may vary.

5.2 Rank Estimation Accuracy

Table 2 shows the rank estimation accuracy of our meta-models compared to Chen, Shih, and Wu (2025)'s models in both Go and chess. With $n = 20$, our meta-models achieved an accuracy of 91.7% in Go and 88.2% in chess, surpassing Chen, Shih, and Wu (2025)'s models by 7.9% and 8.7%, respectively. Additionally, for accuracy$\pm$1, our meta-models achieved 81.4% in Go and 79.8% in chess with merely $n = 1$. Even with a single player instance, our meta-models could effectively estimate a player's rank.

Figure 2 shows the confusion matrices of rank estimation for both our meta-models and Chen, Shih, and Wu (2025)'s models in both Go and chess with $n = 10$, corresponding to the $n = 10$ row in Table 2. Both performed well overall, with most predictions concentrated around $x = y$. While Chen, Shih, and Wu (2025)'s models predicted slightly better at the two ends, i.e., groups 0 and 10 in Go and groups 0 and 7 in chess, our meta-models predicted better in the middle groups (darker colors on $x = y$). This suggests our meta-models achieved more consistent performance across different rank groups, contributing to higher overall accuracy.

Furthermore, we conducted ablation studies to investigate how accuracy was affected by each set of features from strength scores (s-scores), priors from imitation models, and losses. The results are shown in Tables 3 and 4. When s-scores or priors were excluded, the accuracy dropped drastically in both Go and chess. In contrast, removing the losses had a minor effect, with accuracy in Go remaining almost the same and only a slight decrease observed in chess. The

n	Use All	w/o SScore	w/o Prior	w/o Loss
1	36.2 (81.4)	32.7 (77.2)	30.0 (75.9)	**37.1** (**81.7**)
5	**65.9** (**98.7**)	59.2 (96.8)	57.4 (97.4)	65.7 (**98.7**)
10	79.5 (**100**)	73.5 (99.6)	72.3 (99.8)	**80.4** (**100**)
15	87.0 (**100**)	81.7 (**100**)	81.5 (**100**)	**87.2** (**100**)
20	**91.7** (**100**)	86.7 (**100**)	86.7 (**100**)	91.4 (**100**)

Table 3: Ablation study of our meta-model in Go. The boldface indicates the best result among the compared feature settings of our proposed method.

n	Use All	w/o SScore	w/o Prior	w/o Loss
1	**32.9** (**79.8**)	26.2 (73.1)	30.9 (77.7)	32.3 (79.6)
5	**61.2** (98.1)	51.5 (94.5)	57.2 (97.5)	60.2 (**98.3**)
10	**74.9** (**99.8**)	64.4 (98.7)	71.3 (99.6)	74.3(**99.8**)
15	**83.0** (**100**)	72.5 (99.5)	79.3 (**100**)	81.5 (**100**)
20	**88.2** (**100**)	78.1 (99.9)	86.3 (**100**)	87.1 (**100**)

Table 4: Ablation study of our meta-model in chess. The boldface indicates the best result among the compared feature settings of our proposed method.

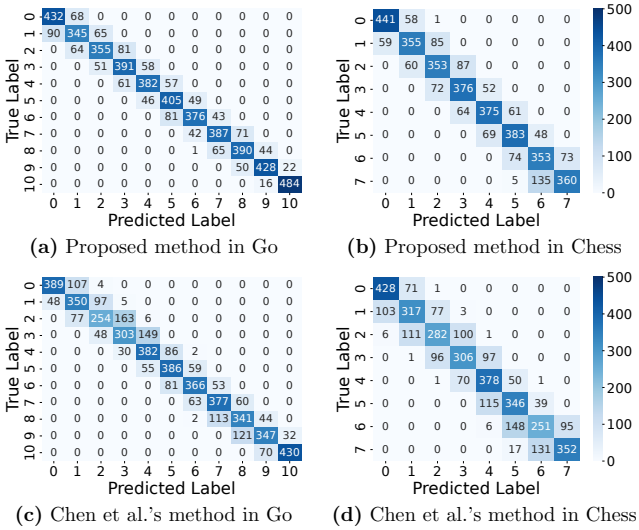
results suggested that s-scores and priors contributed significantly to accurate rank estimation. Interestingly, when only using priors and losses (i.e., the "w/o Strength" column in Tables 3 and 4) , the accuracy in Go was already higher than Chen, Shih, and Wu (2025)'s models, while achieving similar accuracy in chess.

5.3 Discussions

In this subsection, we analyze why prior geometric means across different skill levels contributed to rank estimation and discuss potential issues related to losses.

Effects of Multiple Priors Across Skill Levels Figure 3 illustrates the geometric means of priors across various skill levels in Go and chess for each rank group. Taking Figure 3 (a) as an example, the x-axis represents rank groups from 0 (3–5k, weakest) to 10 (9d, strongest). The y-axis represents the geometric mean of priors, derived from the state-move pairs in the training data described in Subsection 4.1. Each curve corresponds to a specific skill level, such as 10k, 8k, and up to 9d, from which the priors were obtained.

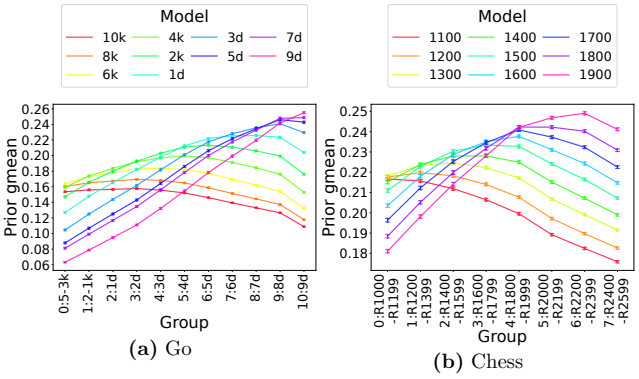
When focusing on a single curve (skill level), we can calculate the slopes for each pair of the adjacent rank groups. A steeper slope suggests that the imitation model at that skill level is more effective in distinguishing between the two rank groups. For example, assume that we want to distinguish 8d and 9d players in Go. We would expect the 2k or 1d model to be more informative than the 5d or 7d model. Specifically, for both 5d and 7d models, the prior geometric means for the 8d and 9d rank groups were very close. In contrast, for both 2k and 1d models, the prior geometric means for the 9d group was significantly lower than that of the 8d group. This indicates that the moves played by 9d players were even less likely to be seen from 2k or 1d players than those played by 8d players.



(a) Proposed method in Go (b) Proposed method in Chess

(c) Chen et al.'s method in Go (d) Chen et al.'s method in Chess

Figure 2: Confusion matrices of rank estimation ($n = 10$).



(a) Go (b) Chess

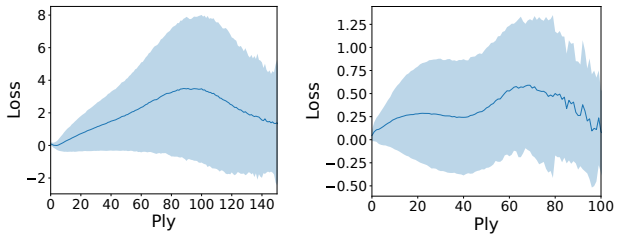
Figure 3: The geometric means of priors, along with 95% confidence intervals, across various skill levels for different rank groups.

From another viewpoint, this time, we take chess as an example. The Maia-1900 model might be useful to distinguish players within rank groups 0 to 3 (R1000 to R1799), while being less useful to distinguish players within rank groups 4 to 7 (R1800 to R2599). On the contrary, the Maia-1500 model might be useful for distinguishing rank groups 4 to 7 but not for 0 to 3. To support the above hypothesis, we conducted a simplified experiment, using only features extracted from Maia models. Specifically, we compared three settings: "Maia-1500 only," "Maia-1900 only," and "both."

Table 5 shows rank estimation accuracy in chess with $n = 20$ when using only Maia-1500, only Maia-1900, or both. Using only Maia-1900 gave relatively high accuracy in low-rated groups (e.g., g_0, g_1), where Maia-1900's prior geometric means were low. In contrast, it failed in g_4 to g_7, predicting most players as g_5, likely due to similar prior geometric means. Using only Maia-1500 worked well in g_3 and g_4 (high prior) but failed in rank groups with low prior geometric means (e.g., g_0, g_1, g_6, g_7). The model could not tell

	g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	Overall
Maia-1500	0.0	0.0	0.2	**87.4**	23.0	0.0	0.0	0.0	13.8
Maia-1900	61.4	54.8	27.2	15.6	18.8	**85.8**	0.0	0.0	33.0
Both	**78.4**	**64.8**	**59.6**	60.2	**70.0**	57.6	**72.2**	**57.8**	**65.1**

Table 5: Accuracy of rank estimation in chess ($n = 20$) using only Maia-1500, Maia-1900, or both. The boldface indicates the best result among the compared feature settings of our proposed method.



(a) Go (b) Chess

Figure 4: The mean loss at each ply.

whether the unlikeness (low prior) came from the player being stronger or weaker. When both were used, the model utilized the pattern that Maia-1500's prior geometric means decreased monotonically in g_4 to g_7, unlike Maia-1900's high prior geometric means there. This allowed the rank estimation model to infer that a lower prior geometric mean with Maia-1500 in this range indicated a stronger player, leading to a significant improvement in overall accuracy.

In addition to the observation that imitation models of different skill levels are useful to distinguish different rank groups, we found another key benefit. The priors associated with each player's moves fluctuate to some extent, as the priors are influenced by the given states and probably the player's thought at that time, causing the prior geometric means to vary even for the same player. To reduce random fluctuations, it may be effective to use multiple indicators that are related to skill levels but not overly correlated with each other. For example, in Figure 3 (b), Maia-1100 and Maia-1200 show similar tendencies, and one may expect that using priors from either model would be sufficient. However, when using priors only from Maia-1100 for rank estimation with $n = 20$, the accuracy was 28.1%, and using only Maia-1200 resulted in 24.3%. In contrast, when using priors from both, the accuracy was improved to 35%.

Even combining models with relatively strong positive correlations, like Maia-1100 and Maia-1200, yielded accuracy improvement. Given this, it is reasonable to expect further accuracy improvement by utilizing more diverse models, such as Maia-1900 or strength estimators.

Potential Issues Related to Losses We included losses in the features because we considered that losses could reflect players' ability in reading, thereby contributing to rank estimation. However, the results in Subsection 5.2 showed that the impact of losses was quite limited. We suspected that

this might be due to the significant variation in losses across different matches.

Figure 4 illustrates the mean loss at each ply for both Go and chess, based on the state-move pairs in the training data described in Subsection 4.1. The shaded areas represent the standard deviations. In both Go and chess, the loss distributions exhibit mountain-shaped patterns. In Go, the peak occurred around the 100th ply, while in chess, two peaks occurred around the 20th and 70th plies. The standard deviations reached their maximum at these peaks in both games.

We interpreted the distributions as follows. In the opening phase, the situation remained relatively stable across all matches. However, in the middle phase, game states varied significantly depending on the match, likely due to increased complexity and strategic divergence. In the endgame phase, the situation became stable again, likely because the outcomes got clearer. As a result, simply averaging the losses over an entire match might lead to underestimation in matches that extended into the endgame phase. This suggests that focusing on a specific range from the opening phase was effective in improving player rank estimation.

6 Extended Experiments and the Results

In the experiments presented in Section 5, we followed the evaluation procedure of Chen, Shih, and Wu (2025), where rank estimation was performed by randomly sampling n player instances from a given rank group. However, in practical applications, we are often interested in estimating a player's rank based solely on their own instances.

When switching from rank-group sampling to player-specific sampling, two key differences arise. First, even within the same rank group, individual players have unique playstyles and weaknesses. For example, some players may be good at positional intuition but struggle with deep reading, leading to frequent mistakes, while others may have strong reading ability to compensate for weak positional intuitions. Such diversity increases the variance in feature distributions, making rank estimation more challenging.

Second, rank-group sampling can mask estimation errors due to averaging effects. For example, if we randomly sample 20 player instances from the R1600–1799 rank group, the average rating of these instances is likely to be around R1700. Even with a prediction error of ± 50 rating points, the model could still estimate the rank group correctly. In contrast, predicting the rank of a player rated R1610 requires more precise estimation. Specifically, the acceptable error range is $[-10, +189]$, indicating the smallest allowable error margin to be 10. From these examples, we can see that the acceptable estimation error differs significantly between the two evaluation procedures. To better reflect real-world settings, we evaluated models under player-specific sampling, where estimation must be accurate for each individual.

6.1 Testing Data and Evaluation Procedure

For testing data, we collected new sets of matches for Go and chess using almost the same selection criteria described in Subsection 4.1, with only one change: Each match must include at least one player who has played 20 or more matches in a given rank group. For each rank group, we collected matches from 100 players, each with 20 instances. As a result, we had a total of 22,000 player instances for Go across 11 rank groups and 16,000 for chess across 8 rank groups.

Using the new testing data, we evaluated models under both rank-group and player-specific samplings to investigate the gap. The rank-group scenario followed the procedure in Subsection 5.1. The player-specific scenario used a similar procedure, with only one change: For each player in a rank group, the models predicted ranks based on n randomly sampled player instances, which were averaged into one testing sample. This procedure was repeated 5 times. Thus, for R rank groups, each model made $5 \times 100 \times R$ predictions, equal to rank-group sampling. In the experiments, we used $n \in \{5, 10, 15\}$. We excluded $n = 1$ because rank-group and player-specific samplings are identical with only one player instance. Additionally, we excluded $n = 20$ because all (20) of a player's instances would be used, yielding the same result in every repetition.

For experiments in this section, we employed the same rank estimation meta-models as those in Section 5. While one might consider developing specific meta-models for player-specific sampling, we believe that additional techniques are necessary to improve the performance, particularly as the size of the training data decreases.

6.2 Rank Estimation Accuracy

Tables 6 and 7 show the rank estimation accuracy of our meta-models compared to Chen, Shih, and Wu (2025)'s models in Go and chess, respectively, under both rank-group and player-specific sampling scenarios. With $n = 15$ under the player-specific sampling, our meta-models achieved an accuracy of 61.6% in Go and 58.6% in chess. Compared to Chen, Shih, and Wu (2025)'s models, the accuracy was 0.5% and 7.8% higher in Go and chess, respectively. The improvement in accuracy for chess was especially significant. The difference in the improvement between Go and chess will be discussed in Subsection 6.3.

Figure 5 shows the confusion matrices of player-specific rank estimation for both our meta-models and Chen, Shih, and Wu (2025)'s models in both Go and chess with $n = 10$, corresponding to the "Player-Specific Sampling" columns in Tables 6 and 7. Similar to Figure 2, most predictions were concentrated around $x = y$, suggesting that both methods performed well. However, compared to the rank-group sampling scenario, errors under player-specific sampling tended to spread more across adjacent rank groups. This indicates a higher variance in individual predictions.

When comparing rank-group to player-specific sampling, we observed significant differences in accuracy, though the differences in accuracy± 1 were relatively minor. For example, with $n = 15$, our meta-models had an accuracy difference of 27.1% in Go and 22.0% in chess between the two scenarios, which was 3.7% and 2.5% for accuracy± 1. To sum up, the results confirmed the gap between rank-group and player-specific sampling for rank estimation.

Similar to Subsection 5.2, we conducted ablation studies under the player-specific sampling scenario to investigate how accuracy was affected by each set of features from

n	Rank-Group		Player-Specific	
	Proposed	Chen et al.	Proposed	Chen et al.
5	**68.1** (**99.0**)	59.8 (97.5)	**56.9** (**95.0**)	51.6 (93.1)
10	**81.3** (**100**)	74.1 (99.7)	**60.0** (95.9)	59.4 (**96.0**)
15	**88.7** (**100**)	81.3 (**100**)	**61.6** (96.3)	61.1 (**96.9**)

Table 6: Rank estimation accuracy using n player instances (accuracy$\pm$1 in parentheses) in Go based on the testing data in Section 6. The boldface indicates the best result between our proposed method and Chen et al.

n	Rank-Group		Player-Specific	
	Proposed	Chen et al.	Proposed	Chen et al.
5	**61.0** (**98.1**)	51.6 (94.3)	**50.6** (**94.4**)	44.4 (87.9)
10	**72.9** (**99.8**)	61.7 (98.6)	**56.7** (**97.0**)	50.0 (91.5)
15	**80.6** (**100**)	67.7 (99.5)	**58.6** (**97.5**)	50.8 (92.7)

Table 7: Rank estimation accuracy using n player instances (accuracy$\pm$1 in parentheses) in chess based on the testing data in Section 6. The boldface indicates the best result between our proposed method and Chen et al.

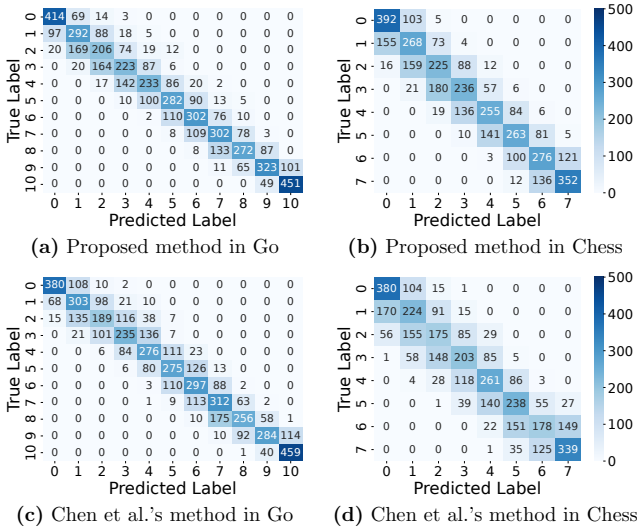
strength scores (s-scores), priors from imitation models, and losses. Tables 8 and 9 show the results of the ablation studies. The tendency was similar to Subsection 5.2. Briefly speaking, when s-scores or priors were excluded, the accuracy dropped drastically. When excluding losses, the accuracy drops were still relatively minor but were bigger and more frequent compared to Tables 3 and 4. We believe this was because losses contributed to evaluating players' reading ability, which became increasingly important for accurately estimating the ranks of individual players. More investigations on the differences between the two scenarios will be made in Subsection 6.3.

6.3 Discussions

In this subsection, we examine the degree of variation in individual players' feature values within the same rank group. Figure 6 shows box plots illustrating the distribution of mean s-scores for 20 sampled player instances from each of the 100 players within every rank group. In both Go and chess, rank groups of stronger players generally had higher mean s-scores. Meanwhile, chess (Figure 6 (b)) tended to show greater overlap between adjacent rank groups compared to

n	Use All	w/o SScore	w/o Prior	w/o Loss
5	56.9 (**95.0**)	51.0 (92.8)	52.1 (93.5)	**57.0** (94.9)
10	60.0 (95.9)	56.5 (94.5)	57.7 (95.9)	**60.3** (**96.1**)
15	61.6 (96.3)	57.5 (94.7)	60.3 (**96.4**)	**61.7** (96.3)

Table 8: Ablation study of our meta-model in Go under the player-specific sampling scenario. The boldface indicates the best result among the compared feature settings of our proposed method.



(a) Proposed method in Go (b) Proposed method in Chess

(c) Chen et al.'s method in Go (d) Chen et al.'s method in Chess

Figure 5: Confusion matrices of player-specific rank estimation ($n = 10$).

n	Use All	w/o SScore	w/o Prior	w/o Loss
5	**50.6** (**94.4**)	46.2 (91.6)	48.4 (92.2)	50.5 (**94.4**)
10	**56.7** (**97.0**)	54.3 (96.1)	52.5 (95.4)	55.5 (**97.0**)
15	**58.6** (**97.5**)	58.3 (96.8)	54.8 (95.7)	58.0 (97.4)

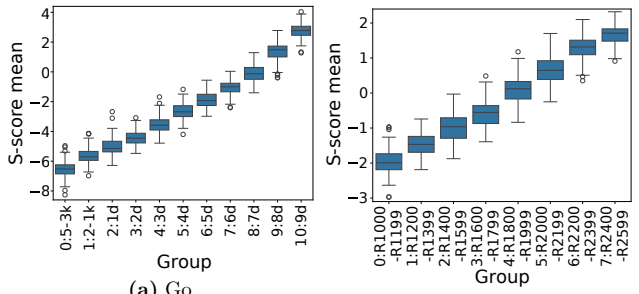
Table 9: Ablation study of our meta-model in chess under the player-specific sampling scenario. The boldface indicates the best result among the compared feature settings of our proposed method.

Go (Figure 6 (a)). This explains the results in Table 7, where using priors or losses in addition to s-scores led to much better rank estimation in chess.

For comparison, Figure 7 shows box plots for the rank-group sampling scenario. Specifically, for each rank group, we repeated the following process 100 times: Randomly sample 20 player instances within the rank groups and obtain the mean s-score. With rank-group sampling, both Go and chess show significantly less overlap (i.e., narrower distributions), suggesting that rank estimation using only mean s-scores would be accurate to some extent. As discussed at the beginning of Section 6, the variance in feature distributions becomes smaller under rank-group sampling because the averaging effects smooth out individual players' characteristics and rating variability. This effect appears to be particularly strong in chess. One possible reason is that each rank group's rating range is slightly wider in chess than in Go if converting ranks 5k, . . . , 9d into ratings. Additionally, it may be that chess shows greater diversity in representations learned inside the strength estimator.

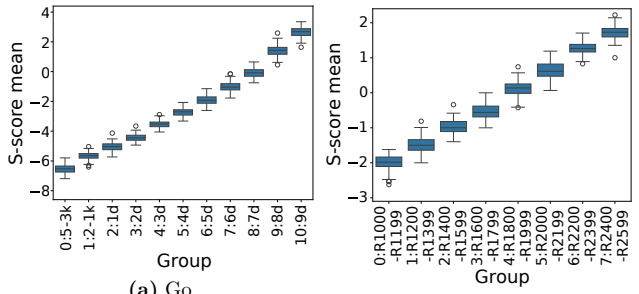
7 Limitations

In this section, we discuss 2 major limitations of this study. (1) Our method relies on models pre-trained on large datasets. For example, the imitation models in chess (Maia)



(a) Go (b) Chess

Figure 6: The mean s-scores of each player in each rank group with $n = 20$.



(a) Go (b) Chess

Figure 7: The mean s-scores of randomly sampled player instances in each rank group with $n = 20$.

were trained on 12 million matches for each skill level. In games where such datasets are unavailable, collecting sufficient training data becomes a major cost. (2) Our method estimates the strength of individual players, but cannot be directly extended to teams. This is because the team strength depends not only on individual players' strength but also on synergy and role division among team members. Designing metrics that capture such team dynamics is necessary to extend the model accordingly.

8 Conclusion

In this study, we proposed a more accurate strength estimation method that combines (a) strength scores from a previous state-of-the-art method, (b) policies that imitate human gameplay across different skill levels, and (c) average advantage losses evaluated by strong AI players.

Experiments on Go and chess showed that our proposed method achieved a higher accuracy compared to a previous state-of-the-art method. In more detail, our method achieved an accuracy of 87.0% in Go and 83.0% in chess when given 15 player instances randomly sampled from the same rank group. The accuracy was 8.8% and 9.4% higher than the previous state-of-the-art method in Go and chess, respectively.

Additionally, we experimented with a more practical setting that estimated individual players' ranks. The accuracy was 61.6% in Go and 58.6% in chess when 15 instances were sampled from each player, which was 0.5% and 7.8% higher than the previous state-of-the-art method. Nevertheless, in all cases, the accuracy was lower than randomly sampling from the same rank group, posing a new challenge.

Our method can be applied to improve the accuracy of initial matching in online matchmaking systems or automatically adjust the level of AI players. Furthermore, the proposed method does not use game-specific features, so it has the potential to be applied to other games.

Acknowledgements

This work was supported by JST BOOST Grant Number JPMJBS2425, JSPS KAKENHI Grant Numbers JP23K11381 and JP23K17021, and the National Science and Technology Council (NSTC) of the Republic of China (Taiwan) under Grant NSTC 113-2221-E-259-018-MY3. The authors acknowledge the use of OpenAI's ChatGPT (OpenAI 2025) for proofreading, language refinement, and assistance in revising the manuscript. The AI-assisted revisions were limited to grammar, style, and clarity improvements, without altering the technical content of the paper.

References

Chen, C. J.; Shih, C.-C.; and Wu, T.-R. 2025. Strength estimation and human-like strength adjustment in games. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*.

Elo, A. E. 2008. *The rating of chessplayers: Past and present*. Ishi Press International, second edition.

featurecat. 2019. Fox Go dataset. https://github.com/featurecat/go-dataset. Accessed March 13, 2025.

Fujita, K. 2022. AlphaDDA: strategies for adjusting the playing strength of a fully trained AlphaZero system to a suitable human training partner. *PeerJ Computer Science*, 8: e1123.

GameSite 2000 Ltd. 2011. eXtreme Gammon Version 2 Documentation. https://www.extremegammon.com/extremegammon2.pdf. Accessed March 14, 2025.

Glickman, M. E. 2012. Example of the Glicko-2 System. Technical report, Boston University.

Guid, M.; and Bratko, I. 2006. Computer analysis of world chess champions. *ICGA Journal*, 29(2): 65–73.

Hsueh, C.-H.; and Ikeda, K. 2024. Improvement of move naturalness for playing good-quality games with middle-level players. *Applied Intelligence*, 54(2): 1637–1655.

Kang, H.; Suh, C.; and Kim, H. K. 2024. Match experiences affect interest: Impacts of matchmaking and performance on churn in a competitive game. *Heliyon*, 10(3).

Kosaka, Y.; and Ito, T. 2018. Examination of indicators for estimating players' strength by using computer Go. In *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 96–101.

LeelaChessZero. 2025. Lc0. https://github.com/LeelaChessZero/lc0. Accessed March 13, 2025.

Lichess. 2024. lichess.org open database. https://database.lichess.org/#standard_games. Accessed March 13, 2025.

lightvector. 2024. New human-like play and analysis. https://github.com/lightvector/KataGo/releases/tag/v1.15.0. Accessed March 13, 2025.

McIlroy-Young, R.; Sen, S.; Kleinberg, J.; and Anderson, A. 2020. Aligning superhuman AI with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1677–1687.

microsoft. 2025. LightGBM. https://github.com/microsoft/LightGBM. Accessed March 13, 2025.

Moudřík, J.; Baudiš, P.; and Neruda, R. 2015. Evaluating Go game records for prediction of player attributes. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 162–168.

Moudřík, J.; and Neruda, R. 2016. Determining player skill in the game of Go with deep neural networks. In *5th International Conference on the Theory and Practice of Natural Computing (TPNC 2016)*, 188–195.

Neubauer, P. 2024. Strength Estimation in the Game of Go. Bachelor's Thesis, TU Wien, Vienna, Austria.

Omori, M.; and Tadepalli, P. 2024. Chess rating estimation from moves and clock times using a CNN-LSTM. In *The 12th International Conference on Computers and Games (CG2024)*.

OpenAI. 2025. ChatGPT. https://chatgpt.com/. Accessed August 28, 2025.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Tang, Z.; Jiao, D.; McIlroy-Young, R.; Kleinberg, J.; Sen, S.; and Anderson, A. 2024. Maia-2: A unified model for human-AI alignment in chess. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*.

Tijhuis, T.; Blom, P. M.; and Spronck, P. 2023. Predicting chess player rating based on a single game. In *2023 IEEE Conference on Games (CoG)*, 1–8.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.; Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wünsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wu, D. J. 2020. Accelerating self-play learning in Go. In *The 34th AAAI Conference on Artificial Intelligence (AAAI-20)*, Workshop on Reinforcement Learning in Games.