Doctoral Dissertation

# Data-Efficient Human-Like Policies at the Group and Individual Levels

Tatsuyoshi Ogawa

Supervisor: Kokolo Ikeda

Japan Advanced Institute of Science and Technology

Kanazawa University

Division of Transdisciplinary Sciences

Doctor of Engineering

March 2026

# Abstract

In this dissertation, we investigate methods for efficiently learning human-like policies under limited data availability. Here, a policy refers to a probability distribution over actions conditioned on states. We address both group-level and individual-level human-like behavior and also explore potential applications of such policies. One such application is human win rate prediction, which aims to estimate the probability that a human player would win the game if playing from a given position. We examine how human-like policies can be effectively applied in human win rate prediction. The primary target game is Shogi, a representative two-player perfect-information game for which large-scale game records are more difficult to obtain than in Chess. For broader comparison and validation, Go and Chess are also included as additional target domains.

First, we investigate the creation of human-like policies at the group-level perspective. We propose a Blend model that mixes the outputs of a supervised learning model and a reinforcement learning model so that human-like moves can be predicted even when human game records are limited. Experiments conducted on Chess, Go, and Shogi demonstrate that the proposed Maia-S, trained with roughly 1/100 of the 12 million-game dataset used in prior work (Maia), can achieve improved human move matching accuracies by combining the supervised model (Maia-S) with an AlphaZero-like reinforcement learning policy. Across the three games, the Blend model improves move-matching accuracy by 0.3–2.9 percentage points for intermediate players and 2.0–6.0 percentage points for advanced players. Additionally, in Chess, the Blend model improves move-matching accuracies by 0.2–1.1 percentage points even when using the original, large-data Maia model. In Go, the Blend model outperforms KL-regularized search, a method that improves move-matching accuracies by searching under calibrated supervised policies. While KL-regularized search itself improves upon Maia, the Blend model yields even greater gains. In Shogi, an analysis of positions where the Blend model is particularly effective reveals that, in positions where supervised models tend to suggest inferior moves, the Blend model more often aligns with human decisions. This indicates that blending compensates for the weaknesses of both supervised and reinforcement-learning-based policies, as originally expected.

Next, we examine how to realize human-like policies at the individual level. We address the challenge that many players have only a small number of game records available. To address this limitation, we propose the Similarity-Guided Fine-Tuning (SGFT) model, which leverages both the target player's games and games similar to them. Two methods for computing similarity are introduced: a feature-based approach using handcrafted game features, and an embedding-based approach using player-behavior embeddings. A two-stage fine-tuning framework is proposed: first fine-tuning with similar games, and then fine-tuning it with the target player's own games. In addition to move-matching accuracy – the primary metric used in prior studies – this dissertation introduces a new evaluation measure based on game features, designed to quantify how well a model imitates individual playing styles. Experiments on Shogi show that SGFT models outperform one-stage fine-tuning, which corresponds to the standard fine-tuning procedure where the target player's game records and similar game records are trained jointly. Moreover, embedding-based SGFT models outperform feature-based ones. The feature-based evaluation further reveals that even Transfer Maia, a prior individual-level imitation method, does not significantly improve certain style-sensitive indicators. These results suggest that SGFT has strong potential for improving human-recognizable, style-specific characteristics.

Furthermore, we discuss the potential applications of human-like policies in playing, teaching, and commentary. A well-known issue in game AI is that AI-estimated win rates often diverge from the win rates that humans would actually achieve. Focusing on human win rate prediction, we propose an inner-product model. It computes the win rate by combining a policy with the predicted win rates of the successor positions. The proposed method surpasses existing models across multiple evaluation criteria – including result accuracy, cross-entropy, and expected calibration error – demonstrating the effectiveness of human-like policies for commentary applications in game AI.

In summary, we present an integrated framework for efficiently learning human-like policies under the practical constraint of limited data. The proposed methods are applicable across multiple games and provide a solid foundation for future research in human-like gameplay, educational AI, and commentary systems.

**Keywords:** human-like behavior, game AI, personalized modeling, supervised learning, reinforcement learning, fine-tuning

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Recent advances in hardware, including GPUs, and improvements in neural network architectures have led to dramatic progress in deep learning technologies used in artificial intelligence (AI). Representative examples include high-quality image generation models based on diffusion models [16] and large language models (LLMs)[2].

AI has not only become more powerful, but has also advanced in a direction that aligns more closely with humans and individual users. A representative example is LLMs used in conversational applications such as ChatGPT. In such applications, LLMs are adapted to human values and preferences through methods such as Reinforcement Learning from Human Feedback (RLHF), enabling the models to generate outputs that are more understandable and appealing to humans [5, 32]. As a result, they can produce outputs that are easy for people to understand and are generally preferred by humans. Research on personalizing LLMs for individual users is growing rapidly. Not limited to LLMs, AI is generally evolving not only to achieve higher performance, but also to become more human-friendly and personalized.

One example of human-friendly AI is an AI that frequently praises a student's strengths. Another example is personalized AI that generates practice problems tailored to the mistakes the student commonly makes. To make AI more human-friendly or more personalized, it is important for AI to understand and/or imitate aspects of human behavior—what we might call human-likeness. However, what does human-likeness mean?

In general, human-likeness can be understood from multiple perspectives, including emotions, preferences, cognitive constraints, perceptual limitations, and ways of speaking, behavior, appearance, and physical characteristics such as body temperature. However, not all of these aspects are equally important in every context. For example, when people expect an AI to play chess in a human-like manner, few would

require the AI's appearance or body temperature to resemble those of a human. In contrast, if an AI is expected to provide caregiving services, human-like appearance and physical warmth may become important factors.

These examples suggest that human-likeness is not a single universal concept. Instead, the aspects of human-likeness required of an AI depend on its intended use and role. In this study, we adopt the position that the definition and evaluation of human-likeness should vary according to the application context.

Developing human-like AI can also be seen in the field of game AI. In two-player perfect information games such as Shogi, Go, and Chess, AI has already achieved performance far beyond that of top human players. Methods based on deep reinforcement learning, such as AlphaGo [42] and AlphaZero [41], have reached superhuman performance through self-play training and have become milestones in game AI research.

Under this background, the goal of modern game AI is no longer limited to simply achieving "victory". From both educational and entertainment perspectives, there is an increasing demand for AI that can reproduce human-like behavior or play under human-like assumptions. For example, in tutoring AIs for beginners, it is useful if the AI can imitate common human mistakes or take them into account. Such behavior makes instruction easier to understand and more effective for users. In spectator-support systems, AIs are expected to predict not only the best move but also the move a human would likely choose. By doing so, they can provide viewers with more intuitive and engaging information, such as the human's winning probability. For these reasons, research on human-like game AI has become increasingly active.

Furthermore, if the behavior of a target player can be successfully imitated, enabling the creation of a game AI that resembles humans at an individual level, its potential applications will greatly expand. For example, in tutoring AIs, it would be possible to present strategies and weaknesses tailored to each player's personal style. In commentary AIs, the system could predict not only which move a specific player is likely to make but also their winning probability at that moment — offering more precise predictions than models that assume an average human. Given these prospects, developing game AI that behaves in a human-like manner — especially at the individual level — is becoming an increasingly important research challenge.

If a large amount of human or individual data is available, it becomes easier to train game AIs that resemble humans or specific individuals. However, games with such abundant data are rare. Few games, like Chess, yield billions of game records and allow access to more than ten thousand games per player for a significant number of players. Therefore, methods that work with limited or even no data are also necessary. For example, when only a small amount of data is available, models

can be trained using evolutionary methods to match in-game statistical properties to those of human players [34]. When no data is available at all, reinforcement learning methods incorporating biological constraints — such as fatigue or reaction speed — can be employed as an alternative [10].

In this study, we focus on games where the amount of available data lies in the middle range — specifically, at least several thousand to tens of thousands of game records in total, and several dozen to several hundred records per individual player. In games such as Shogi and Go, the amount of human gameplay data is smaller than in Shogi, and the number of games available per player varies greatly. These data limitations pose a significant challenge in reproducing human-like and individual-like behavior. To address this challenge, it is essential to establish methods that can effectively utilize limited data to build game AIs that closely resemble human play.

The goal of this study is to establish a method that efficiently learns human-like policies from limited data. Here, a policy refers to a probability distribution that determines which move (action) to select in a given position (state). This study approaches human-like behavior from two perspectives: the group level and the individual level. At the group level, the objective is to imitate general behavioral tendencies shared among many players within a certain group (e.g., players around a rating of 1500). At the individual level, the objective is to learn a policy that reflects the unique characteristics of a specific player.

This dissertation, titled "Human-Like Policies at the Group and Individual Levels Based on Limited Data," consists of seven chapters. Chapter 2, "Target Games and Human-Like Behavior Metrics," describes the games targeted in this study and explains evaluation metrics for measuring human-like behavior in gameplay. Chapter 3, "Literature Review," reviews prior studies related to human-like game AI. Chapter 4, "Human-Like Policies at Group Level," proposes a method that combines supervised learning and reinforcement learning models to develop human-like policies at the group level using limited data. Chapter 5, "Human-Like Policies at Individual Level," proposes a method for creating human-like policies at the individual level by utilizing not only the target player's data but also similar data from other players. Chapter 6, "Applications of Human-Like Policies," discusses the potential applications of human-like policies and presents one example — using such policies to predict human players' win rates more accurately. Finally, Chapter 7 summarizes the findings and contributions of this research and outlines future directions.

# Chapter 2

# Target Games and Human-like Behavior Metrics

This chapter describes the basic rules, gameplay platforms, and data of the target games — Shogi, Go, and Chess. It also explains the evaluation metrics used to assess human-like behavior in these games.

## 2.1 Shogi

### 2.1.1 Overview and Characteristics

Shogi is a two-player perfect-information game that originated in Japan and, like Western Chess, is widely studied as a representative example of such games. Both games are believed to have originated from an ancient Indian game called Chaturanga, which spread in different directions and evolved independently. The version that spread westward developed into Chess in Europe, while the eastern branch evolved into Xiangqi in China, Janggi in Korea, and Shogi in Japan.

Although Shogi and Chess share a common origin, they differ significantly in gameplay due to their distinct rule systems. One of the most notable differences is the piece-in-hand rule. In Shogi, when a player captures an opponent's piece, they keep it as their own piece in hand and, with a few exceptions, may drop it onto any empty square on the board in later turns. As the game progresses, the number and variety of pieces in hand increase, leading to a larger set of legal moves available to the player. Consequently, the state space in Shogi tends to expand in the later stages of the game. This characteristic distinguishes Shogi from games like Chess and Go in terms of search and reinforcement learning.

The Shogi board consists of a $9 \times 9$ grid with 81 squares, and two players take turns alternately making moves. Each player starts with 20 pieces: one king (Gyoku-shō), two gold (Kin-shō), two silver (Gin-shō), two knights (Kei-ma), two lances (Kyō-sha), one rook (Hi-sha), one bishop (Kaku-gyō), and nine pawns (Fu-hyō). The objective of the game is to capture the opponent's king; a player wins by creating a position in which the opponent's king will inevitably be captured on the next move (checkmate). Because Shogi includes the piece-in-hand rule, there is no distinction between the pieces belonging to the two players. However, when colors are used for illustration, the first player is usually represented as black and the second player as white.

Another important rule in Shogi is promotion, known as "nari." All pieces on the board except the king and the gold can be promoted if they are either in or move into the opponent's territory —the farthest three ranks from the player's perspective. When promoted, pawns, lances, knights, and silvers gain the same movement as a gold, while the rook and bishop gain the ability to move one additional square in directions they previously could not.

Shogi, along with Go and Chess, was one of the target games studied in DeepMind's AlphaZero [41]. Although the specific reasons for choosing Shogi were not explicitly stated, it is presumed that its unique characteristics and the prior development of strong Shogi AIs contributed to its inclusion.

### 2.1.2 Data

In this study, the Shogi game records are obtained from Shogi Quest (`https://play.google.com/store/apps/details?id=fm.wars.shogiquest&hl=en` ), one of the largest online Shogi platforms in the world. Each game is stored in the CSA format, which includes information such as moves, player names, and ratings, time consumption, how the game was won (e.g., checkmate, resignation, or a win on time), and whether any handicap was applied. Although these datasets are not publicly available, they were provided by the developer upon request. In this study, we restrict our analysis to even games without hadicaps.

There are three time controls on the platform: 2-minute sudden death, 5-minute sudden death, and 10-minute sudden death. In this study, we use the 10-minute games, as they provide the longest thinking time and has the largest number of games among the three.

## 2.2 Go

### 2.2.1 Overview and Characteristics

Go is a two-player perfect-information game that originated in East Asia and has gained worldwide popularity. Unlike Shogi and Chess, Go is characterized by the placement of stones on the board rather than moving existing pieces, with the objective of securing territory. This distinction makes Go a highly strategic game that requires players to balance local battles with global board control.

In Go, the standard board is a $19 \times 19$ grid, although smaller boards such as $13 \times 13$ and $9 \times 9$ are also widely used. Players take turns placing black (first) and white (second) stones on the intersections of the grid. Once placed, stones cannot normally be moved, and they are removed from the board if completely surrounded by the opponent's stones. The objective of the game is to secure more territory than the opponent by the end of the game, calculated as the total of one's enclosed area plus captured stones. Due to the large board size, the state space of Go is vastly greater than that of Shogi or Chess.

In Go, unlike in Shogi or Chess, it is difficult to design effective feature representations, and accurately evaluating board positions has long been a major challenge. To address this problem, DeepMind's AlphaGo [42] and AlphaZero [41] achieved groundbreaking results by combining deep reinforcement learning with Monte Carlo Tree Search. In particular, AlphaGo's victory over one of the world's top professional players in 2016 marked a major turning point in the history of AI research, demonstrating the power and effectiveness of deep learning–based approaches.

### 2.2.2 Data

In this study, the Go game records are obtained from *Fox Go* (`https://www.foxwq.com/` ), one of the largest online Go platforms in the world. Each game is stored in SGF (Smart Game Format), which contains information such as moves, player names, ratings, and whether any handicap was applied. In this study, we restrict our analysis to even games without hadicaps. Part of this dataset is publicly available and can be downloaded from `https://github.com/featurecat/go-dataset` . The dataset includes approximately 20 million game records.

The standard rule settings use a 6.5-point komi under Chinese rules. Each game allocates 20 minutes of main time per player, followed by a byoyomi phase, in which each move must be played within 30 seconds once the main time expires. Although the time settings can be flexibly adjusted, a match cannot begin unless both players agree on the conditions.

## 2.3 Chess

### 2.3.1 Overview and Characteristics

Chess is a two-player perfect-information game that originated in Europe and is now one of the most widely played board games in the world. It is believed to have originated from the ancient Indian game of Chaturanga. Like Shogi, the objective of Chess is to checkmate the opponent's king by moving one's pieces strategically. However, since Chess does not have a piece-in-hand rule, the number of pieces on the board tends to decrease as the game progresses.

The chessboard consists of an $8 \times 8$ grid with 64 squares, and two players alternately move their pieces — white (first) and black (second). Each player starts with 16 pieces: one king, one queen, two bishops, two knights, two rooks, and eight pawns. The objective of the game is to checkmate the opponent's king, meaning to create a position where the king is inevitably captured on the next move. Chess also includes a promotion rule: when a pawn reaches the farthest rank on the opponent's side, it can be promoted to any piece except the king and another pawn.

Chess has long been an important subject of research in artificial intelligence. It began with Shannon's theoretical analysis in the 1950s [40], and in 1997, IBM's Deep Blue defeated the reigning world champion, Garry Kasparov. Later, AlphaZero [41] demonstrated that a single reinforcement learning algorithm could master three distinct perfect-information games — Chess, Shogi, and go — and surpassed existing Chess AIs such as Stockfish. For this reason, Chess continues to serve as a representative benchmark for evaluating and testing AI systems.

### 2.3.2 Data

In this study, the Chess game records are obtained from Lichess (`https://lichess.org/` ), one of the largest online Chess platforms in the world. Each game is stored in PGN (Portable Game Notation) format, which includes information such as moves, player names, and ratings. These datasets are publicly available and can be downloaded from `https://database.lichess.org/` . The number of available game records reaches several billion.

On Lichess, games are categorized based on the chosen time control, which is defined by the base time per game and the increment added after each move. The classifications are as follows (base time in minutes + increment in seconds):

- Bullet: 1 + 0, 2 + 1
- Blitz: 3 + 0, 5 + 0, 5 + 3

- Rapid: 10 + 0, 10 + 5, 15 + 10
- Classical: 30 + 0, 30 + 20

In this study, we mainly use data from games other than the Bullet category. The reason is that Bullet games have extremely short time limits, leading players to prioritize quick moves over high-quality decisions.

## 2.4 Human-like Behavior Metrics

We consider that the definition of human-likeness depends on the intended to application. In this study, multiple evaluation metrics were used to quantitatively assess how closely the model's policy resembles human behavior. Specifically, three types of metrics were employed: move matching accuracy, likelihood (cross-entropy loss), and game feature based evaluation.

### 2.4.1 Move-Matching Accuracy

Move-matching accuracy is the proportion of positions in which the move chosen by a human player matches the model's top-ranked move. It intuitively represents how often the model selects the same move as a human and has been widely used in prior studies. However, since it considers only the most probable move, it cannot capture the overall shape of the predicted probability distribution.

For example, if the model predicts move A with 51% and move B with 49%, and the human plays move B, this case is evaluated the same as when the model predicts 99% for A and 1% for B. Although the first prediction better reflects human uncertainty, move-matching accuracy cannot account for such differences.

### 2.4.2 Cross-Entropy Loss

To account for the shape of the policy distribution, we use the model's likelihood for the human move $a_{\text{human}}$. Specifically, it is defined as the negative log-likelihood, or cross-entropy loss, as follows:

$$\text{CrossEntropy} = -\frac{1}{N}\sum_{i=1}^{N}\log \pi_\theta(a_{\text{human},i}|s_i), \tag{2.1}$$

where $N$ is the number of positions, $\pi_\theta$ denotes the model's output parameterized by $\theta$, $s_i$ is the $i$-th position, and $a_{\text{human},i}$ is the human move at position $s_i$. Minimizing the

cross-entropy loss is mathematically equivalent to maximizing the likelihood. This metric measures how much probability the model assigns to the human's actual move, thereby capturing the overall shape of the probability distribution. Compared with move-matching accuracy, cross-entropy loss provides a more fine-grained evaluation. While move-matching accuracy measures whether the most probable move matches the human's move, cross-entropy loss evaluates the human-like behavior of the entire probability distribution. In other words, even if the model's most probable move differs from the human's move, the model will still be rated favorably if it assigns high probability to the human's choice.

### 2.4.3 Game Feature Based Evaluation

Move-matching accuracy and likelihood evaluate the model's output distribution based on the human's actual moves. These metrics measure consistency in terms of which move was chosen, but human-like behavior cannot always be captured solely by matching individual moves. For example, if the model predicts move A as the most probable choice while the human actually plays move B, and both moves are offensive in nature, the model's prediction should still be evaluated favorably to some extent, even though the moves differ.

To address this, this study also introduces evaluations based on game features. In the case of Shogi, for example, we analyze features such as how frequently a player drops pieces in their own or the opponent's territory, and how often each piece type is moved. These statistics are then compared across players to evaluate human-like behavior from a broader, behavioral perspective.

# Chapter 3

# Literature Review

We argue that the aspects of human-likeness that should be emphasized depend on the intended application. Accordingly, this chapter reviews prior studies on human-likeness across a range of applications, including opponent AI, cooperative partners, neutral NPCs, test-play agents, spectator support, and instructional agents.

## 3.1 Human-likeness in Opponent AI

In this section, we review and discuss prior studies on human-likeness in opponent AI, with a primary focus on Shogi, Go, and Chess.

Turing predicted that machines would eventually be able to compete with humans in intellectual domains, and he identified Chess as one such promising domain [49]. Since then, extensive research on game AI — including Chess and related board games — has been conducted.

In this study, we examine the development of game AI in Shogi, Go, and Chess from the perspective of human-like behavior, and we categorize its evolution into the following three stages:

1. Imitating humans to make AI stronger: using human game records and human knowledge to improve the strength of AI.

2. Using self-play data to become even stronger: primarily strengthening AI through self-play games between AI agents.

3. Imitating humans in order to imitate: aiming not only for strong play but also for reproducing "human-like" behavior.

### Imitating humans to make AI stronger

Shannon proposed a framework for constructing Chess AI based on two key components: evaluation and search [40]. Evaluation is the process of quantifying the quality of a board position using a combination of multiple features. Shannon listed possible features such as material balance, piece locations, and whether squares adjacent to the king are being attacked. Search refers to the process of looking ahead through multiple candidate moves and comparing them. Shannon distinguished between a simple full-width search (Type A strategy) and a selective search that focuses only on important moves (Type B strategy), noting that the latter resembles the thinking process of skilled human players.

An example of a large-scale implementation of this framework is IBM's Chess AI, Deep Blue [3]. Deep Blue defeated the world champion at the time, Garry Kasparov, in 1997. Its evaluation function incorporated knowledge extracted from more than 700,000 grandmaster games. Although its search included heuristics such as move ordering, it employed a method close to a full-width Type A search.

In Shogi, Hoki et al. developed Bonanza [17], which optimized its evaluation function using human game records. Bonanza achieved a strength comparable to the top amateur level at the time. Whereas traditional Shogi engines relied on handcrafted evaluation functions, Hoki et al. proposed a method to tune the parameters of the evaluation function using 60,000 human games. In this method, the parameters were adjusted so that, in each position, the move selected by the evaluation function would match the move actually played by a strong human player. This approach is conceptually similar to inverse reinforcement learning, in the sense that it estimates an evaluation (value) function from observed actions.

In Go, unlike in Shogi or Chess, it had been difficult to define effective features directly from board positions, and the state space was extremely large, making the development of strong game AI particularly challenging. To address this issue, Coulom proposed a Bayesian supervised learning approach to train a model that predicts the probability distribution of human moves [6]. He trained this predictive model using game records from strong human players and integrated it into Monte Carlo Tree Search (MCTS). As a result, the strength of Go AI improved significantly.

### Using self-play data to become even stronger

Game AI research then shifted toward acquiring knowledge autonomously through self-play. Silver et al. developed AlphaGo, which combined MCTS with deep learning, and defeated top professional Lee Sedol 4-1 in a five-game match [42]. AlphaGo first trained a policy network using supervised learning on human game records to

output move probabilities. It then improved this policy through reinforcement learning by using the network as the initial policy and conducting self-play games. A value network was also trained using reinforcement learning, predicting the win probability of a position based on the outcomes of self-play games. The inputs to these neural networks included binary feature planes representing the locations of the player's and the opponent's stones on the board, as well as additional features based on Go-specific knowledge, such as the set of legal moves. AlphaGo integrates the prior probabilities given by the policy network with the position evaluations provided by the value network through MCTS, thereby greatly improving both the efficiency and accuracy of the search.

Silver et al. subsequently proposed AlphaGo Zero [43], which removed all human game records and domain knowledge from AlphaGo and learned both the policy and value functions solely through self-play, starting from random play. AlphaGo Zero defeated AlphaGo 100-0, demonstrating the potential of AI systems that do not rely on human knowledge. This approach was further generalized to Chess and Shogi as AlphaZero [41], achieving performance exceeding that of the strongest existing engines, such as Stockfish and elmo. These achievements established a general and powerful methodology for building strong game AI in two-player perfect-information games.

### Imitating humans in order to imitate

In recent years, research has increasingly focused on reproducing not only strong play but also human-like behavior, motivated by applications such as education and game commentary. McIlroy-Young et al. developed Maia, a Chess AI that uses a network architecture similar to AlphaZero but is trained to imitate human play rather than to maximize win rate [28]. Maia is trained to predict the moves that players in specific rating ranges are likely to make, and can be viewed as a form of imitation learning, similar to approaches used in robotics. The training data were divided into nine rating ranges, with each model trained on the game records belonging to a single range. As a result, Maia's move-matching accuracy tended to be highest for games near the rating range it was trained on, achieving approximately 50%. McIlroy-Young et al. also argued that using the raw outputs of the neural network, without combining them with search, yields higher move-matching accuracy.

In contrast, Jacob et al. proposed KL-regularized search, which regularizes the search policy so that it becomes closer to the distribution of human moves [23]. They showed that even when using a network similar to Maia, one can achieve both stronger play and higher move-matching accuracy by properly tuning the search

parameters based on theoretical insights.

McIlroy-Young et al. further proposed Transfer Maia, which aims to imitate play at the level of individual players rather than broader rating ranges [27]. Transfer Maia uses an existing Maia model as the base model and fine-tunes it using the game records of specific players (transfer learning) in an attempt to improve player-specific move-matching accuracy. Experiments showed that when only 1,000 game records were available, the Maia model trained on the rating range closest to the target player achieved higher accuracy than Transfer Maia; however, when 5,000 records were available, Transfer Maia outperformed Maia in move-matching accuracy.

A wide range of studies have aimed to develop human-like opponent AI not only in Shogi, Go, and Chess. For example, in the domain of first-person shooters (FPS), a prize-based competition called the 2K BotPrize was held using the FPS game Unreal Tournament 2004. The goal of this tournament is creating agents that behave sufficiently like human players. In the 2012 competition, two agents — UT^2 [38] and MirrorBot [35] — were judged to be sufficiently human-like.

Both agents are composed of broadly defined if–then rules. Furthermore, in UT^2, the agent's main combat behaviors are optimized using Neuroevolution [26], a method that combines neural networks with evolutionary algorithms for optimization. For non-combat situations, it performs navigation based on human player trajectory data. In contrast, MirrorBot attempts to imitate human players during combat. When it detects an enemy who is not overly aggressive and is facing the agent, it begins imitation. More specifically, it reproduces the enemy's actions in a point-symmetric manner approximately 0.5 seconds later, including simulated communication delay.

## 3.2 Human-likeness in Teammate

In cooperative games, a human-like teammate AI refers to an agent that can communicate, make decisions, and coordinate with human players in a manner similar to a human partner. This includes not only understanding the player's intentions but also taking actions that make its own intentions easy for the player to understand. Even if an AI exhibits objectively superior performance, players may feel dissatisfied or lose trust if the AI ignores human intentions or behaves in ways that are difficult for humans to interpret.

A prominent example of this phenomenon is an experiment conducted in 2021 [44] on the cooperative card game Hanabi [1]. In this study, players were paired either with a rule-based bot or with a state-of-the-art reinforcement learning agent. Although the reinforcement learning agent achieved performance comparable to that

of the rule-based bot in terms of objective strength, players reported negative experiences when playing with the reinforcement learning agent, describing it as unpredictable, untrustworthy, and unreliable. In contrast, the simpler and more transparent bot was preferred, even when the team scores were similar. This study demonstrates that, for a teammate AI, subjective factors such as predictability, clarity of intentions, and adaptation to the human partner can be more important than raw strength or optimality.

A representative line of research on human–AI coordination uses the cooperative cooking game Overcooked [20, 37, 11] as a testbed for studying collaboration with human partners. Carroll et al. demonstrated that agents trained solely via self-play often fail to coordinate effectively with humans, and showed that explicitly learning models of human behavior significantly improves coordination performance [4]. This work highlighted that raw task performance is insufficient for successful cooperation, and that understanding human behavior is a key factor in human–AI teaming.

Building on this insight, Strouse et al. addressed the challenge of coordinating with humans without access to human gameplay data [45]. They proposed Fictitious Co-Play, a training framework in which agents learn to cooperate with a diverse population of synthetic partners. Their results showed that exposure to partner diversity during training enables zero-shot coordination with previously unseen human players, emphasizing the importance of robustness and adaptability in cooperative agents.

These cooperative AI approaches implicitly assume that their partners also aim to maximize the environmental reward. However, humans exhibit individual biases and preferences, and often take actions that do not maximize the environmental reward. To address this issue, Yu et al. proposed a framework called Hidden-Utility Self-Play (HSP), which assumes that humans are biased and trains agents to adapt to such biased partners, thereby facilitating more effective cooperation with humans [54]. The HSP model not only achieves higher rewards than baseline methods but is also evaluated by human players as being more supportive.

## 3.3 Human-likeness in Neutral NPC

Neutral NPCs are not designed to directly compete with or cooperate with players, yet they play a crucial role in shaping the perceived realism and liveliness of a game world. In this context, human-likeness is characterized by behavioral consistency and the presence of lively, ongoing activities.

One influential line of research explores generative agent architectures that aim to simulate everyday human behavior. Park et al. propose agents equipped with long-term memory, reflection mechanisms, and forward planning, allowing them to

autonomously generate daily routines, social interactions, and context-dependent actions [33]. These agents are not optimized for player interaction or explicit game objectives; instead, they act according to their own internal states and accumulated experiences. As a result, human-likeness in neutral NPCs emerges from behavioral continuity over time and from the impression that each agent possesses an internal life that exists independently of the player. This perspective highlights that, for neutral NPCs, appearing socially alive and temporally coherent can be more important than responsiveness or utility-driven behavior.

A complementary perspective emphasizes emotion as a key internal driver of human-like behavior. Croissant et al. introduce an appraisal-based emotional architecture integrated with language-model-based agents, in which events are cognitively evaluated to update affective states such as fear, anticipation, or relief [7]. These emotional states subsequently bias reasoning and action generation, producing behavior that is not strictly rational or deterministic but remains psychologically interpretable. In the context of neutral NPCs, such emotion-driven biases enable players to intuitively attribute intentions and feelings to NPCs, even when those reactions are not required by game mechanics.

Taken together, these studies suggest that human-likeness in neutral NPCs is best understood as the presence of internally motivated and emotionally grounded behavior. Park et al. emphasize memory, planning, and social continuity as foundations for human-like background activity, while Croissant et al. demonstrate how affective appraisal introduces cognitive bias and non-determinism characteristic of human behavior. Despite their different implementations, both approaches converge on the idea that neutral NPCs contribute to human-likeness primarily by enriching the ambient social dynamics of the game world, making it feel psychologically and socially alive rather than strategically driven.

## 3.4 Human-likeness in Playtesting Agents

Playtesting agents are primarily designed to explore game mechanics, identify bugs, and assess game balance, rather than to win games or provide challenging opponents. In this context, human-likeness is not defined by skill level or optimal play, but by whether an agent's behavior resembles that of human testers in terms of exploration patterns, failure modes, and sensitivity to game difficulty.

One line of research demonstrates that language-model-based agents, despite not achieving human-level gameplay performance, can effectively function as playtesting agents. Xiao et al. show that guiding agents using simple and general prompt-based techniques can strengthen the correlation between their performance and the

difficulty levels exhibited by human players [51].

A complementary approach focuses on exploration-driven reinforcement learning agents designed to improve playtesting coverage. Gordillo et al. propose curiosity-driven agents that prioritize novel or unexplored game states rather than reward maximization [12]. Such agents exhibit exploratory behavior patterns similar to those of human testers, who often probe edge cases, experiment with unusual actions, and unintentionally trigger unexpected game states. By emphasizing novelty and coverage, these agents uncover a broader range of gameplay scenarios, including rare or unintended interactions that are critical for testing. The resulting behavior is not strategically optimal but is effective for simulating the diverse and sometimes erratic exploration strategies observed in human playtesting.

Importantly, these agents tend to make suboptimal decisions, misunderstand mechanics, or fail repeatedly in ways that mirror novice or intermediate human players. From a playtesting perspective, these limitations are not weaknesses but desirable properties, as they allow the agents to reveal usability problems and difficulty spikes that highly optimized agents would bypass. Consequently, human-likeness in this setting emerges from imperfect understanding and non-optimal behavior that align with typical human testing behavior.

Together, these studies suggest that human-likeness in playtesting agents should be understood as functional similarity to human testers rather than similarity in playing strength. Xiao et al. highlight the importance of agents that fail, misinterpret, and struggle in human-like ways, while Gordillo et al. emphasize exploratory behavior driven by curiosity rather than goals. Both perspectives converge on the idea that, for playtesting purposes, human-likeness is characterized by suboptimal decision-making, diverse exploration, and sensitivity to game difficulty, all of which contribute to more effective and human-relevant testing outcomes.

## 3.5 Human-likeness in Spectator Support System

Human-likeness in spectator support systems does not lie in simply presenting accurate evaluations or optimal moves. Rather, it emerges from how information is presented in ways that align with how human spectators understand a game, where they direct their attention, and how they form convincing explanations. In this section, we characterize human-likeness in spectator support systems from two perspectives: shared attentional focus and concept-level interpretability.

First, the work by Puri et al. proposes a feature attribution method that highlights only those features that are specific and relevant to a particular action when explaining an agent's decision [36]. Instead of emphasizing the entire board, this ap-

proach selectively visualizes the local elements that are essential to the chosen move. Such explanations align well with how human spectators naturally follow a game: rather than processing all available information, they implicitly focus on tactically or strategically salient regions of the board. From this perspective, human-likeness in spectator support emerges not from exhaustive information display, but from sharing the same points of attention that humans are likely to notice.

In contrast, concept-guided chess explanation frameworks focus on interpreting model decisions through high-level strategic concepts, such as central control, king safety, or piece coordination [24]. Rather than directly presenting numerical evaluations or raw search results, these approaches translate model reasoning into the strategic vocabulary that human spectators already possess. For spectators, the key question is not how accurate an AI's evaluation is, but whether they can understand why a position is favorable or unfavorable. Concept-level explanations therefore play a crucial role in supporting intuitive understanding during observation.

Together, these studies suggest that human-likeness in spectator support systems is grounded not in mimicking the decision-making process of players, but in supporting and complementing the cognitive processes of spectators. A human-like spectator support system does not merely assert the best move; instead, it highlights where attention should be directed and provides strategic interpretations that encourage spectators to form their own understanding of the game. In this sense, human-likeness manifests not as similarity in playing strength or behavior, but as similarity in how information is presented and interpreted, closely mirroring the way humans naturally watch and make sense of games.

## 3.6 Human-likeness in Instructional Agents

Human-likeness in instructional agents does not arise from playing strength alone, nor from the mere ability to identify optimal actions. Instead, it emerges from the agent's capacity to support human learning by adapting its behavior and explanations to the learner's level of understanding. In this section, we characterize human-likeness in instructional agents through two key aspects: adaptive control of learning experiences and concept-mediated knowledge transfer.

First, research on producing good-quality games for weak players demonstrates that an instructional agent does not need to play at maximum strength to be effective [19]. Much like a human teacher, a human-like instructional agent should regulate difficulty, avoid overwhelming the learner, and create situations that encourage exploration and reflection. The idea of combining programs with different roles mirrors how human instructors implicitly balance competing objectives — such

as challenge, clarity, and engagement — when guiding a learner. This ability to "hold back" or shape the interaction for educational purposes is a central component of human-likeness in instructional agents.

Second, work on concept discovery and transfer in AlphaZero highlights the importance of how knowledge is expressed [39]. Human instructors rarely teach by exposing raw internal reasoning; instead, they translate their understanding into familiar concepts and abstractions. By extracting human-interpretable strategic concepts from a powerful AI model and using them as explanatory tools, instructional agents can communicate in ways that align with human cognitive frameworks. This concept-level mediation allows learners to internalize strategies rather than merely imitate moves, supporting deeper understanding.

Taken together, these studies suggest that human-likeness in instructional agents is not defined by similarity in performance or decision-making processes, but by similarity in teaching behavior. A human-like instructional agent prioritizes the learner's perspective, adjusts difficulty and interaction dynamics, and conveys knowledge through concepts that humans can readily understand. In this sense, human-likeness in instructional agents is fundamentally about the ability to teach, rather than the ability to win.

# Chapter 4

# Human-Like Policies at Group Level

This chapter is primarily based on the following peer-reviewed journal article, with additional revisions and extensions [30]. Furthermore, Section 4.6.3 presents new experimental results that were not included in the original publication.

## 4.1 Introduction

As the performance of artificial intelligence (AI) has improved, AI can be used for a variety of targets, such as object detection in image processing and question answering in natural language processing. As a target for AI, games have a long history. This is mainly because games have clear rules, are easy to measure players' strength, and are inherently interesting. Game AI has become strong enough to defeat human champions.

However, game AI is not yet a good enough human opponent, and cannot always teach humans the appropriate moves and advice. This is because strong game AI's value functions (predicting win rates) and policies (predicting actions) are not designed for humans. In fact, McIlroy-Young et al. [28] have shown that the policies of strong game AI are very different from those of humans. This becomes challenges when humans try to learn from or enjoy playing against game AIs. In addition, Yannakakis and Togelius [53] stated that human-like agents are important when designing games and generating game content. The reason is that non-human-like agents might give wrong estimation (e.g., whether a level can be cleared). Human-likeness is not only a problem for games, but for AI as a whole, in terms of what kind of action is human-like and natural.

If we have human-like game AI, there are many possible applications. In the following, we illustrate three examples.

1. Human players can enjoy playing against human-like game AI as opponents that have various skill levels to serve a wide range of players.

2. Human-like game AI is also useful as teacher AI. For example, when recommending moves to human players, it is preferable to suggest safer moves that are more likely to lead to wins. In contrast, the best moves of strong game AI may be risky and cause players to lose due to a single mistake in the succeeding moves.

3. Human players sometimes solve problems such as Chess mating problems to improve their game skills, and human-like AI can be used to generate good problems. Human-like AI policy can estimate the difficulty of the problems more accurately than strong game AI.

Human-like game AI can be created through various approaches, including machine learning. For example, Fujii et al. [10] introduced biological constraints into reinforcement learning to train human-like AI in Infinite Mario Bros. When a certain amount of human game records are available, it is straightforward to employ supervised learning to train game AI that mimics human behaviors. One of the most representative examples in recent years is Maia, human-like Chess AI created by McIlroy-Young et al. [28]. Maia used neural networks with similar structures to those used in AlphaZero [41], where the former was trained using human game records while the latter using self-play games. In their experiments, Maia had higher move-matching accuracy with human moves than AlphaZero. In addition, their experiments showed that incorporating the neural networks into tree search decreased the move-matching accuracy. Regarding the incorporation into tree search, Jacob et al. [23] showed different results that tuning parameters appropriately could further improve the move-matching accuracy.

With a large number of game records, supervised learning can imitate human moves to some extent. However, most games do not have as many game records as Maia (Chess). In such cases, it is unclear whether supervised learning can predict human moves well. In this study, we follow Maia's procedure, but create each model with 1% of the number of games to be learned and call it Maia-S (S stands for the initial of "small data"). We also create a model trained on a wide range of data, six times as many as Maia-S, and call it Maia-S-Wide. Moreover, we propose Blend models for Chess, Go, and Shogi (Japanese Chess-like game) that blend Maia-like and AlphaZero-like game AI policies to create more human-like game AI.

This paper is extended from a preliminary version studying Shogi [31] by applying the Blend model to Chess (section 4.4.4) and Go (section 4.4.5) to test the generalization ability. In addition, we propose Maia-S-Wide in section 4.3 and conduct experiments on all three games, Chess, Go, and Shogi (section 4.4.4 to 4.4.6). We also compare the original Maia with the Blend model using Maia in section 4.4.4, compare Kullback-Leibler (KL)-regularized search [23] with the Blend model using Maia-S-Wide in section 4.4.5, and analyze the Blend model using Maia-S-Wide in detail in section 4.4.6.

The rest of the paper is structured as follows. Section 4.4.2 introduces related research. In section 4.4.3, we propose Maia-S, Maia-S-Wide, and the Blend model. Section 4.4.4 to section 4.4.6 show the experimental results of applying these proposed methods to the target games, Chess, Go, and Shogi. Finally, section 4.4.7 makes concluding remarks and discusses future research directions.

## 4.2 Related Research

In the field of games, researchers have created game-playing programs for various purposes. Roughly speaking, in the early stages, more focus was put on creating strong programs. As superhuman programs have been achieved in many games, where the programs' strategies or behaviors are sometimes very different from those of human players, research on human-like game-playing programs attracts more and more attention. The following provides a brief review of research on creating game-playing programs.

In the era when game-playing programs were still much weaker than human players, mimicking human players was an effective way to create strong programs. To create strong Go programs, Coulom [6] proposed a new Bayesian technique for supervised learning for training a model to predict the probability distribution of human players' moves. He used strong human players' games to train the prediction model and then combined the model into a Go program based on Monte-Carlo tree search (MCTS). The Go program's strength was greatly improved. Similarly, some other researchers strengthened their game AI by incorporating move prediction models [48] or evaluation functions trained using human players' games [17].

Reinforcement learning is another effective way to create strong game-playing programs, with the advantage of requiring no human game records. AlphaZero [41] is one of the most famous reinforcement learning methods, which achieved superhuman levels of play by learning from self-play games. AlphaZero used a policy network to predict probabilities of moves and a value network to predict win rates for given positions. The training data of the networks came from self-play games played by a

variant of MCTS that incorporates the networks. AlphaZero beated world champion game AI in Chess, Go, and Shogi.

Well-trained reinforcement learning programs can play smartly, but the strategies or behaviors are sometimes not like humans. With respect to human-likeness, Togelius et al. [47] introduced the concept of "believability". Believability refers to the ability to make a character or bot seem as if it were controlled by a human being. Various approaches have been proposed to achieve human-like characteristics [10, 14, 15].

For Chess, a program called Maia [28] was explicitly designed to imitate human players' moves. This Chess AI used deep neural networks for supervised learning. Human players were divided into 9 groups according to their ratings (numerical strength). Each neural network corresponded to a rating range and was trained using 12 million games from the players in the rating range. Their results showed that moves in a rating range were nearly best predicted by the neural network of the corresponding rating range, where the move-matching accuracy was about 50% [28]. McIlroy-Young et al. claimed that using neural networks alone obtained higher move-matching accuracy than combining the neural networks into tree search as AlphaZero did. However, Jacob et al. [23] showed that even with the same training model as Maia, combining the model into search was stronger and had higher move-matching accuracy if parameters were adjusted properly.

As another approach to create human-like AI, Kinebuchi and Ito [25] proposed to improve move-matching accuracy of Shogi AI by considering the flow of preceding moves. Similar to Maia, they also targeted players in a wide range of skill levels. They represented the flow by combining a search-based value function [17] using a transition probability function [48]. Linear combination was used and the weight was trained using human moves. Their proposed method predicted human moves significantly better than each function alone.

## 4.3 Proposed method

In this study, we propose to combine two policies to create a more human-like policy. The following explains our motivations. It is well-known that supervised learning requires a huge amount of data; or if the amount is not high, the data should have high quality. Previous studies used 12 million amateur game records in Chess [28, 23] or 73 thousand professional game records in Go [23] to train a policy. However, in many games, only a relatively small number of amateur game records are available. In such cases, the policies obtained from supervised learning are imperfect in terms of human-likeness. The reason is that when there are not many similar positions,

generalization is insufficient, and high probabilities may be assigned to bad moves that human players would not play. In subsection 4.4.6.3, we present concrete ratios of such positions. We aim to obtain more human-like policies in such cases. If there are other policies that can compensate for this imperfection, it is valuable to incorporate such policies.

We use move-matching accuracy as a measure of human-like behavior[1]. For a given set of positions with humans' moves, move-matching accuracy of a model is the ratio that the model's moves match the humans' moves. In this study, models select the move with the highest probability for a given position.

We model our problem using a finite Markov decision process ($\mathcal{S}$, $\mathcal{A}$, $T$, $R$) with the following components.

- State space $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$
- Action space $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$
- Transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
- Reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Then, the conditional probability $Pr(A = a \,|\, S = s)$ that a player chooses action $a$ given state $s$ is called policy $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ .

We propose to blend two policies $\pi_1$ and $\pi_2$ to create a human-like policy that takes advantage of each. The new policy $\pi_{blend}$ is calculated as follows:

$$P_{s,a} = \pi_1(a|s)^{\alpha} \times \pi_2(a|s)^{(1-\alpha)} \tag{4.1}$$

and

$$\pi_{blend}(a|s) = \frac{P_{s,a}}{\sum_{i=1}^{|A|} P_{s,a_i}}, \tag{4.2}$$

where $\alpha \in [0, 1]$ is used to represent the importance of $\pi_1$. While this formula blends the two policies as a weighted geometric mean, it is possible to blend the two policies as a weighted arithmetic mean like

$$P_{s,a} = \alpha \times \pi_1(a|s) + (1 - \alpha) \times \pi_2(a|s) \tag{4.3}$$

[1]Other measures of human-like behavior also exist. For example, we used likelihood in addition to move-matching accuracy in our previous work [31]. Since the general tendency of the two measures is similar, we present only move-matching accuracy in this paper to make the results easier to understand.

or a more general form from both (4.1) and (4.3) like

$$P_{s,a} = (\alpha \times \pi_1(a|s)^\beta + (1-\alpha) \times \pi_2(a|s)^\beta)^{1/\beta}. \tag{4.4}$$

In this paper, we use formula (4.1) because preliminary experiments showed that (4.1)'s move-matching accuracy was higher than (4.3) and was about the same as (4.4).

In this study, we focus on two policies for Blend: one from AlphaZero-like reinforcement learning and the other from Maia-like supervised learning. AlphaZero-like policies are strong but not human-like. Maia is basically human-like, but it can make mistakes that humans rarely make. We consider that Blend can prevent non-human-like mistakes and create policies that are basically human-like.

Maia used 12 million game records to train each of their 9 policies, but many other games do not have such a large number of game records. It still remains an issue how to create human-like game AI when the number of game records is small. In this study, we follow Maia's procedure but create each policy with 1% of the number of games to be learned, which we call Maia-S. With only this number of data, policies may not learn well. We suspect that including more data, even from other rating ranges, helps policies to learn better. Therefore, we create another policy trained using game records from a wide rating range (e.g., 1100–1999 in Chess), which we call Maia-S-Wide.

By adjusting the $\alpha$, i.e., the importance of $\pi_1$, on a group or individual basis, it is possible to create $\pi_{blend}$ suitable for that group or individual. In Formula (4.1), consider the case where $\pi_1$ is a Maia-like policy and $\pi_2$ is an AlphaZero-like policy. We expect a larger $\alpha$ fits players with lower ratings and a smaller $\alpha$ fits players with higher ratings.

In the following sections, we will apply the Blend model to Chess, Go, and Shogi and create the necessary Maia-S and Maia-S-Wide policies. We divide our analysis into two parts: Common fundamental analysis and specific analysis for each game. In the fundamental analysis, we aim to answer the following research questions.

- How many differences are in move-matching accuracy between Maia-S and Maia-S-Wide?
- How much improvement does the move-matching accuracy of the Blend model have over each individual policy?
- How does the move-matching accuracy change according to players' ratings?
- How does the move-matching accuracy change for the individual policies and the Blend models?

Although we use Maia-like supervised learning policies and AlphaZero-like reinforcement learning policies in our Blend models and experiments on Chess, Go, and Shogi, we consider that our approach is general and can be applied to other games or even other tasks, explained as follows. For the supervised learning part, policies can be trained as long as humans' gameplay (behavioral) data (state and action pairs) are available. For the reinforcement learning part, policies can be trained using policy-based algorithms as long as the game (the task) can be modeled as a Markov decision process. With these two kinds of policies, we can blend them, aiming to create more human-like policies.

## 4.4 Chess

We chose Chess, Go, and Shogi as our targets, and this section presents the analyses on Chess. Chess is one of the most popular games in the world. Every day, many players play Chess (e.g., on online platforms), and the games are saved. Such plenty of human data serves as a good testbed for studying human-like behavior. Moreover, the game records contain players' ratings (indicators of skill levels), which further enables the possibility to investigate the tendencies of players at different skill levels.

In our Blend model for Chess, we employed an existing reinforcement learning policy (Leela Chess Zero's policy network) and trained our own supervised learning policies. For the latter, we trained Mais-S (using 1% games of the original Maia for each rating range) and Maia-S-Wide (using 6 times as many games as each Maia-S policy but with a wider rating range). Subsection 4.4.1 describes the data and model settings. In subsection 4.4.2, we then apply the Blend model to Chess for basic analysis: We compare Maia-S, Maia-S-Wide, and AlphaZero-like policies with Blend model combining the policies. In the individual analysis in section 4.3, we compare the original Maia with the Blend model using Maia.

### 4.4.1 Data and Model Settings

We used games played on Li hess for training Maia-S-1100, Maia-S-1900, and Maia-S-Wide and for evaluating move-matching accuracy in Chess. Lichess is a popular Chess platform that adopts the Glicko-2 rating system (an extension of the well-known Elo rating system) to evaluate players' skill levels. On this platform, players can choose HyperBullet (30 seconds), Bullet (1 minute), Blitz (3–8 minutes), Rapid (8–15 minutes), and Classical (longer). The time shown in parentheses is the thinking time per player per game, and when a player runs out of this time limit, he or she loses the game immediately.

We employed Maia's codes[2] to create our Maia-S-1100, Maia-S-1900, and Maia-S-Wide policies. In our work, some settings were the same as theirs but the others differed, explained as follows. Settings that were the same as the original Maia were (i) we ignored Bullet and HyperBullet games since the move quality was generally lower, (ii) we discarded the first 10 plies to ignore frequent patterns in the opening, (iii) we discarded moves played when the remaining time was less than 30 seconds to ensure better move quality, (iv) we separated players into nine groups according to their ratings, which were 1100–1199, 1200–1299, ..., 1900–1999, and (v) we collected game records for each rating range where both players were in the same group.

Our Maia-S and Maia-S-Wide differed from the original Maia mainly in the amount of training data, and the learning settings were also adjusted accordingly. In more detail, Maia-S-1100 and Maia-S-1900 used 120,000 game records to train each policy, which was 1% of Maia. Maia-S-Wide used 80,000 game records from each of the nine rating ranges, i.e., a total of 720,000 games. The reason for using 80,000 games instead of 120,000 games from each rating range was to have the same number of games as in the experiments for Go and Shogi. The game records were those played on Lichess in December 2019. 90% of data in each group were training data, 5% were validation data, and the remaining 5% were test data for evaluation[3].

The changes of learning settings were as follows. (i) For Maia-S, the total steps of training was 50,000, i.e., 1/8 of Maia. (ii) For Maia-S-Wide, the total steps of training were 100,000, i.e., 1/4 of Maia. From preliminary experiments, we determined that these numbers of steps were sufficient. (iii) The scheduling of the learning rate defined in terms of the number of steps also changed to fit the lower number of steps. More specifically, we let the scheduling be the same as Maia in terms of the ratio of steps. The learning rate started at 0.1 and was divided by 10 at 1/5, 1/2, and 9/10 of the total steps.

These Maia-S policies served as one of the two policies (say $\pi_1$) in our Blend model. For the other policy $\pi_2$, we employed an AlphaZero-based program, Leela Chess Zero[4] (Lc0). More specifically, we used the neural network's policy and did not incorporate the network into tree search. As Formula (4.1) shows, the Blend model includes the parameter, $\alpha \in [0, 1]$, that determines the influence of $\pi_1$. The larger the value of $\alpha$, the greater the influence of $\pi_1$. We performed preliminary experiments to

[2]https://github.com/CSSLab/maia-chess

[3]In preliminary experiments, we ran several trials to train different Maia-S models for each rating range using different separations of training data, validation data, and test data while keeping the same ratios of 90%, 5%, and 5%. The results of the move-matching accuracy for these Maia-S models and the Blend models were similar (explained in Appendix A). Therefore, in this paper, we present the results of a single trial for each model, which we consider to be reliable enough.

[4]https://github.com/LeelaChessZero/lc0

find promising $\alpha$ settings using a grid search with 0.01 increments. In the following sections, each of the Blend model used a setting of $\alpha$ with the highest move-matching accuracy in each rating range's validation data.

### 4.4.2 Fundamental Analysis

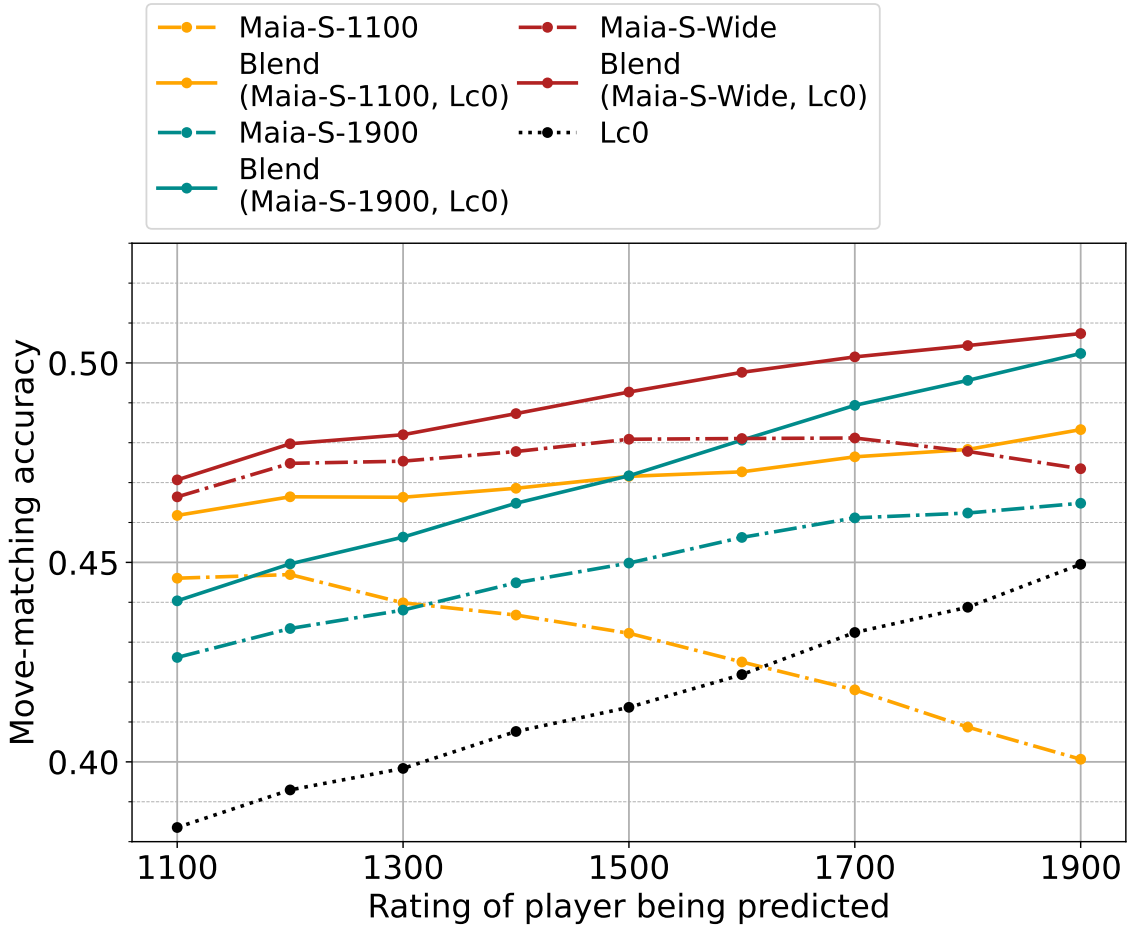


Figure 4.1: Move-matching accuracy of Maia-S models, Leela Chess Zero, and Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rating range) is approximately ±0.002, calculated based on the test data.

In this section, we provide fundamental analyses of the experimental results. Figure 4.1 shows the move-matching accuracy of Maia-S-1100, Maia-S-1900, Maia-S-Wide, Leela Chess Zero, and the Blend models tested on the test data of all the nine

rating groups 1100 to 1900. The x-axis represents ratings and the y-axis represents the move-matching accuracy. The types of lines represent the types of the models. Specifically, the chain lines represent supervised learning policies such as Maia-S and Maia-S-Wide, the dotted line represents Leela Chess Zero (Lc0), and the solid lines represent their Blend models. The colors of the lines represent the rating of the human games used in the training data for the corresponding policy. Specifically, yellow represents using the training data of rating 1100, blue represents using the training data of rating 1900, red represents using the training data of rating 1100 to 1900, and black was Leela Chess Zero, which did not use human game records for its training data.

From the move-matching accuracy of Maia-S-1100 (yellow chain curve) and Maia-S-1900 (blue chain curve), we observed that Maia-S followed the same tendency as Maia: each model has the highest move-matching accuracy in a test rating range that is close to the trained rating range.

Maia-S-Wide (the red chain curve) had higher move-matching accuracy than Maia-S: 0.020 higher than Maia-S-1100 for the rating 1100 data, and 0.009 higher than Maia-S-1900 for the rating 1900 data. Maia-S-Wide had about the same move-matching accuracy regardless of the rating of the data. We considered such improvement in accuracy to be contributed from the increase amount of training data for supervised learning.

The move-matching accuracy of Leela Chess Zero (the black dotted curve) increased as the rating of the test data increased, though it was not as high as Maia-S-Wide even for the rating 1900 data. The results showed that Leela Chess Zero, which was trained without human game records, played less human-like, especially for human players with lower ratings.

The move-matching accuracy of the Blend (Maia-S-1100, Lc0) model (the yellow solid curve) was 0.015 higher than Maia-S-1100 with the best $\alpha$ of 0.69 for the rating 1100 data. The move-matching accuracy of the Blend (Maia-S-1900, Lc0) model (the blue solid curve) was 0.038 higher than Maia-S-1900 with the best $\alpha$ of 0.53 for the rating 1900 data. These results showed that as the player's rating increased, so did the effectiveness of the Blend model. This was also the same tendency as in Leela Chess Zero. We considered this was because we were able to increase the move-matching accuracy for the more difficult moves mentioned earlier.

The move-matching accuracy of the Blend (Maia-S-Wide, Lc0) model (the red solid curve) was 0.004 higher than Maia-S-Wide with the best $\alpha$ of 0.85 for the rating 1100 data, and was 0.034 higher than Maia-S-Wide with the best $\alpha$ of 0.56 for the rating 1900 data. As the same tendency using Maia-S, although with a smaller increase in move-matching accuracy, the effectiveness of the Blend model

also increased as the players' rating increased. We considered this was also because we were able to increase the move-matching accuracy for the more difficult moves mentioned earlier.

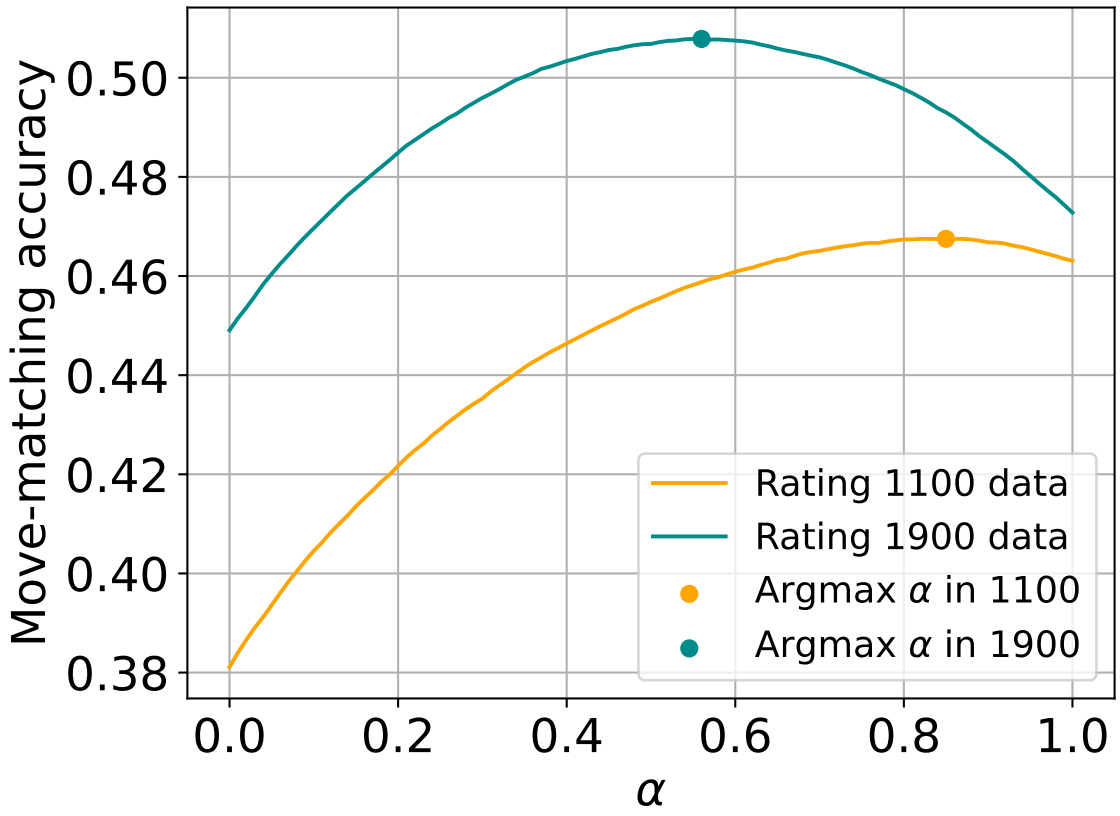


Figure 4.2: Move-matching accuracy of Blend (Maia-S-Wide, Leela Chess Zero) to the rating 1100 and 1900 data for different $\alpha$ using validation data.

We investigated the robustness of the Blend model to $\alpha$, i.e., the influence of $\pi_1$. We targeted the Blend (Maia-S-Wide, Leela Chess Zero) model, which had the highest move-matching accuracy over the validation data in Chess. Figure 4.2 shows the move-matching accuracy of the Blend (Maia-S-Wide, Leela Chess Zero) model for the rating 1100 and 1900 data with different $\alpha$. The x-axis represents $\alpha$ and the y-axis represents the move-matching accuracy. The colors of the lines represent the players' rating. Specifically, yellow represents the validation data for the rating 1100 players, and blue represents the validation data for the rating 1900 players. The dots represent the best $\alpha$ for the validation data. The best $\alpha$ for the rating 1100

was higher than the rating 1900. Namely, $\pi_1$, i.e., Maia-S-Wide, should be weighted more than Leela Chess Zero when predicting low-rated players' moves compared to high-rated players'.

Generally speaking, the appropriate $\alpha$ depends on the target of imitation. When we want to imitate stronger players, the reinforcement learning policy (Leela Chess Zero) should be weighted higher (i.e., smaller $\alpha$). Regarding the robustness, even if $\alpha$ deviates from the optimal value by 0.1, the move-matching accuracy only drops about 0.001 to 0.002. Therefore, $\alpha$ is not a very sensitive parameter, although it needs to be optimized depending on the target data.

### 4.4.3 The Effectiveness of the Blend Model on the Original Maia

In this section, we compare the original Maia policies with the Blend models that used them. The original Maia learned 100 times more game records than Maia-S. Figure 4.3 shows the move-matching accuracy of Maia, Leela Chess Zero, and the Blend model tested on the test data of all the nine rating groups 1100 to 1900. The x-axis represents ratings and the y-axis represents the move-matching accuracy. The types of lines represent the types of the models. Specifically, the dashed lines represent Maia, supervised learning policies, and the dotted line represents Leela Chess Zero (Lc0), and the solid lines represent their Blend models. The colors of the lines represent the rating of the human games used in the training data for the corresponding policy. Specifically, yellow represents using the training data of rating 1100, blue represents using the training data of rating 1900, and black represents Leela Chess Zero, which did not use human game records for its training data.

The move-matching accuracy of the Blend (Maia-1100, Lc0) model (the yellow solid curve) was 0.002 higher than Maia-1100 with the best $\alpha$ of 0.91 for the rating 1100 data. The move-matching accuracy of the Blend (Maia-1900, Lc0) model (the blue solid curve) was 0.011 higher than Maia-1900 with the best $\alpha$ of 0.68 for the rating 1900 data. Even for the original Maia that already used a huge amount of game records to train the policies, the Blend models were still able to improve the move-matching accuracy, though the improvement was smaller than Blend models using Maia-S compared to Maia-S. These results showed that as the player's rating increased, so did the effectiveness of the Blend model. This was also the same tendency as in Leela Chess Zero. We considered this was also because we were able to improve the accuracy of matching moves that were difficult even for models with 12 million game records learned.

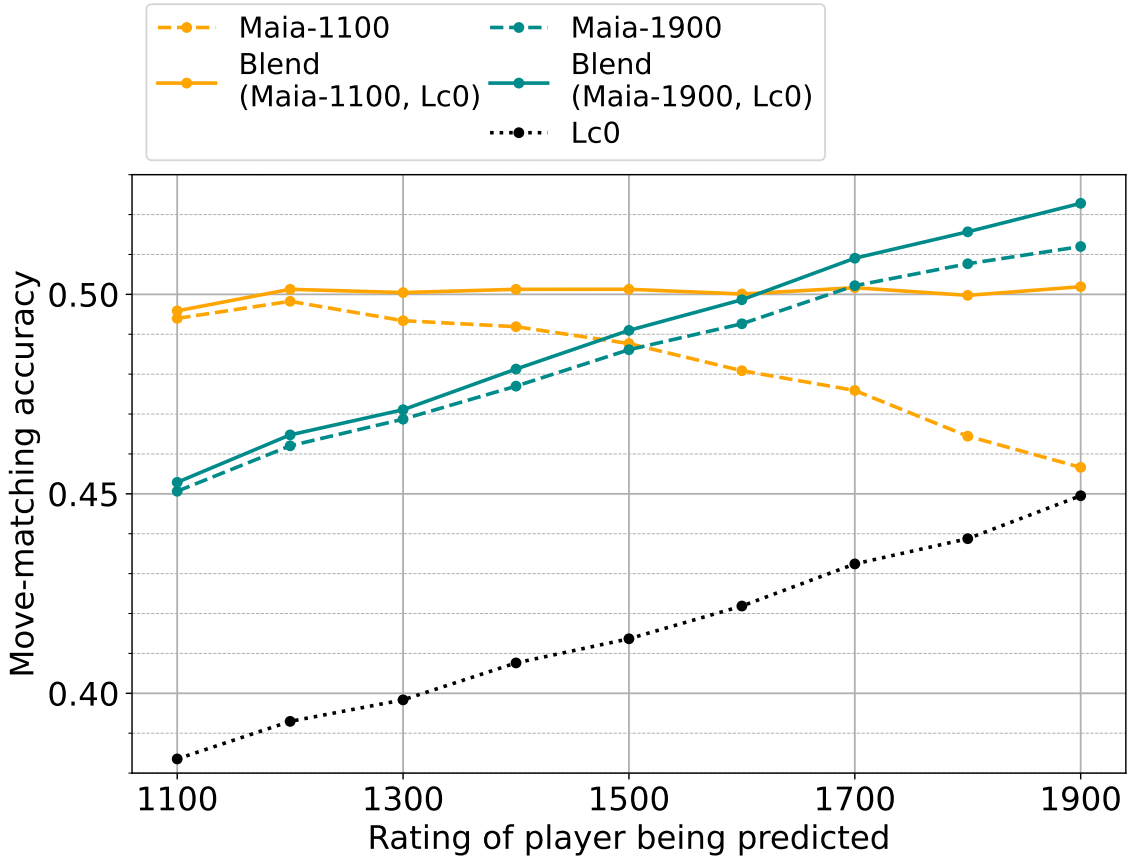


Figure 4.3: Move-matching accuracy of Maia models, Leela Chess Zero, and Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rating range) is approximately ±0.002, calculated based on the test data.

## 4.5 Go

This section presents the analyses on the game of Go. Go is one of the most challenging games to play despite its simple rules. The complexity of Go is higher than that of Chess, and one key point to creating clever game AI is the evaluations of board positions. The game has been actively studied for a long time for various purposes, such as creating strong game AI [41, 6], solving smaller boards [50], and teaching human players [22, 18].

In our Blend model for Go, we employed an existing reinforcement learning policy (KataGo' policy network) and trained our own supervised learning policies. For the latter, we trained Mais-S using 120,000 games (the same number of games as Maia-S in Chess) and Maia-S-Wide using 720,000 games (6 times as many games as each Maia-S policy but with a wider rank range). Subsection 4.5.1 describes the data and model settings. In subsection 4.5.2, we then apply the Blend model to Go for basic analysis: We compare Maia-S, Maia-S-Wide, and AlphaZero-like policies with Blend model combining the policies. In the individual analysis in subection 4.5.3, we compare our Blend model with KL-regularized search, where both methods try to solve the problem that Maia policies sometimes make mistakes even amateurs do not make.

### 4.5.1 Data and Model Settings

We used games played on Fox Go[5] for training Maia-S and Maia-S-Wide in Go and for evaluating move-matching accuracy. Fox Go is a popular Go platform employed a common ranking system to evaluate player's skill levels from beginners (such as 18-kyu or 18k) to experts (such as 9-dan or 9d)[6]. The board size of 19×19 is the most popular, and players can choose the thinking time settings of 20m60s, 5m30s, and 1m20s. In Fox Go, 20m60s means a main time of 20 minutes and three times of 60-second byo-yomi. In this rule, each player is first given 20 minutes to play, and once the 20 minutes are used up, the player must play each move in 60 seconds; if the player goes over 60 seconds three times, the player loses. The same applies to other thinking time.

We employed Leela Zero's codes[7] to create our supervised learning model, Maia-

[5]https://github.com/featurecat/go-dataset

[6]The level of players are defined using kyu and dan ranks. Beginners start from 18-kyu (18k) on Fox Go, and the best kyu grade is 1-kyu (1k). The next grade to 1-kyu is 1-dan (1d), and very strong players may receive 7–9 dan.

[7]https://github.com/leela-zero/leela-zero, which is an AlphaZero-like implementation but also provides codes for supervised learning.

S and Maia-S-Wide policies. We separated players into twelve groups according to their rank, which were, 6k, 5k, ..., 2k, 1k, 1d, 2d, ..., 8d, and 9d. We collected game records for each rank where both players were in the same group. We removed potentially defective game records that contained two continuous moves from the same player or starting from the second player, where a possible reason was that the pass moves were not saved.

The following explains the details of the training data for Maia-S and Maia-S-Wide. We trained Maia-S-3k and Maia-S-6d using 120,000 games of 3k players and 6d players, respectively. For Maia-S-Wide, we used 80,000 game records from each of the ranks between 3k and 6d, which contained a total of 720,000 games. The reason for the number of games per rank was to keep the total number consistent for both Chess and Shogi. In each group, 90% of the game records were the training data, 5% were the validation data, and 5% were the test data. From the validation and test data, we randomly sampled 50,000 positions for each set[8].

The learning settings for Maia-S and Maia-S-Wide were as follows. For Maia-S, the total steps of training were 120,000 steps. For Maia-S-Wide, the total steps of training were 520,000 steps. We determined these numbers of steps by checking the validation loss trend in our preliminary experiments. Regarding the structure of the neural network based on ResNet [13], the number of residual blocks was 6 and the number of filters was 128. All other settings are defaults.

These Maia-S policies served as one of the two policies (say $\pi_1$) in our Blend model. For the other policy $\pi_2$, we employed an AlphaZero-based program, KataGo[9]. Similar to the experiments on Chess in section 4, we used the policy of KataGo's neural network and tuned the parameter $\alpha$ for the Blend model using each rank's validation data.

### 4.5.2 Fundamental Analysis

In this section, we provide fundamental analyses of the experimental results. Figure 4.4 shows the move-matching accuracy of Maia-S-3k, Maia-S-6d, Maia-S-Wide, KataGo, and the Blend models tested on the test data of all the 15 rank groups 6k to 9d. The x-axis represents rank and the y-axis represents the move-matching accuracy. The types of lines represent the types of the models. Specifically, the chain

[8]The main reason for sampling 50,000 positions instead of using the whole 5% of the game records was to reduce the experiment time as one of the compared approaches, KL-regularized search, was time-consuming (for comparison, other approaches were based on the outputs of neural networks).

[9]https://github.com/lightvector/KataGo

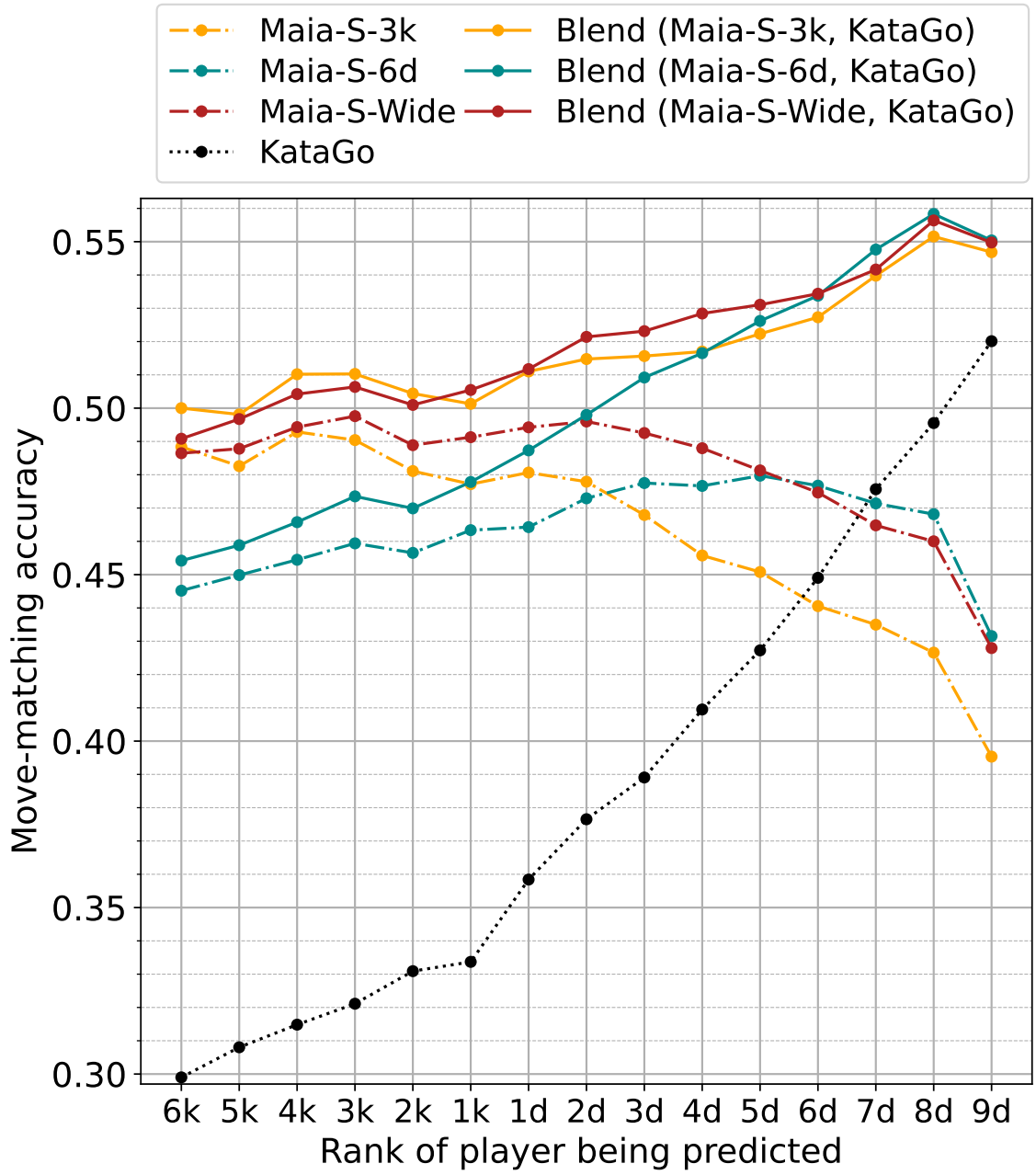


Figure 4.4: Move-matching accuracy of Maia-S models, KataGo, and the Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rank) is approximately ±0.004, calculated based on the test data.

lines represent supervised learning policies such as Maia-S and Maia-S-Wide, the dotted line represents KataGo, and the solid lines represent their Blend models. The colors of the lines represent the rank of the human games used in the training data for the corresponding policy. Specifically, yellow represents using the training data for 3k players, blue represents using the training data for 6d players, red represents using the training data for 3k to 6d players, and black represents KataGo, which did not use human game records for its training data.

From the move-matching accuracy of Maia-S-3k (yellow chain curve) and Maia-S-6d (blue chain curve), we observed that Maia-S followed the same tendency as Maia in Chess: each model has the highest move-matching accuracy in a test rank that is close to the trained rank.

Maia-S-Wide (the red chain curve) had about the same move-matching accuracy as Maia-S: 0.007 higher than Maia-S-3k for the 3k data but 0.002 lower than Maia-S-6d for the 6d data. Maia-S-Wide had about the same move-matching accuracy regardless of the ranks that were covered by the training data. We considered such improvement in accuracy to be contributed from the increase amount of training data for supervised learning.

The move-matching accuracy of KataGo (the black dotted curve) increased as the rank of the test data increased, though it was not as high as Maia-S-6d even for the 6d data. The results showed that KataGo, which was trained without human game records, played less human-like, especially for human players with lower ranks.

The move-matching accuracy of the Blend (Maia-S-3k, KataGo) model (the yellow solid curve) was 0.020 higher than Maia-S-3k with the best $\alpha$ of 0.78 for the 3k data. The move-matching accuracy of the Blend (Maia-S-6d, KataGo) model (the blue solid curve) was 0.057 higher than Maia-S-6d with the best $\alpha$ of 0.58 for the 6d data. These results showed that as the player's rank increased, so did the effectiveness of the Blend model. This was also the same tendency as in KataGo. We considered this was because we were able to increase the move-matching accuracy for the more difficult moves mentioned earlier.

The move-matching accuracy of the Blend (Maia-S-Wide, KataGo) model (the red solid curve) was 0.009 higher than Maia-S-Wide with the best $\alpha$ of 0.85 for the 3k data, and was 0.060 higher than Maia-S-Wide with the best $\alpha$ of 0.54 for the 6d data. The tendency was the same, but stronger when Maia-S was used: the effectiveness of the Blend model also increased as the players' rank increased. We considered this was also because we were able to increase the move-matching accuracy for the more difficult moves mentioned earlier.

We investigated the robustness of the Blend model to $\alpha$, i.e., the influence of

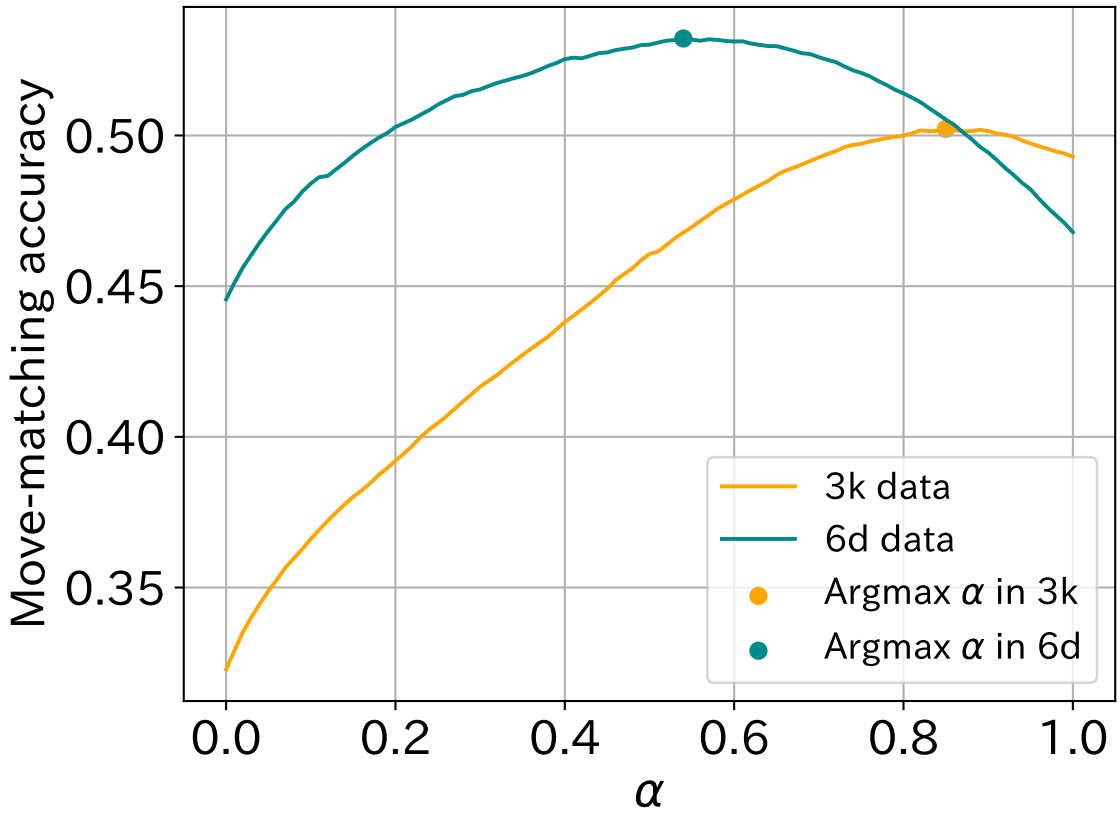


Figure 4.5: Move-matching accuracy of Blend (Maia-S-Wide, KataGo) to 3k and 6d data for different $\alpha$ using validation data.

$\pi_1$. We targeted the Blend (Maia-S-Wide, KataGo) model, which had the highest move-matching accuracy over the validation data in Go. Figure 4.5 shows the move-matching accuracy of the Blend (Maia-S-Wide, KataGo) model for 3k and 6d data with different $\alpha$. The x-axis represents $\alpha$ and the y-axis represents the move-matching accuracy. The colors of the lines represent the players' rank. Specifically, yellow represents the validation data for 3k players (low-rank players), and blue represents the validation data for 6d players (high-rank players). The dots represent the best $\alpha$ for the validation data. The best $\alpha$ for 3k was higher than 6d. Namely, $\pi_1$, i.e., Maia-S-Wide, should be weighted more than KataGo when predicting low-rank players' moves compared to high-rank players'.

Generally speaking, the appropriate $\alpha$ depends on the target of imitation. When we want to imitate stronger players, the reinforcement learning policy (KataGo) should be weighted higher (i.e., smaller $\alpha$). Regarding the robustness, even if $\alpha$ deviates from the optimal value by 0.1, the move-matching accuracy only drops about 0.001 to 0.002. Therefore, $\alpha$ is not a very sensitive parameter, although it needs to be optimized depending on the target data.

### 4.5.3 Comparison with KL-Regularized Search

In addition to our Blend model, the KL-regularized search [23] is another approach aiming to improve Maia's move-matching accuracy to human players. Our Blend model combines two policies from different neural networks, while the KL-regularized search incorporates neural networks into tree search. The KL-regularized search has one parameter, $c_{puct}$. As $c_{puct}$ approaches $\infty$, the policy after search gets closer to the policy from the neural network; as $c_{puct}$ approaches 0, the policy after search becomes greedy and always selects the move leading to the highest value estimated by the neural network. We conducted preliminary experiments on $c_{puct}$ with settings of 0.5, 1, 2, 5, 10, and $\infty$. For each rank, KL-regularized search used the $c_{puct}$ with the highest move-matching accuracy. To reduce computation time, we measured move-matching accuracy only for 6k, 3k, 3d, 6d, and 9d.

In the following, we compare our Blend model with the KL-regularized search, where both used the Maia-S-Wide neural network. Figure 4.6 shows the move-matching accuracy of Maia-S-Wide, KataGo, the Blend model, and KL-regularized search tested on the test data of 6k to 9d. The x-axis represents rank and the y-axis represents the move-matching accuracy. The types of lines represent the types of the models. Specifically, the chain line represents Maia-S-Wide policy, the dotted line represents KataGo, the solid line represents their Blend models, and the dashed line represents KL-regularized search using the Maia-S-Wide network. The colors of

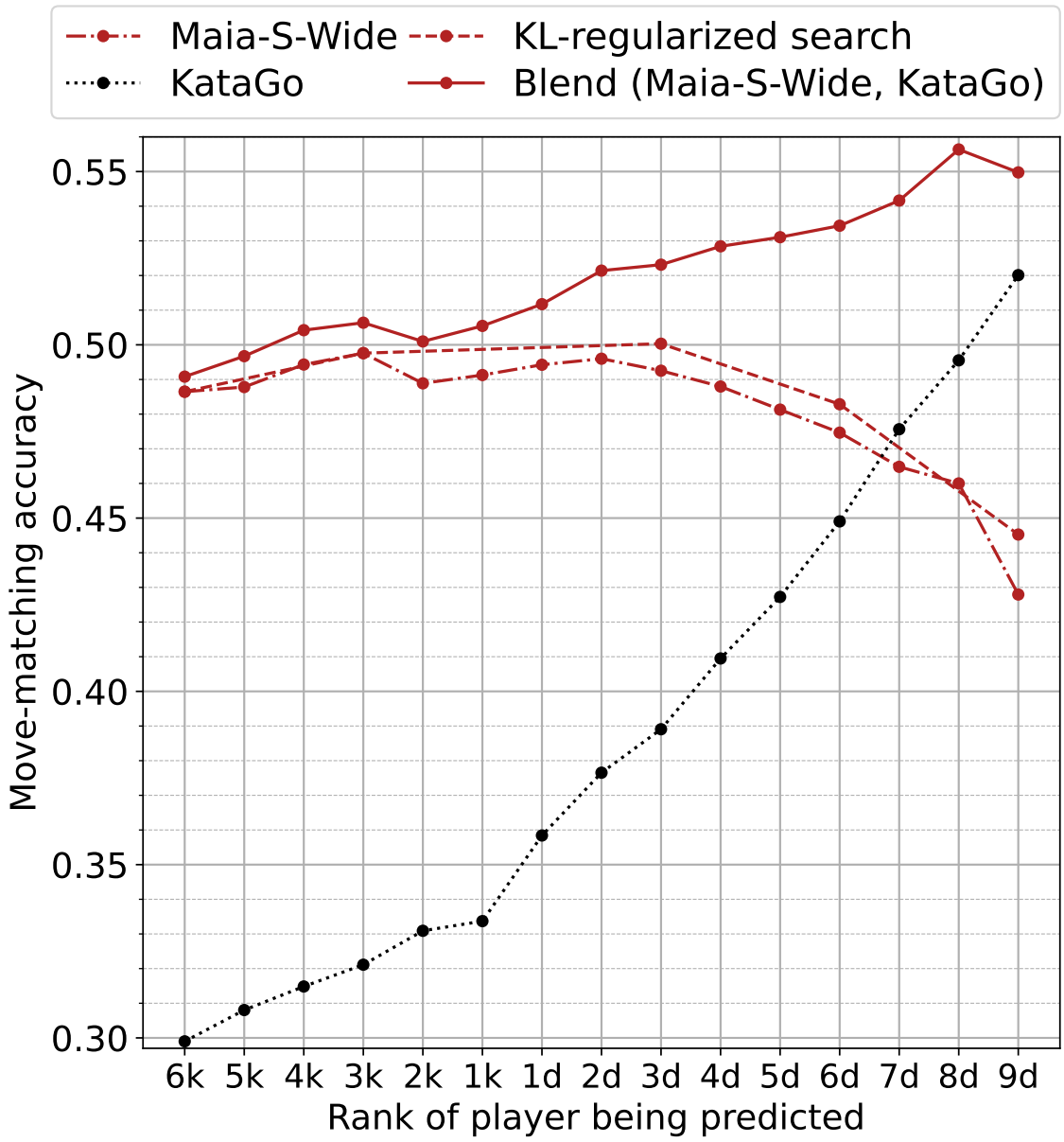


Figure 4.6: Move-matching accuracy of Maia-S-Wide model, KataGo, Blend model, and KL-regularized search. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rank) is approximately ±0.004, calculated based on the test data.

the lines represent the rank of the human games used in the training data for the corresponding policy. Specifically, red represents using the training data for 3k to 6d players, and black represents KataGo, which did not use human game records for its training data.

The move-matching accuracy of KL-regularized search (the red dashed curve) was the same as Maia-S-Wide (i.e., $c_{puct} = \infty$) for the 3k data and was 0.008 higher than Maia-S-Wide with the best $c_{puct}$ of 1 for the 6d data. Compared to our Blend model (the red solid curve), the improvement over Maia-S-Wide was limited. The differences between the Blend model and the KL-regularized search were especially significant for high-ranked players. We considered the results reasonable, explained as follows. Since Maia-S-Wide was trained on amateur game records (relatively poor-quality data), difficult moves that were seldomly played in those games were still hardly selected even with search. In contrast, these difficult moves may be selected in our Blend model owing to KataGo's policy.

## 4.6 Shogi

This section presents the analyses on Shogi. Shogi is a Japanese Chess-like game. The main difference between Shogi and Chess is allowing captured pieces to be returned to the board by the capturing player. Because of this rule, the pieces involved in the games are never reduced, making the endgames remain complex. Shogi has been actively investigated in the research field of games for a long time [21], which is also one of the three targets of AlphaZero [41].

In our Blend model for Shogi, we employed an existing reinforcement learning policy (DLShogi's policy network) and trained our own supervised learning policies. For the latter, we trained Mais-S (using 1% games of the original Maia for each rating range) and Maia-S-Wide (using 6 times as many games as each Maia-S policy but with a wider rating range). Subection 4.6.1 describes the data and model settings. In subsection 4.6.2, we then apply the Blend model to Shogi for basic analysis: We compare Maia-S, Maia-S-Wide, and AlphaZero-like policies with Blend model combining the policies. In the individual analysis in subsection 4.6.3, we investigate the strongness of the Blend models and the mechanism why the Blend models can better predict human moves.

### 4.6.1 Data and Model Settings

Shogi-Quest is a popular Shogi platform that adopts the Elo rating system to evaluate players' skill levels. On this platform, players can choose 2-minute, 5-minute, or 10-

minute games. These minutes are the thinking time per player, and when a player runs out of this time limit, he or she loses the game immediately.

To create Maia-S and Maia-S-Wide in Shogi, we collected 10-minute games and did some filtering on the game records, similar to Maia in Chess. The three conditions are explained as follows. (i) We eliminated games where players lost due to running out of the time. The reason for this was that there may be noisy behaviors specific to be losing the game by out-of-time, such as moving the piece that was easiest to operate. (ii) We eliminated games with a player rating difference of 50 or more. The reason for this was that rating difference could adversely affect the learning of the value function as well as the policy function, as the stronger player may win from an extremely disadvantageous situation, making the data noisy. (iii) We used the positions in which the number of moves was after the 50th move. The reason for excluding the early positions was that there are many similar positions in the early stages of the game, and having many similar data may harm the learning process.

We obtained 760,000 games that satisfied all the three conditions. We further divided the games into six groups so that each group has the same number of games. In more detail, for each game, we calculated the average rating of the two players. We then sorted all games according to the average ratings. The 1/6 of games with the lowest ratings became Group 1, the next 1/6 became Group 2, and so on. The rating range for each group was as follows.

- Group 1: R1433 – R1591
- Group 2: R1592 – R1655
- Group 3: R1656 – R1708
- Group 4: R1709 – R1768
- Group 5: R1769 – R1855
- Group 6: R1856 – R2140

As a result, we used about 120,000 games for training each Maia-S-1 and Maia-S-6. This is about 1% the number of games compared to Maia's 12 million. For Maia-S-Wide, we used all game records.

90% of data in each group were training data, 5% were validation data, and the remaining 5% were test data for evaluation. We performed multi-task supervised learning like Maia, in which the policy and value were simultaneously trained using a single network. We referred to the python-dlshogi2 library[10] for the network structure

[10]https://github.com/TadaoYamaoka/python-dlshogi2

and learning options. The major difference from the library was that we included past positions in the network's input instead of only inputting the current position. This is because in Maia's study, move-matching accuracy was significantly improved after including the recent history of 12 plies. In our preliminary experiment, a model that included the last 12 positions improved move-matching accuracy by 0.012 compared to a model based only on the current positions. Thus, we adopted the model that includes the last 12 positions. We performed 10 epochs of training for each group. This is because we observed that the loss often converged at around 10 epochs.

These Maia-S policies served as one of the two policies (say $\pi_1$) in our Blend model. For the other policy $\pi_2$, we employed an AlphaZero-based program, DLShogi[11]. Similar to the experiments on Chess in section 4, we used the policy of DLShogi's neural network and tuned the parameter $\alpha$ for the Blend model using each group's validation data.

### 4.6.2 Fundamental Analysis

In this section, we provide fundamental analyses of the experimental results. Figure 4.7 shows the move-matching accuracy of Maia-S-1, Maia-S-6, Maia-S-Wide, DLShogi, and the Blend models tested on the test data of all the six groups. The x-axis represents rating group and the y-axis represents the move-matching accuracy. The types of lines represent the types of the models. Specifically, the chain lines represent supervised learning policies such as Maia-S and Maia-S-Wide, the dotted line represents DLShogi, and the solid lines represent their Blend models. The colors of the lines represent the group of the human games used in the training data for the corresponding policy. Specifically, yellow represents using the training data for group 1 players (low-rated players), blue represents using the training data for group 6 players (high-rated players), red represents using the training data for group 1 to 6, and black represents DLShogi, which did not use human game records for its training data.

From the move-matching accuracy of Maia-S-1 (yellow chain curve) and Maia-S-6 (blue chain curve), we observed that Maia-S followed the same tendency as Maia in Chess: each model has the highest move-matching accuracy in a test group that is close to the trained group.

Maia-S-Wide (the red chain curve) had higher move-matching accuracy than Maia-S: 0.050 higher than Maia-S-1 for group 1 data, and 0.051 higher than Maia-S-6 for group 6 data. Maia-S-Wide had about the same move-matching accuracy regardless of the ratings that were covered by the training data. We considered such

[11] https://github.com/TadaoYamaoka/DeepLearningShogi

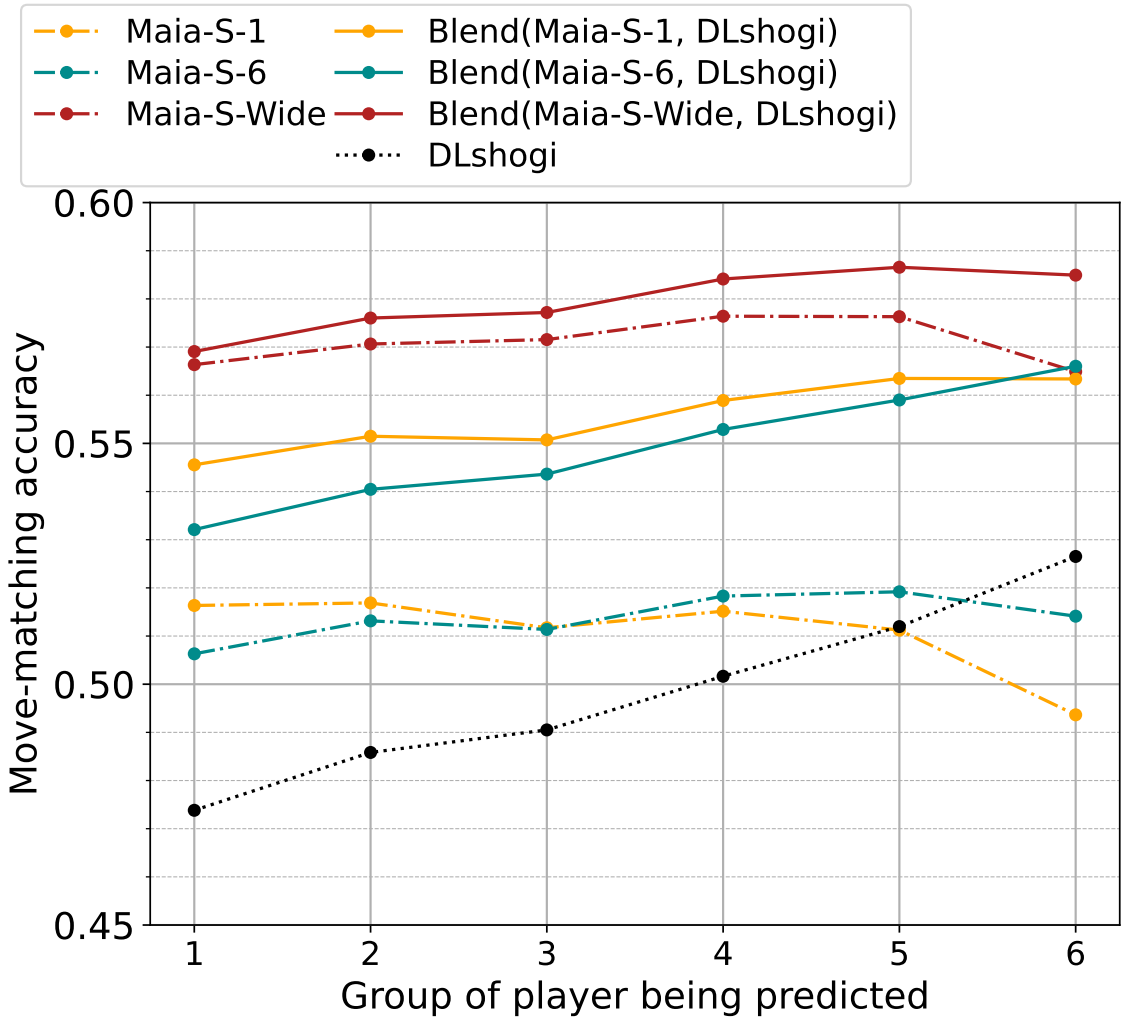


Figure 4.7: Move-matching accuracy of Maia-S models, DLShogi, and Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a group) is approximately ±0.002, calculated based on the test data.

improvement in accuracy to be contributed from the increase amount of training data for supervised learning.

The move-matching accuracy of DLShogi (the black dotted curve) increased as the rating of the test data increases, and it was higher than the move-matching accuracy of Maia-S-Wide for group 6 data. The results showed that DLShogi, which was trained without human game records, played less human-like, especially for human players with lower ratings.

The move-matching accuracy of the Blend (Maia-S-1, DLShogi) model (the yellow solid curve) was 0.029 higher than Maia-S-1 with the best $\alpha$ of 0.64 for group 1 data. The move-matching accuracy of the Blend (Maia-S-6, DLShogi) model (the blue solid curve) was 0.052 higher than Maia-S-6 with the best $\alpha$ of 0.48 for group 6 data. These results showed that as the player's rating increased, so did the effectiveness of the Blend model. This was also the same tendency as in DLShogi. We considered this was because we were able to increase the move-matching accuracy for the more difficult moves mentioned earlier.

The move-matching accuracy of the Blend (Maia-S-Wide, DLShogi) model (the red solid curve) was 0.003 higher than Maia-S-Wide with the best $\alpha$ of 0.89 for group 1 data, and was 0.020 higher than Maia-S-Wide with the best $\alpha$ of 0.63 for group 6. As the same tendency using Maia-S, although with a smaller increase in move-matching accuracy, the effectiveness of the Blend model also increased as the players' rating increased. We considered this was also because we were able to increase the move-matching accuracy for the more difficult moves mentioned earlier.

We investigated the robustness of the Blend model to $\alpha$, i.e., the influence of $\pi_1$. We targeted the Blend (Maia-S-Wide, DLShogi) model, which had the highest move-matching accuracy over the validation data in Shogi. Figure 4.8 shows the move-matching accuracy of the Blend (Maia-S-Wide, DLShogi) model for group 1 and group 6 data with different $\alpha$. The x-axis represents $\alpha$ and the y-axis represents the move-matching accuracy. The colors of the lines represent the players' group, i.e., the players' rating. Specifically, yellow represents the validation data for group 1 players (low-rated players), and blue represents the validation data for group 6 players (high-rated players). The dots represent the best $\alpha$ for the validation data. The best $\alpha$ for Group 1 was higher than Group 6. Namely, $\pi_1$, i.e., Maia-S-Wide, should be weighted more than DLShogi when predicting low-rated players' moves compared to high-rated players'.

Generally speaking, the appropriate $\alpha$ depends on the target of imitation. When we want to imitate stronger players, the reinforcement learning policy (DLShogi) should be weighted higher (i.e., smaller $\alpha$). Regarding the robustness, even if $\alpha$ deviates from the optimal value by 0.1, the move-matching accuracy only drops

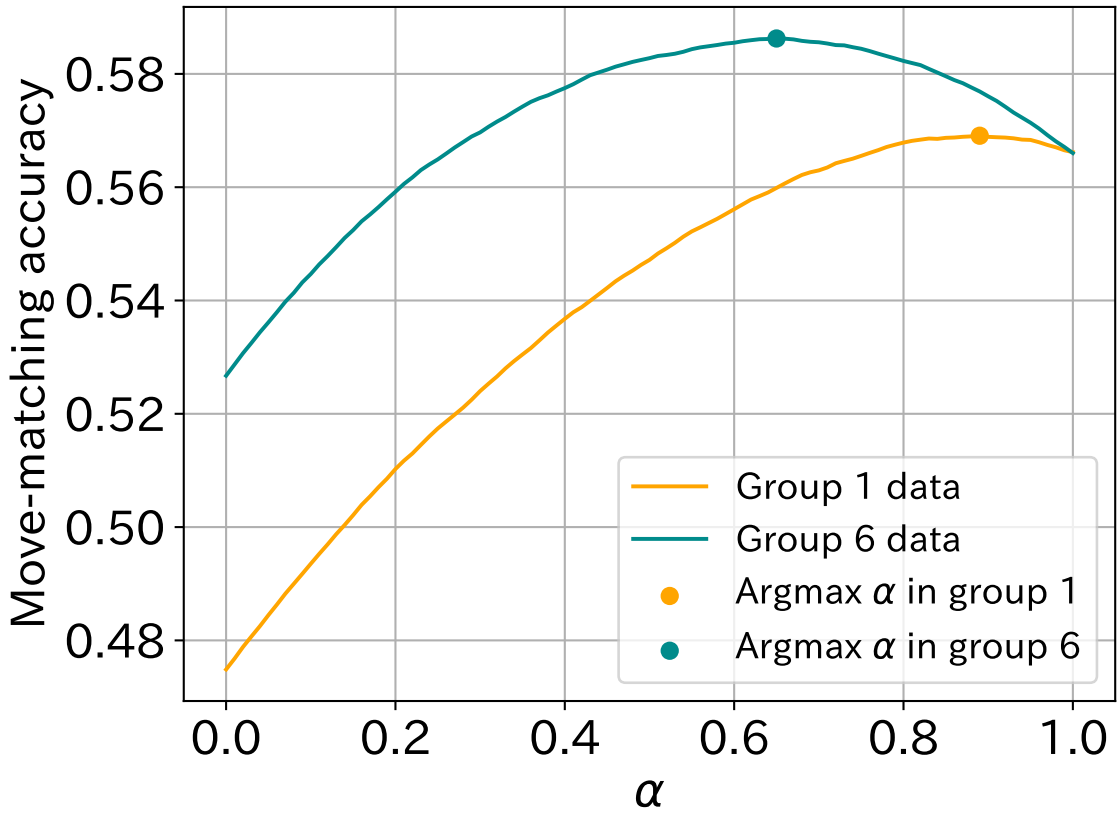


Figure 4.8: Move-matching accuracy of Blend (Maia-S-Wide, DLShogi) to group 1 and group 6 data for different $\alpha$ using validation data.

about 0.001 to 0.002. Therefore, $\alpha$ is not a very sensitive parameter, although it needs to be optimized depending on the target data.

### 4.6.3 Analysis of the Blend Model

In this subsection, we analyze the Blend model using shogi as the target domain. First, we analyze the relationship between the amount of training data and the performance of the Blend model based on its move-matching accuracy. For this purpose, we train a model using a further reduced subset of Maia-S, limiting the dataset to 10,000 positions. We call this model Maia-S-S. Next, we evaluate the playing strength of the Blend model to examine whether it achieves a level comparable to that of human players. Finally, to investigate the mechanisms underlying the effectiveness of the Blend model, we analyze in which positions accuracy is improved by the Blend model

#### Analysis of training data size

To investigate how the accuracies of the Blend models vary with the amount of training data, we conducted experiments using Maia-S-S trained on 10,000 games. Figure 4.9 plots the move-matching accuracy across different skill brackets for Maia-S-6, Maia-S-S-6, DLShogi, and their Blend models. Since Maia-S-S-6 is trained on approximately one-tenth the amount of data compared with Maia-S-6, it exhibits lower accuracy. The Blend model improves the move-matching accuracy for Group 6 by 5 percentage points over Maia-S, and by 16 percentage points over Maia-S-S. These results indicate that the Blend model is more effective when the amount of training data is limited.

#### Analysis of playing strength

Next, we measure the playing strength of the Blend model using win-rate loss, which quantifies how much win probability is decreased compared to the best move. we used Maia-S, DLShogi, and the Blend model to predict moves for test positions in Group 4 (n = 1000), and computed the decrease in win probability caused by each predicted move as the win-rate loss. For move prediction, we used the output of the neural network after applying the softmax function. It is known that sampling from the model's output distribution (temperature = 1) yields play weaker than the training player's level, while greedy selection (temperature = 0, argmax) is overly strong.

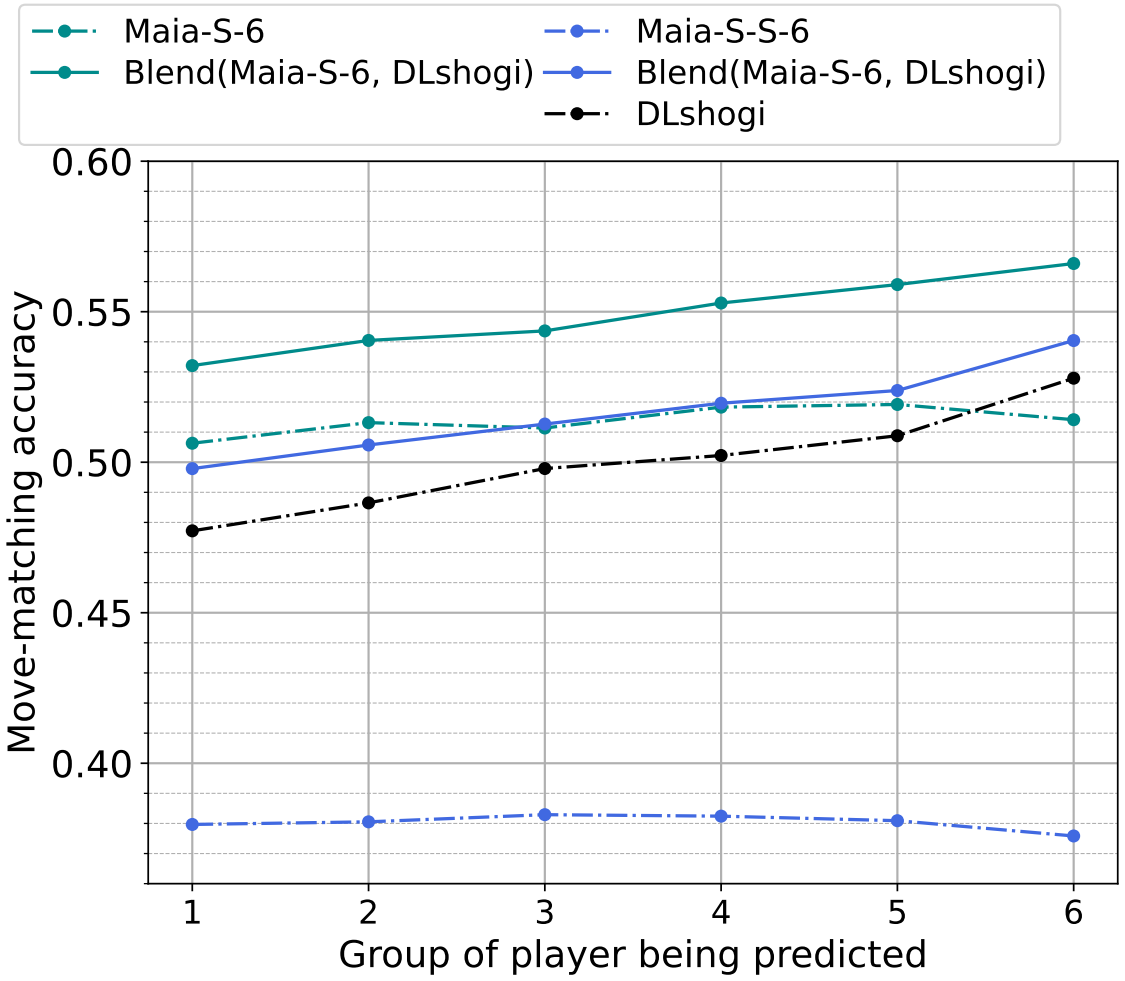


Figure 4.9: Move-matching accuracy of Maia-S-6, Maia-S-S-6, DLShogi, and the Blend models.

Therefore, the temperature parameter was searched from 0 to 1 in increments of 0.05. we selected the parameter that minimized the mean squared error with respect to the human win-rate loss on the validation data. For human players, the win-rate loss was computed based on their actual moves.

Figures 4.10, 4.11, and 4.12 show scatter plots of human loss versus the loss of Maia-S-Wide, DLShogi, and the Blend model, respectively. Since it is desirable for a model's loss to match human loss, an ideal plot would lie along the diagonal line. However, Maia-S-Wide tends to have many points in the upper-left region, whereas DLShogi and the Blend model tend to have many points in the lower-right region.

For Maia-S-Wide, the proportion of positions where the model loss exceeds the human loss by more than 0.05 is 0.062, while the proportion where it is smaller by more than 0.05 is 0.102. For the Blend model, the corresponding proportions are 0.041 and 0.125, respectively. For DLShogi, these proportions are 0.037 and 0.147, respectively.

Comparing these values, we find that Maia-S-Wide often makes mistakes in positions where humans do not, whereas DLShogi and the Blend model more often avoid making mistakes in positions where humans do make mistakes. We consider it more unnatural for a model to make mistakes in positions where humans do not, and therefore regard suppressing such behavior as more important. Thus, although the Blend model still has room for improvement in terms of playing strength, it is more effective at preventing situations in which the model makes mistakes despite humans not doing so.

**Analysis of the Blend model mechanism**

We also investigate the mechanism why the Blend model improves move-matching accuracy. We used test data from Group 6 in section 4.6.1, which had the highest move-matching accuracy and better blend effects. For our verification experiments, we randomly sampled approximately 9,000 positions where the final wins/losses were unknown[12].

Figure 4.13 shows the Venn diagram of the prediction results of all the experimented positions (n = 8926). The chain-line circle on the left contains positions that Maia-S-Wide's moves matched humans', which was 56.2% of the experiments positions. The solid-line circle on the right contains positions that the Blend model's moves matched humans', which was 58.4% of the experiments positions. In these

[12]The judgment was based on YaneuraOu + Suisho with 100,000 search nodes, the same setting as described in Appendix A.

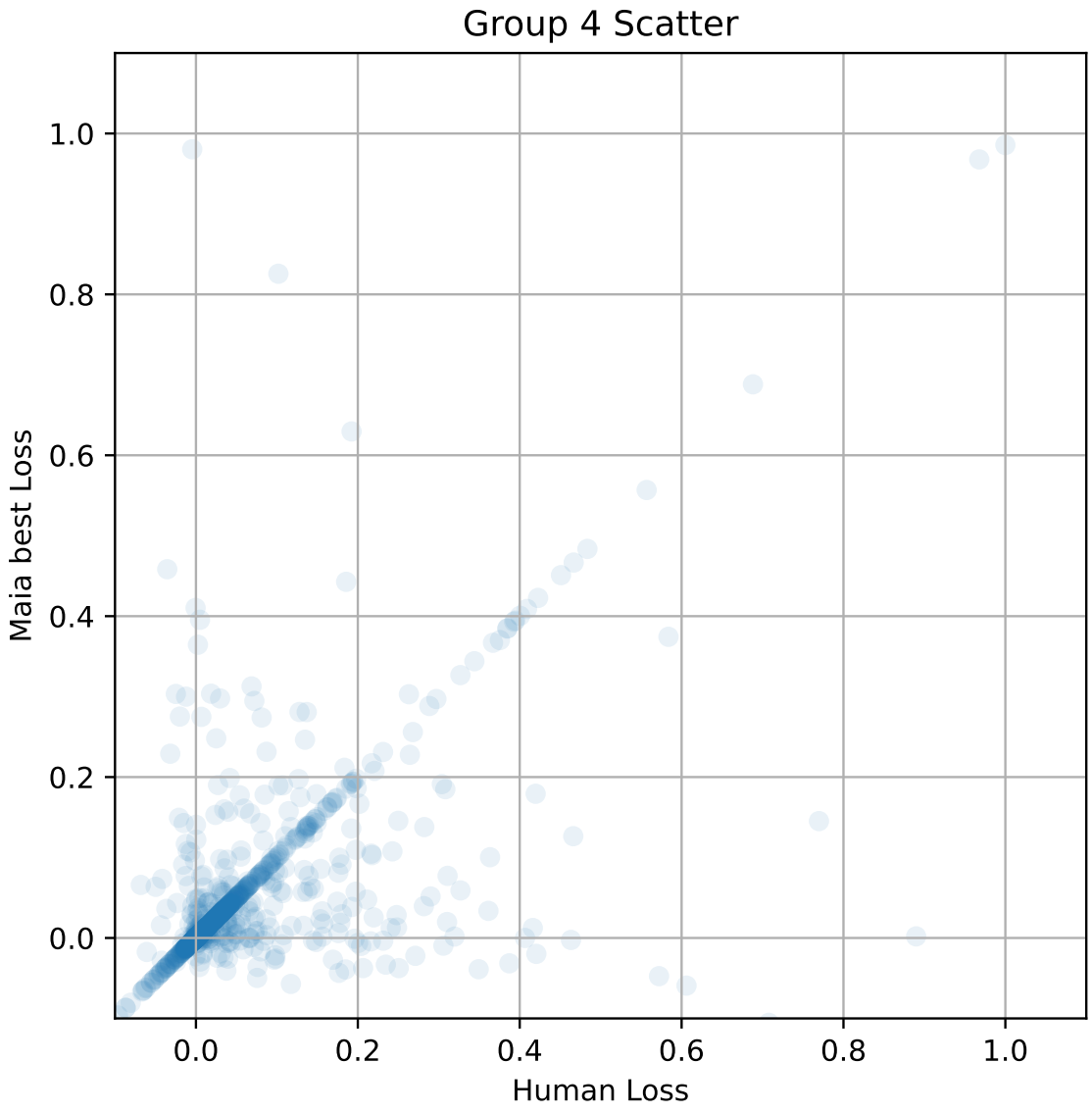


Figure 4.10: A scatter plot of human loss and Maia-S-Wide loss.

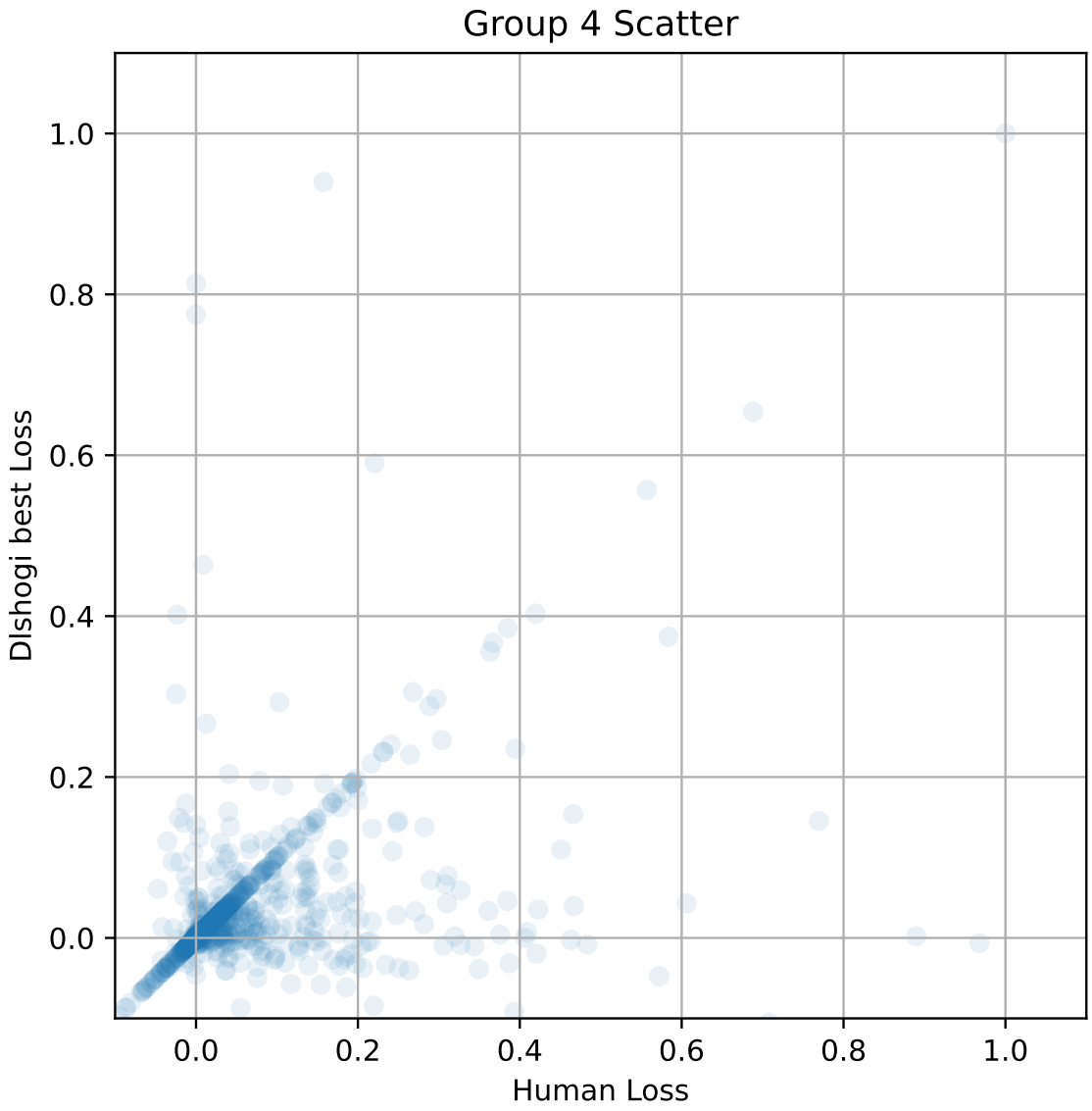


Figure 4.11: A scatter plot of human loss and DLShogi loss.

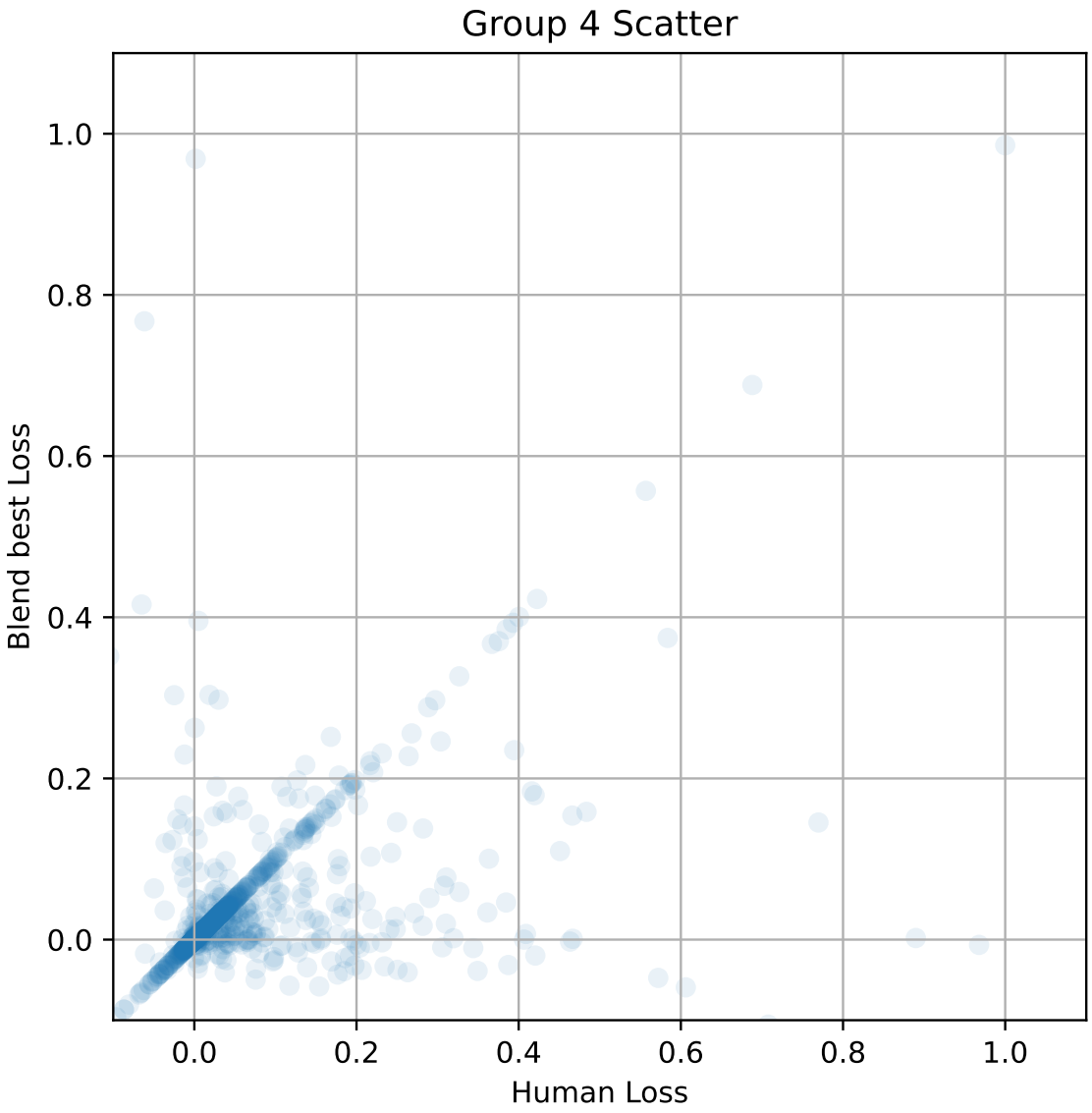


Figure 4.12: A scatter plot of human loss and the Blend model loss.

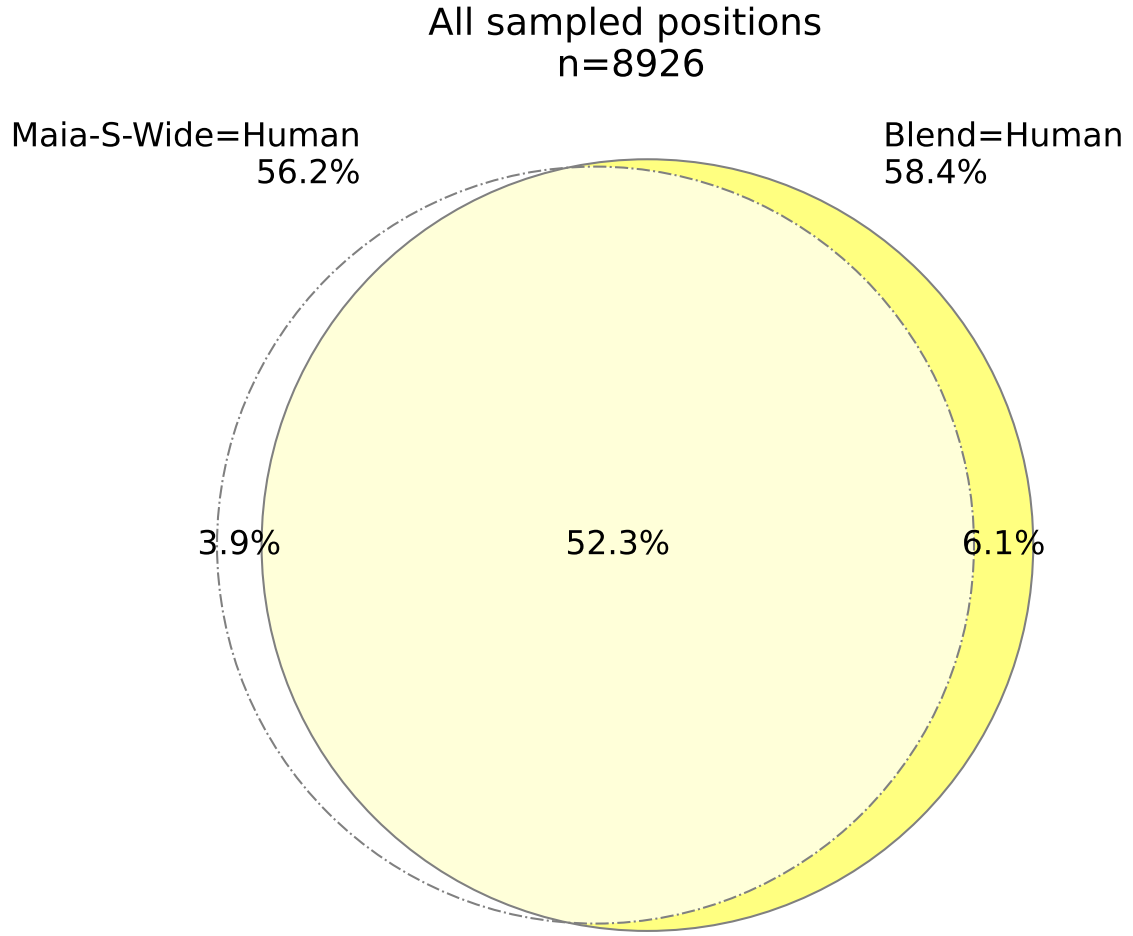


Figure 4.13: The Venn diagram of Maia-S-Wide and the Blend model's prediction results of the all experimented positions.

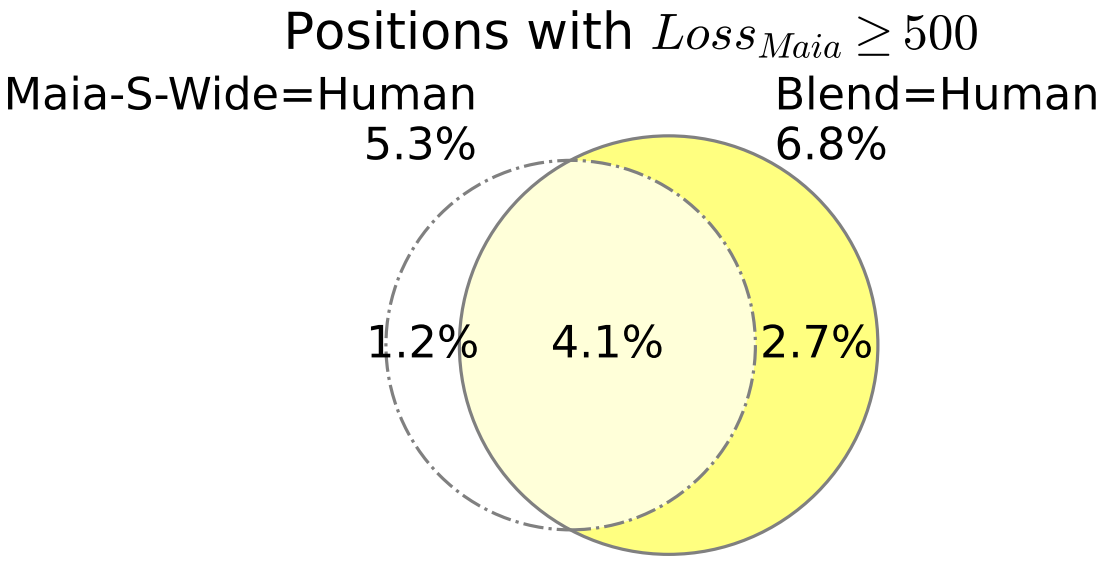


Figure 4.14: The Venn diagram of Maia-S-Wide and the Blend model's prediction results of the experimented positions with $Loss_{Maia} \geq 500$.

positions, the Blend model improved move-matching accuracy by 2.2% (58.4% minus 56.2%). The amount of improvement matched the results in Figure 4.7.

We further looked into these positions and the moves of Maia-S-Wide and Blend. Among moves that Maia-S-Wide predicted to play, we observed some bad moves that human players could easily avoid after a bit of searching (thinking). We assumed that the main reason for the improved move-matching accuracy of the Blend model was the ability to eliminate these bad moves. In other words, even if the Maia-S-Wide policy assigns the highest probability to such a bad move, if the AlphaZero-like DLShogi policy evaluates the move with a low probabiilty, the Blend model will also assign a low probability according to Formula (4.1) and choose another move. We hypothesized that move-matching accuracy for the Blend model would be improved in this way.

To verify our assumption, we defined the loss of a move and used it as an indicator to determine whether a move is bad or not. The detailed definition of loss is explained in Appendix A. The minimum value of a loss is 0. The larger the loss, the worse the move generally is. The average loss of Group 6 human players' moves was about 300. In this work, we used a threshold of 500 to indicate bad moves.

In the following, we focus on positions that Maia-S-Wide made bad moves, i.e.,

$Loss_{Maia} \geq 500$. Figure 4.14 shows the Venn diagram of the prediction results of the experimented positions with $Loss_{Maia} \geq 500$ (n = 1,466). Similar to Figure 4.13, the chain-line and solid-line circles contain positions that Maia-S-Wide's and Blend's moves matched humans', respectively. The 1,466 positions that Maia-S-Wide played bad moves were only 1/6 of the sampled positions. However, in these positions, the Blend model improved move-matching accuracy by 1.5% (6.8% minus 5.3%), which accounts for the majority of the 2.2% improvement shown in Figure 9.

From these results, we confirmed that avoiding Maia-S-Wide's bad moves contributed significantly to the Blend model's improvement in the move-matching accuracy.

## 4.7 Conclusion

Modeling human players' behaviors in games is a key challenge for making natural computer players, evaluating games, and generating content. Researchers have tried various methods to create human-like AI. For Chess, a supervised learning approach called Maia is known as one of the most successful ways. Maia employed a huge amount of training data (human game records) and achieved high move-matching accuracy without game tree search. For many games, however, it is hard to collect a similar amount of game records as in the case of Chess.

Our main contribution in this study was to propose a combination of supervised and reinforcement learning policies to compensate for the weakness of each other — supervised learning models, especially when using small- or medium-scale training data, sometimes play bad moves, and reinforcement learning models' moves sometimes are not human-like. More specifically, the proposed model, Blend, combined Maia-like supervised learning policies and AlphaZero-like reinforcement learning policy, and succeeded in significantly improving the move-matching accuracy in Chess, Go, and Shogi.

For Maia-like supervised learning policies, we created Maia-S following Maia's procedure but using only 1% of the game records. Our experiments showed that the Blend model improved the move-matching accuracy by 0.003–0.029 for intermediate players and 0.020–0.060 for advanced players. In more detail, the move-matching accuracy of the Blend model was 0.471 for Chess intermediate players (rating 1100–1199), 0.507 for Chess advanced players (rating 1900–1999), 0.510 for Go intermediate players (rank 3k), 0.534 for Go advanced players (rank 6d), 0.569 for Shogi intermediate players (rating 1433–1591), and 0.585 for Shogi advanced players (rating 1856–2140). The experiments also showed that the Blend model was robust to the parameter $\alpha \in [0, 1]$ — even when $\alpha$ deviated 0.1 from the best setting, the move-matching accuracy decreased only 0.001–0.002.

In addition, using Shogi as an example, we analyzed why the Blend model worked well compared to Maia-like supervised learning policies. We confirmed that blending helped prevent mistakes that humans very rarely make, but supervised learning does.

Some other worth-mentioning contributions are presented as follows. (i) The Blend model helped improve the move-matching accuracy over the original Maia trained using a huge amount of data. Our experiments in Chess showed improvements of 0.002–0.011. (ii) The Blend model had higher move-matching accuracy than KL-regularized search, one of the improvement methods of Maia. Our experiments in Go showed that the Blend model's move-matching accuracy was 0.004–0.105 higher. (iii) Among Maia-like policies, we showed that Maia-S-Wide, trained using six times as many games as Maia-S that contained a wider range of strength, had higher move-matching accuracy than Maia-S.

Regarding future research, we consider several directions. First, we plan to improve the Blend method to make it match more humans' moves. One possible way is to blend three or more policies, e.g., blending Maia-S, Maia-S-Wide, and AlphaZero-like policies. Another possible way is to look into the positions and moves that the Blend method cannot predict well, analyze the features of these positions or moves, and then come up with other methods to resolve the mismatches.

Second, we plan to apply the Blend method to other tasks. The concept of Blend is general and can be applied to one-player games, imperfect information games, real-time games, or even non-game tasks as long as supervised and reinforcement learning policies (probability distribution of actions) are available. To obtain supervised learning policies, one needs to prepare some training data. In the case of board games, at least a few thousand games may be required according to our experience. To obtain reinforcement learning policies, one needs to model the task as Markov decision processes.

Finally, it should be mentioned that although our paper aimed at obtaining human-like policy, we only evaluated its move-matching accuracy. It remains for future work to determine how human players can actually recognize the proposed method as human-like and whether the proposed method reaches the same skill level as the trained players.

# Chapter 5

# Human-Like Policies at Individual Level

## 5.1 Introduction

In recent years, game AI has made remarkable progress, and in two-player perfect information games such as Shogi, Go, and Chess, it has already surpassed top human players. However, from the perspective of satisfying humans as opponents, or teaching appropriate moves as instructors, there are still many open challenges. One reason is that most strong game AIs are designed to maximize their own win rate, rather than to play moves that resemble those chosen by humans. As discussed in Section 4.1, if we can build AIs that play human-like moves or accurately predict human moves, they can be applied as sparring partners, instructors, and commentators.

In this chapter, we further argue that, if we can build AIs that imitate the play style and move tendencies of specific *individual* players, these applications can be extended as follows.

- Match simulation: Simulated games against rivals or specific opponents, enabling preparation before actual matches.
- Personalized instruction: A model with a play style similar to one's own can suggest openings, strategies, and improvements tailored to the player.
- Advanced commentary: By predicting the moves that the actual players are likely to choose, the AI can provide commentary that more closely reflects the real game situation.

For these reasons, human-like policies, and especially imitation at the individual level, constitute an important research topic.

McIlroy-Young et al. showed that simply weakening a strong game AI, for example by using an early version of the training process or uniformly shallower search, does not significantly improve its ability to predict human moves [28]. They therefore proposed Maia, a Chess AI that aims to imitate human play: human game records are partitioned by rating range, and a separate model is trained for each range [28].

Building on Maia, which imitates humans at the rating-group level, McIlroy-Young et al. further proposed Transfer Maia [27], which fine-tunes the model on the game records of a target player in order to imitate that individual. Transfer Maia achieves a higher move-matching accuracy than Maia if 5,000 game records are available for the target player. However, when only 1,000 game records are available, the Maia model trained on the rating band containing the target player attains a higher move-matching accuracy than Transfer Maia.

If we assume that one game takes about 20 minutes, then playing 5,000 games requires roughly 1,000 hours, which corresponds to almost three years even if the player plays one hour per day. In games other than Chess, there are few players who have played more than 5,000 games, and even in Chess, many players have played fewer than 5,000 games. Therefore, it is desirable to develop more data-efficient methods that can imitate individuals using fewer game records.

In this study, we propose a method that fine-tunes not only on the target player's game records but also on similar game records, so that we can imitate individuals even when only a limited number of game records are available.

## 5.2 Related Work

### 5.2.1 Personalized Imitation Learning in Games

In this study, we consider models that imitate individual players in games by taking a position as input and predicting a probability distribution over moves. Such individual-level imitation models are derived from models that imitate humans at the population level, and population-level imitation models, in turn, are derived from research on predicting good moves from positions.

Silver et al. developed AlphaZero, a game AI that performs deep reinforcement learning from self-play without relying on human game records [41]. The neural network in AlphaZero takes a position as input and is trained in a multitask manner with a policy head that predicts the probability distribution over moves and a value head that predicts the winning probability. AlphaZero defeated existing top-level

AIs in three two-player perfect information games — Shogi, Go, and Chess — and has become a milestone in game AI research.

McIlroy-Young et al. built Maia, a Chess AI that has same network architecture as AlphaZero but learns from human game records and aims to imitate human play [28]. In Maia, human games are partitioned into nine rating bands, and a separate model is trained for each band. More concretely, game records by players rated 1100–1199 are used to train Maia 1100, and game records by players rated 1200–1299 are used to train Maia 1200, and so on.

Based on Maia, McIlroy-Young et al. created TransferMaia [27], a Chess AI that aims to imitate individual players by fine-tuning Maia using the game records of a target player. When 5,000 game records are available for the target player, Transfer-Maia predicts that player's moves with higher move-matching accuracy than Maia. However, when only 1,000 game records are available, Maia trained on game records from the target player's rating band achieves higher move-matching accuracy.

Regarding methods for computing similarity between game records, McIlroy-Young et al. also proposed a Chess player identification method that predicts who played a given game [29]. In their approach, given a game database annotated with player names, they learn an embedding function that maps game records by the same player to nearby vectors and those by different players to distant vectors. To predict who played a new game, they first embed the candidate players' games in advance, then embed the target game and select the player whose embedded game is most similar.

### 5.2.2 Learning from Limited Data

Methods for learning effectively from limited data have been widely studied not only in the domain of games, but also in robotics, computer vision, and machine learning more broadly. In robotics in particular, a significant body of work has been devoted to overcoming the simulation-to-real (Sim-to-Real) gap, which arises when models trained in simulation fail to generalize to real-world environments. A representative line of research focuses on Sim-to-Real transfer, where physical parameters in simulation are extensively varied to obtain policies that remain robust under real-world conditions. Tobin et al. proposed Domain Randomization, in which lighting conditions, textures, object shapes, and other factors are randomized during training, enabling object detection in the real world without using any real images [46]. These approaches demonstrate that robust policies can be built even when real-world data are extremely limited, making them important examples of methods designed for learning in low-data settings.

In computer vision, data augmentation is widely employed to maintain high performance when the amount of training data is limited. In addition to classical techniques such as flipping, rotation, and cropping, more advanced methods have been introduced — such as Mixup [56], CutMix [55], and AutoAugment [8] — which create synthetic samples or learn optimal augmentation policies. These approaches artificially expand data diversity and are highly effective at preventing overfitting, thus improving generalization in low-data regimes.

Meta-learning has also gained attention as a powerful framework for addressing the challenge of learning from few samples. Meta-learning aims to acquire "rapidly adaptable initial parameters" or "task-invariant structures" from a diverse set of tasks, enabling high performance on new tasks with only a small amount of data (few-shot learning). A representative method, MAML [9], learns an initialization that can be adapted to new tasks with only a few gradient steps, and has become one of the most widely used approaches for few-shot learning.

In summary, research aimed at building high-performance models in low-data environments has progressed across a wide range of fields, including robotics, computer vision, and meta-learning. These approaches tackle the limited-data problem from complementary perspectives: (1) generating diverse, task-relevant data through simulation, (2) artificially increasing data diversity through augmentation, and (3) pretraining parameters to enable rapid adaptation to new tasks.

## 5.3 Proposed Method

We propose to imitate individual players using not only the target player's own game records but also similar game records, so that imitation remains possible even when relatively few game records are available for the target player.

In the prior work Transfer Maia [27], the model aims to imitate individual players by fine-tuning Maia on the game records of the target player. When 5,000 game records are available for the target player, Transfer Maia achieves a higher move-matching accuracy than Maia. However, when only 1,000 game records are available, Transfer Maia's move-matching accuracy is lower than that of Maia. Even in Chess, there are not many players who have played 5,000 games, and this issue becomes even more severe in other games. Thus, methods that can imitate individuals from fewer game records are desirable.

To address this issue, we propose a method that performs fine-tuning using game records similar to those of the target player in addition to the target player's own records. The model can learn efficiently even when only a small number of game records are available. We refer to the proposed method as Similarity-Guided Fine-

Tuning (SGFT). In this work, we focus on three questions: how to measure similarity, how to perform fine-tuning, and how to evaluate the quality of individual-level imitation.

For similarity, we propose the following two approaches:

- A method that measures game-record similarity using game-specific features: we represent each game as a feature vector computed from the positions and moves so that it reflects playing style, and then measure similarity between these vectors.

- A method that measures game-record similarity using embedding representations: we represent each game as a vector using game embeddings trained for player identification, and then measure similarity between these embedding vectors.

An advantage of using game-specific features is that we deliberately adopt features that can express playing style, which may make it easier to imitate individual styles. On the other hand, it requires designing appropriate features for each game. In the embedding-based approach, such feature engineering is unnecessary and the method is more general and game-independent; however, learning good embedding representations requires a sufficient number of game records, and there is no guarantee that the resulting embeddings capture playing styles in a way that is intuitive to humans.

Regarding fine-tuning, we propose a two-stage approach in which the model is first fine-tuned on game records similar to those of the target player, and then further fine-tuned on the target player's own game records. In preliminary experiments, we attempted to fine-tune a model using a mixture of the target player's game records and similar game records, but this approach failed to imitate individuals effectively. One reason may be that, although the target player's own games should be more important than similar games, the model cannot distinguish them during training. We therefore propose a two-stage fine-tuning procedure in which the model is first fine-tuned on similar game records, and then fine-tuned on the target player's own game records.

For evaluation, there is still no consensus on how to measure the quality of individual-level imitation. Most prior work evaluates models using move-matching accuracy, but even if accuracy improves by a few percentage points, the resulting model may still have a very different playing style. Thus, move-matching accuracy alone is not sufficient. In this study, we therefore propose to evaluate individual-level imitation using multiple features computed from positions and moves that can capture the playing style.

## 5.4 Experiments

In this chapter, we evaluate the ability of the proposed Similarity-Guided Fine-Tuning (SGFT) method to imitate individual players from a small number of target player's game records. SGFT extracts game records similar to the target player and fine-tunes the model using their game records. We compare two ways of computing similarity: a feature-based similarity and an embedding-based similarity. We also compare one-stage fine-tuning with two-stage fine-tuning, where the model is fine-tuned first on similar game records and then on the target player's game records. Furthermore, in addition to the conventional move-matching accuracy, we evaluate how much closer the individual-level model gets to the target player than the group-level imitation model at the feature level. We consider this feature-level evaluation to be important because move-matching accuracy alone makes it difficult to assess whether the model's predicted moves are stylistically similar to the target player's moves or fundamentally different.

### 5.4.1 Experimental Setup

#### Dataset

We use game records from the online Shogi platform "Shogi Quest." Using Figure 5.1, we first describe the dataset used for each model, and then explain the model configurations.

We first outline how we split the game records for training. In SGFT, we use a pre-trained Maia-S network as the base model before fine-tuning, and an embedding network to compute similarity. Because we would like to compare SGFT models trained with different numbers of game records per target player, we need to use players who have as many games as possible. Thus, the SGFT models are trained on a fixed number of game records from the top 100 players in terms of number of games. The embedding network used in SGFT requires that each player has a sufficient number of games, so we train it on the game records of players ranked 101st–2000th in number of games (about one million games in total). The base model Maia-S is trained on the game records of players ranked below 2001st (about 1.7 million games in total). Some game records belonging to players ranked within the top 100 are not used by SGFT; for fairness, we do not use those leftover games for other training purposes.

We now explain how we split the game records into training, validation, and test sets for each player. We first split all game records into training (80%), validation (10%), and test (10%) sets. We then apply this split to each player. In this way, we

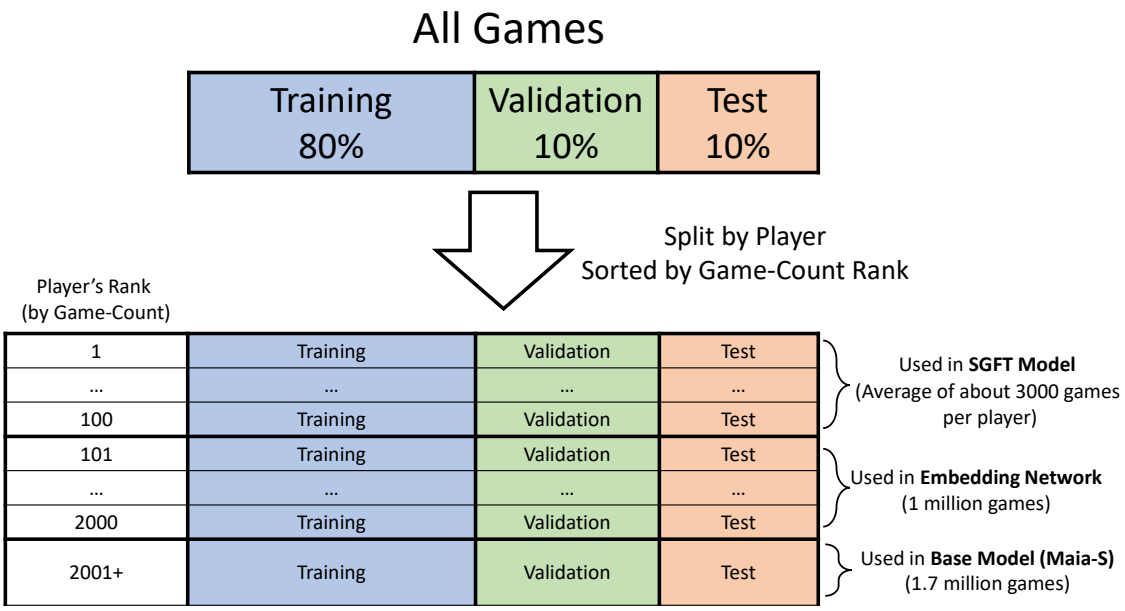


Figure 5.1: Game splitting procedure

avoid situations where a game is used as a training example for one player but as a test example for their opponent, which makes the comparison fairer.

### Base Model Configuration

As the base model before fine-tuning, we adopt Maia-S, which has same network architecture as AlphaZero but learns policies and value functions from game records. We train it on about 1.7 million game records from players ranked below 2001st in number of games. In the prior work Transfer Maia, the base model Maia-1900 was trained on 12 million game records. Although our data size is roughly one tenth of that, Maia-S still achieves reasonable performance — 56% move-matching accuracy — in predicting human moves, and so we do not regard this data size as too small.

### Embedding Network Configuration

As the embedding network for computing similarity, we adopt the network architecture used for player identification [29]. In this network, we construct pairs of pre-move and post-move positions from the moves in each game. Each pair is input into a network that has a similar architecture to Maia-S but without the policy and value heads. The outputs corresponding to each move are then concatenated into a sequence and used as the input to a Transformer. We train this network on the game records of players ranked 101st–2000th in number of games.

### Similarity-Guided Fine-Tuning Model Configuration

In the SGFT models, we start from Maia-S as the base model and fine-tune it using the target player's game records and similar game records. We use the game records of the top 100 players in terms of number of games for training the SGFT models. For training data, we use 100 games for each target player. In this study, we refer to the number of game records from the target player as the *query size*, and the number of similar game records used for training as the *reference size*. For validation, we use 10 games, which is one tenth of the training size. For testing, we use 10% of all games of the target player as test data. In our experiments, we compare conventional fine-tuning, which trains on a mixture of the target player's game records and similar game records in a single stage,[1] with two-stage fine-tuning, which first fine-tunes on similar game records and then further fine-tunes on the target player's game records. We also compare feature-based and embedding-based

[1]In one-stage fine-tuning, one could also assign larger weights to the target player's games to emphasize them.

similarity measures between game records. In the feature-based case, the feature vectors are relatively low-dimensional, so we measure similarity as negative Euclidean distance. In the embedding-based case, the vectors are high-dimensional, so we use cosine similarity. The feature-based similarity method and the features used for evaluation are summarized in Table 5.1.

### 5.4.2 Experimental Results

#### Evaluation by Move-Matching Accuracy

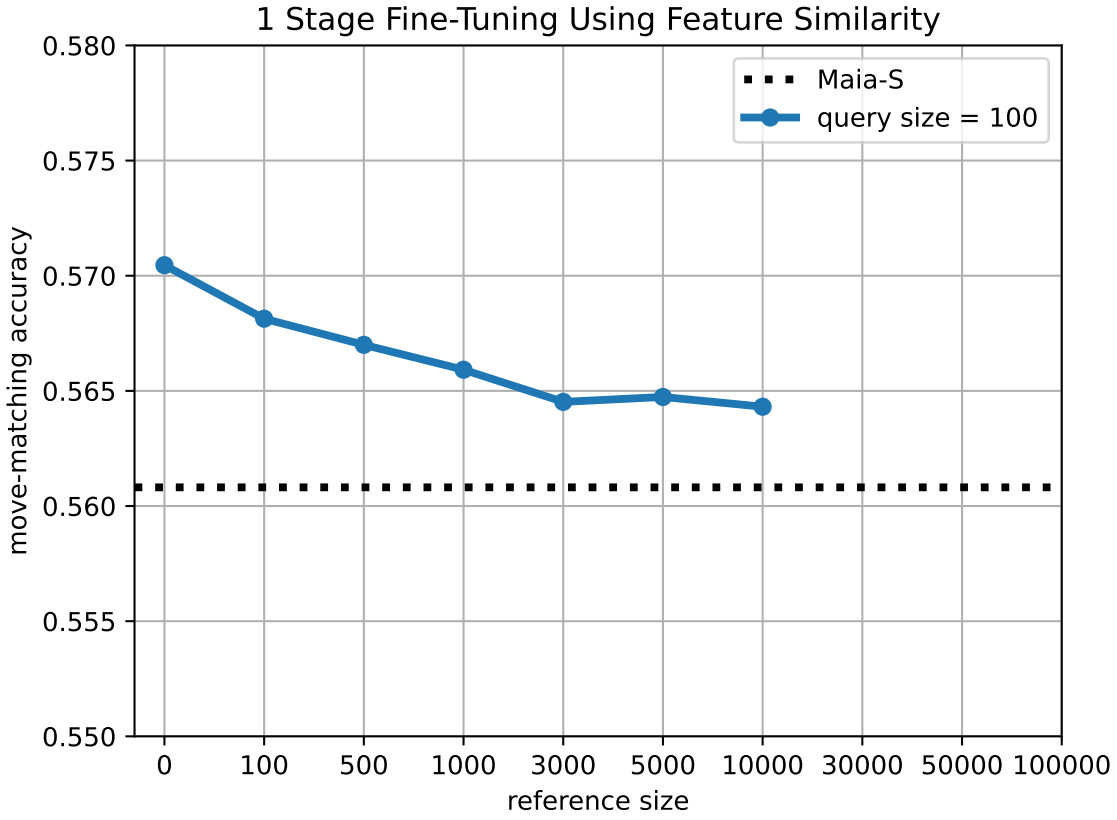


Figure 5.2: Results of one-stage fine-tuning using feature-based SGFT

Figures 5.2 and 5.3 show the results of one-stage fine-tuning using feature-based SGFT. When the reference size is 0, i.e., when the model is trained only on 100 games from the target player (the conventional method), the move-matching accuracy is the

Table 5.1: List of features and their descriptions

| **Feature Name** | **Description** |
|---|---|
| control_cnt | Total number of squares controlled by the player's pieces on the board. |
| own_piece_control_cnt | Total number of controls exerted by the player's pieces on their own pieces on the board. |
| opponent_piece_control_cnt | Total number of controls exerted by the player's pieces on the opponent's pieces on the board. |
| own_side_control_cnt | Total number of controls exerted by the player's pieces on squares in their own side. |
| opponent_side_control_cnt | Total number of controls exerted by the player's pieces on squares in the opponent's side. |
| king_surround1_control_per_sq | Average number of controls per square within one square of the king. |
| king_surround2_control_per_sq | Average number of controls per square within two squares of the king. |
| king_surround1_piece_per_sq | Average number of pieces per square within one square of the king. |
| king_surround2_piece_per_sq | Average number of pieces per square within two squares of the king. |
| hand_cnt | Total number of pieces in hand. |
| drop_own_side_cnt | Number of drops made into the player's own side. |
| drop_opponent_side_cnt | Number of drops made into the opponent's side. |
| cover_ratio | Ratio of squares on the board that are controlled by at least one of the player's pieces. |
| {PIECE} | Frequency with which piece {PIECE} is moved. |

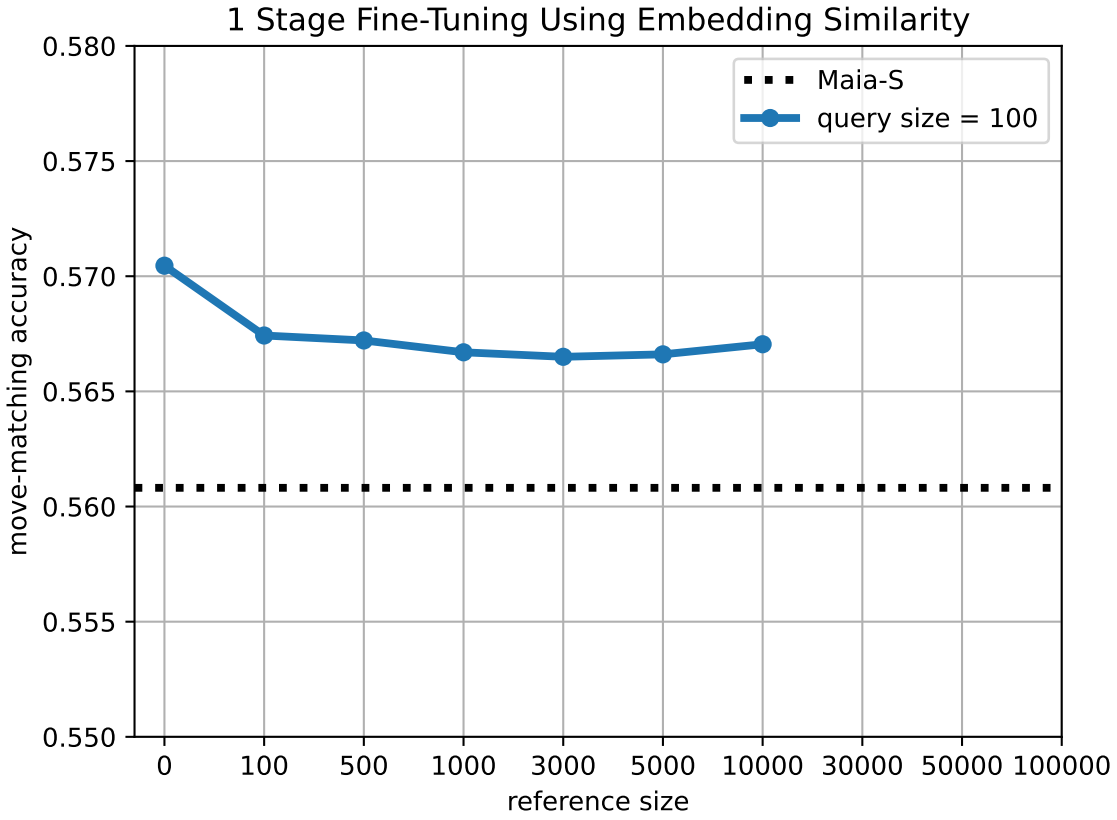


Figure 5.3: Results of one-stage fine-tuning using embedding-based SGFT

highest, and the accuracy decreases as the reference size increases. Thus, in the one-stage fine-tuning setting, similar game records have a negative effect when evaluated by move-matching accuracy. We suspect that one reason is that the similar game records differ from the target player's own style and are of relatively lower quality, so training on them simultaneously has an adverse effect.

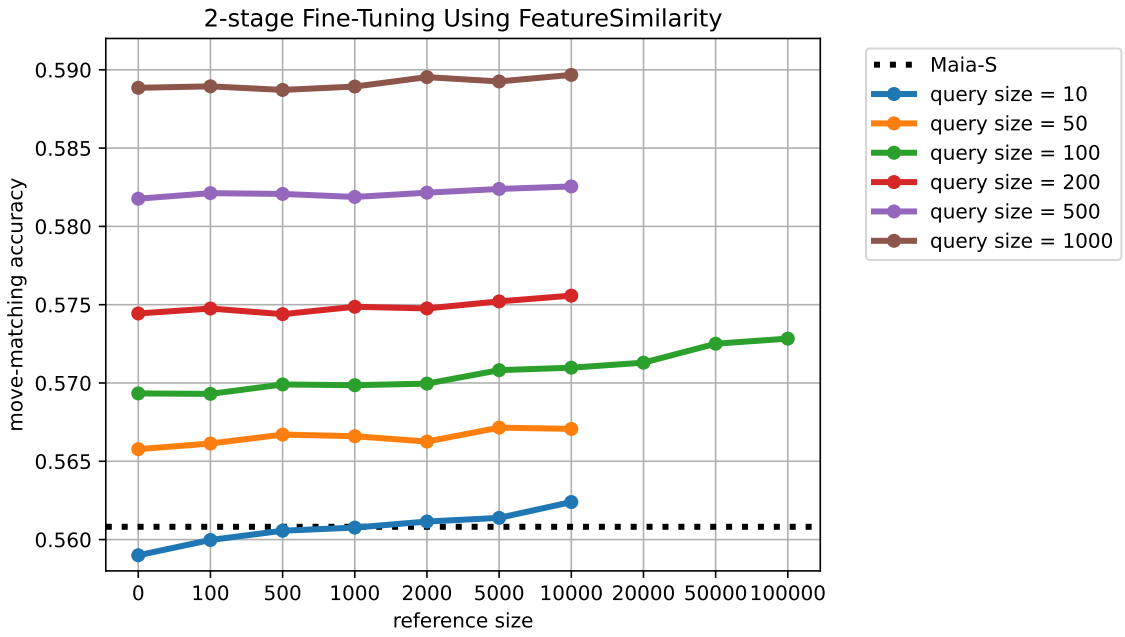


Figure 5.4: Results of two-stage fine-tuning using feature-based SGFT

Figure 5.4 shows the results of two-stage fine-tuning using feature-based SGFT. As the query size, i.e., the number of game records of the target player, increases, the move-matching accuracy also improves. Focusing on the plots with a query size of 100 or fewer games, we can see that the proposed method improves the move-matching accuracy. As the reference size increases, the accuracy becomes higher; in particular, for the plots with a query size of 100, the accuracy continues to improve even when the reference size reaches 100,000, and further improvement can be expected with larger reference sizes. These results indicate that, when performing two-stage fine-tuning, conducting an initial fine-tuning step using similar game records has a positive effect.

Figure 5.5 shows the results of training with embedding-based SGFT using two-stage fine-tuning, where the similarity measure is replaced with an embedding-based one. Compared with feature-based SGFT, the improvement in move-matching accuracy is larger, and improvements can be observed even when the query size is 500

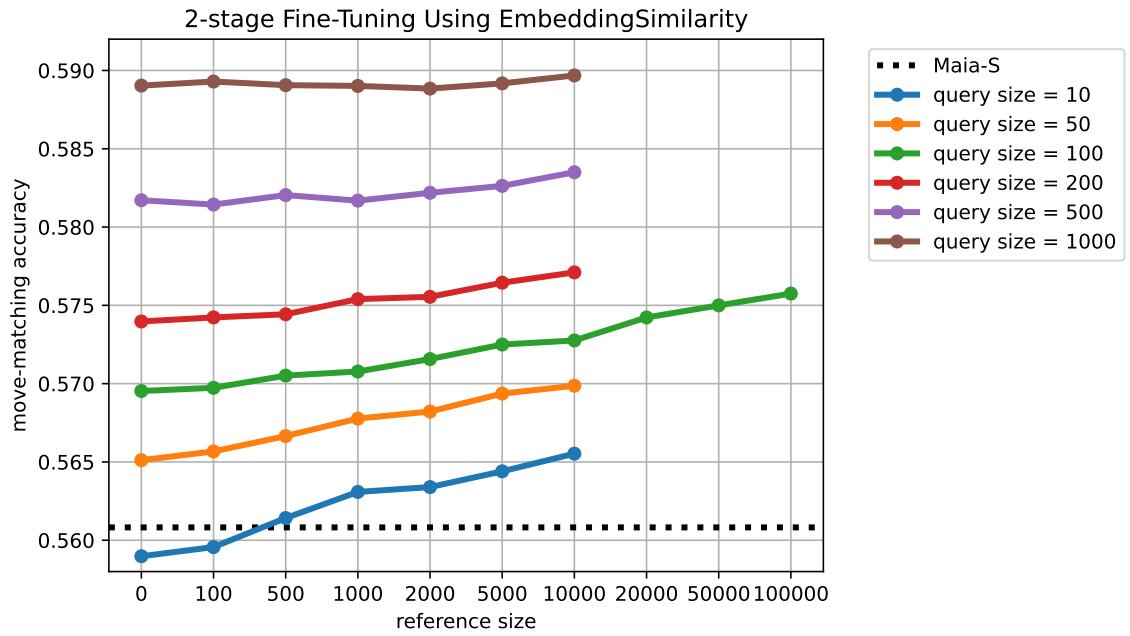


Figure 5.5: Results of two-stage fine-tuning using embedding-based SGFT

or fewer games. As the reference size increases, the accuracy becomes higher; in particular, for the plots with a query size of 100, the accuracy continues to improve even when the reference size reaches 100,000, and further improvements are expected with larger reference sizes.

### Evaluation by Features

In this section, we evaluate each model using feature-based metrics to assess stylistic similarity, which may be difficult to capture using move-matching accuracy alone. From the set of features, we focus on three features with different characteristics: drop_own_side_cnt, drop_opponent_side_cnt, and control_cnt. The evaluated models include Maia-S, Transfer Maia, SGFT-E, and SGFT-F, as well as SGFT-F-One, which applies SGFT-F using only a single feature. All models use a query size of 100 and a reference size of 10,000.

Figure 5.6 shows plots of drop_own_side_cnt values for each model, sampled from 10 players. Each color corresponds to a target player. While this feature exhibits some variation across target players, Maia-S shows almost no such variation, and its proportion is less than half that of the target players. In contrast, Transfer Maia,

| | Maia-S | Transfer Maia | SGFT-E | SGFT-F | SGFT-F-One |
|---|---|---|---|---|---|
| drop_own_side_cnt | 0.214 | 0.093 | **0.090** | 0.096 | 0.094 |
| drop_opponent_side_cnt | 0.224 | 0.101 | **0.098** | 0.103 | 0.241 |
| control_cnt | 0.134 | 0.143 | 0.143 | 0.144 | **0.116** |

Table 5.2: Feature value RMSE of each model. SGFT-F-One refers to SGFT-F-One (Target Feature).

SGFT-E, SGFT-F, and SGFT-F-One (drop_own_side_cnt) produce values that are relatively close to those of the target players. In other words, they largely preserve the target players' relative ordering: players with higher values among the targets also tend to have higher values under these models, and players with lower target values tend to remain lower. These results indicate that the models successfully imitate this feature. Moreover, the other SGFT-F-One variants emphasize non-target features. As a result, they fail to collect game records that are similar to the target in terms of the target's features. Consequently, their performance does not improve upon Maia-S and can even deteriorate.

Table 5.2 reports the RMSE between human features and model features for these metrics after standardization. SGFT-E achieved the best performance in terms of move-matching accuracy. We initially expected SGFT-F to be promising for aligning feature-level behaviors; however, the results in Table 5.2 do not indicate that it matches the target players more closely than SGFT-E. We also experimented with selecting games by focusing on a single feature, hoping to at least improve alignment on that feature alone. While this strategy occasionally worked as intended, it was not consistently successful; moreover, other features did not become more similar, making the approach impractical in practice.

## 5.5 Conclusion

In this study, we aimed to imitate individual players from a small number of game records and proposed a method that fine-tunes models using game records similar to those of the target player. In particular, we proposed methods for computing similarity using game-specific features and embeddings, a two-stage fine-tuning procedure, and an evaluation metric for assessing individual-level imitation.

Our results showed that, in terms of move-matching accuracy, two-stage fine-tuning outperforms one-stage fine-tuning. We also found that embedding-based SGFT achieves better performance than feature-based SGFT. When evaluated using

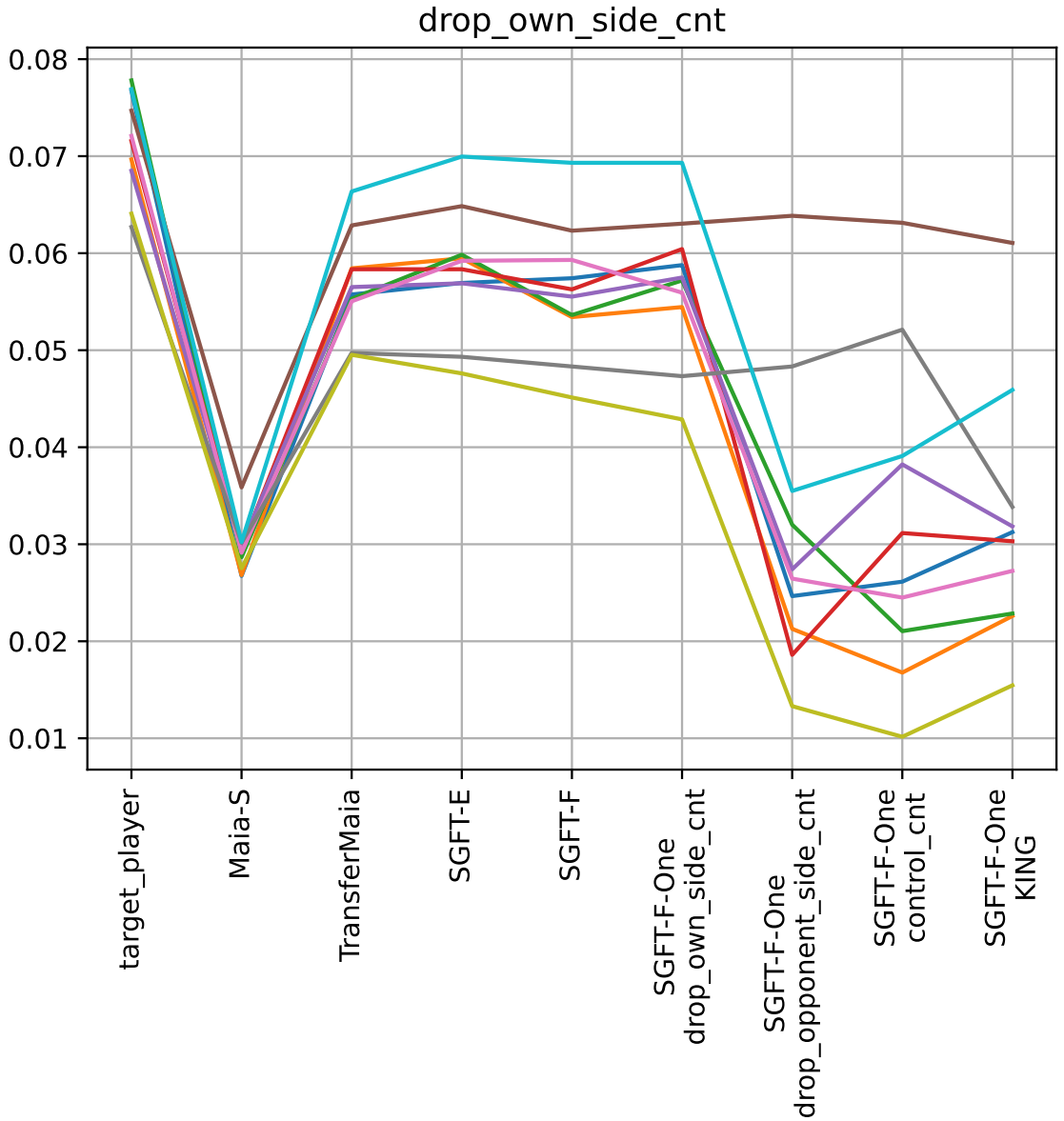


Figure 5.6: Evaluation of each model by drop_own_side_cnt.

features, we observed that many indicators that should be important for playing style show little improvement under existing methods.

Even when data from the target player cannot be prepared in advance, it may still be possible to achieve more accurate predictions. For example, to improve predictive accuracy during an ongoing game, it may be effective to exploit assumptions such as the following: (i) a player's style can be inferred to some extent from the moves played so far; (ii) if the opponent is operating under a misunderstanding, they are likely to continue acting on it; and (iii) if a sequence of moves follows a particular context (e.g., an attack), subsequent moves are likely to remain consistent with that context.

# Chapter 6

# Applications of Human-Like Policies

## 6.1 Introduction

As discussed in Chapters 4 and 5, once human-like policies are achieved, they can be applied in various ways. For example, the following applications become possible:

- As an opponent: Instead of overwhelming human players, the AI can provide a suitable level of challenge for practice by choosing moves that humans are likely to play.
- As a tutor: The AI can suggest good moves that humans are likely to consider, rather than moves that are too difficult to discover during actual play.
- As a commentator: Rather than predicting the moves and win rates of a strong game-playing AI, the system can predict the moves and win rates that a human would likely produce, enabling commentary that better reflects real human play.

Furthermore, if we can develop an AI that mimics the play style and move tendencies of a specific individual player, its applications can be expanded even further.

- Match Simulation: Enables simulated games against rivals or specific players, allowing preparation before actual matches.
- Personalized Training: A model with a play style similar to the user can provide tailored strategies and point out areas for improvement.

- Advanced Commentary: By predicting the moves that a particular player would likely choose, the AI can offer commentary that more closely reflects real game situations.

In this chapter, as one application of human-like policies, we use models that imitate human behavior at the rating-group level to predict human win rates. If win rates can be estimated accurately, the AI can assess actual game situations more precisely, which is beneficial for both tutoring and commentary. Although this chapter uses human-like policies at the group level, we expect that employing human-like policies at the individual level would further improve prediction accuracy.

In two-player perfect-information games such as Shogi, Go, and Chess, artificial intelligence (AI) has already surpassed top human players in playing strength. Consequently, there has been active research on utilizing strong game AI for teaching and game analysis support, as well as on developing game AI specifically tailored for teaching and spectating support. From the perspective of instruction and spectating, it is not only useful to know the best move for strong game AI, but also to know which moves human players are likely to choose and what the resulting win rate would be. Since strong game AIs are trained to maximize their own winning probability, they do not necessarily predict human moves or human win rates accurately. For example, consider a situation in which there exists a move that is extremely difficult for a human player to find, but once played, the winning rate becomes close to 100%. In such a case, the win rate for a strong game AI may be close to 100%, whereas for a human player, reaching that position itself is difficult, and thus the actual human win rate may not be close to 100%.

In recent years, there has been active research on predicting human moves [28, 27, 23, 30], whereas many open challenges remain in predicting human win rates. In this study, focusing on Shogi, we first investigate to what extent existing methods such as reinforcement learning methods like AlphaZero [41] and supervised learning methods [28] can predict human win rates. We then propose a win rate prediction method that combines win rate prediction with move probability distributions so that the predicted win rate reflects the ease with which humans play each move.

## 6.2 Related Work

AlphaZero [41] is a reinforcement learning method that constructs a strong game AI solely from self-play results, without using human game records. AlphaZero learns the win rate and move probability distribution from game outcomes and search results at each position.

McIlroy-Young et al. pointed out that the moves played by strong game AI such as AlphaZero, as well as by weaker AI obtained by taking snapshots of intermediate checkpoints during training, differ from human intuition [28]. They further proposed Maia, which has the same network architecture as AlphaZero but is supervised training on human game records, using positions together with human moves and game outcomes. Maia is trained separately for each rating range, and each model learns the characteristics of its rating band from 12 million game records.

Ogawa et al. showed that, when only about 1% of the game records used in Maia are available, a Maia-S-All model that does not split training data by rating band predicts human moves more accurately than Maia [30]. They also pointed out that the supervised Maia-S-All model sometimes plays blunders that humans rarely choose. To address this, they proposed a method that mixes the move probability distributions of Maia-S-All and DLShogi, the latter being trained via reinforcement learning in a manner similar to AlphaZero, so that moves can be predicted more accurately for target human strength. Under their experimental setting, the Blend model achieves a human move-matching accuracy that is higher than Maia-S-All by about 0.003 to 0.020 for each rating band. Similarly, the Blend model outperforms DLShogi by about 0.050 to 0.100 in move-matching accuracy across rating bands.

As a study that converts the win rate predicted by AlphaZero into a win rate more suitable for human-versus-human games, the work by Hiraoka can be mentioned [57]. Hiraoka applied temperature scaling to the win rate predictions of AlphaZero, which often take values close to 0 or 1, in order to calibrate them toward 0.5, and proposed a method that determines the degree of calibration using the entropy of ownership.

## 6.3 Proposed Method

The author focuses on the fact that, in many situations where the predicted win rate for human players differs from human intuition during spectating or post-game analysis, the move probability distribution also diverges from human perception. The authors therefore hypothesizes that, when predicting human win rates, incorporating a move probability distribution that can reasonably predict human moves will lead to more accurate win rate predictions. In this study, as the simplest way of combining a move probability distribution $\pi$ and a win rate prediction $V$, we propose an inner product model. The reason for taking an inner product is that, under the assumption that human players choose moves according to the move probability distribution, the resulting win rate should appropriately reflect the predicted win rates at the successor positions.

In the inner product model $V_{ip}(\pi, V)$, for each legal move *move* at the root position, we infer the move probability $\pi_{move}$ and the predicted win rate $V_{move}$ at the successor position, and then take their inner product as in Eq. (1):

$$V_{ip}(\pi, V) = \sum_{move} (\pi_{move} \times V_{move}). \tag{6.1}$$

One advantage of the proposed method is its computational efficiency. The model only requires the move probability distribution at the root position and the predicted win rates at positions reached after one move from the root. A mini-batch that includes these inputs fits within the memory of a typical consumer GPU, and thus all outputs can be obtained with a single forward pass.

## 6.4 Experiments

### 6.4.1 Common Experimental Settings

For the models and data that output move probability distributions and win rate predictions, we follow the settings used in the work by Ogawa et al. [30]. A major difference is that, while the supervised learning models in Ogawa et al. are trained only on positions from move 50 onward, the supervised learning models used in this study are trained starting from the initial position. The models are: (1) Maia-S-All, trained in a supervised manner; (2) DLShogi, trained via reinforcement learning in a manner similar to AlphaZero; and (3) the Blend model, which mixes the move probability distributions of Maia-S-All and DLShogi. Specifically, Maia-S-All provides both move probability distributions and win rate predictions, DLShogi provides only win rate predictions, and the Blend model provides only move probability distributions.

The training data for Maia-S-All consist of game records from Shogi Quest. Approximately 720,000 games are divided into six groups by player rating. The group with the lowest ratings is referred to as "Group 1," the next as "Group 2," and so on, with the highest-rating group referred to as "Group 6." For each group, 90% of the game records are used as training data, 5% as validation data, and 5% as test data. For the test data, only a single position from move 40 onwards is used from each game. The reason is that, for a given game record, the game outcome is the same for all positions, and repeatedly evaluating multiple positions from the same game could bias the results. Moreover, positions before move 40 generally correspond to the opening or early middle game, which we consider less suitable for evaluating win rate prediction.

In this study, we use three evaluation metrics for win rate prediction: result accuracy, cross entropy, and expected calibration error. In the following subsections, we discuss the experimental results based on each metric.

### 6.4.2 Result Accuracy

Result accuracy is defined as the agreement rate between the predicted win rate $\hat{V}$ and the final game outcome: if the side to move finally wins and $\hat{V} > 0.5$, or if the side to move finally loses and $\hat{V} < 0.5$, the prediction is counted as correct. In particular, for spectating, only one outcome is observed in each game. If the predicted win rate in the late stage of the game disagrees with the actual result, the prediction becomes harder to accept, and thus result accuracy is one of the important metrics.

Figure 6.1 shows the result accuracy of the existing models and the inner product models using the Blend model for each rating group. The line type indicates the type of model: dashed lines represent existing models, and solid lines represent inner product models that use the Blend model as the move probability distribution. The line color indicates the type of win rate predictions used: blue lines indicate the use of Maia-S-All win rate predictions, and red lines indicate the use of DLShogi win rate predictions. The same line types and colors are used consistently in the subsequent figures.

We compare the win rate predictions of the existing models (dashed lines) with those of the inner product models using the Blend model (solid lines). Among the models that use Maia-S-All win rate predictions (blue), the inner product model achieves a higher result accuracy by about 0.2 to 0.8 percentage points. Among the models that use DLShogi win rate predictions (red), the inner product model achieves a higher result accuracy by about 0.1 to 0.8 percentage points.

Comparing the win rate predictions of the existing models, namely Maia-S-All (blue dashed line) and DLShogi (red dashed line), we observe that DLShogi predicts game outcomes more accurately, with result accuracy higher by about 0.8 to 1.8 percentage points. In move prediction, Maia-S-All predicts human moves more accurately than DLShogi, so it is somewhat surprising that in win rate prediction DLShogi produces more accurate human win rate predictions than Maia-S-All. One possible reason is that the number of game records used to train Maia-S-All is smaller than that of DLShogi, and thus the amount of data may be insufficient for Maia-S-All to generalize well.

Overall, result accuracy tends to be lower for higher rating groups. The author guesses that this is because positions become more complex as player strength increases, making outcome prediction more difficult.

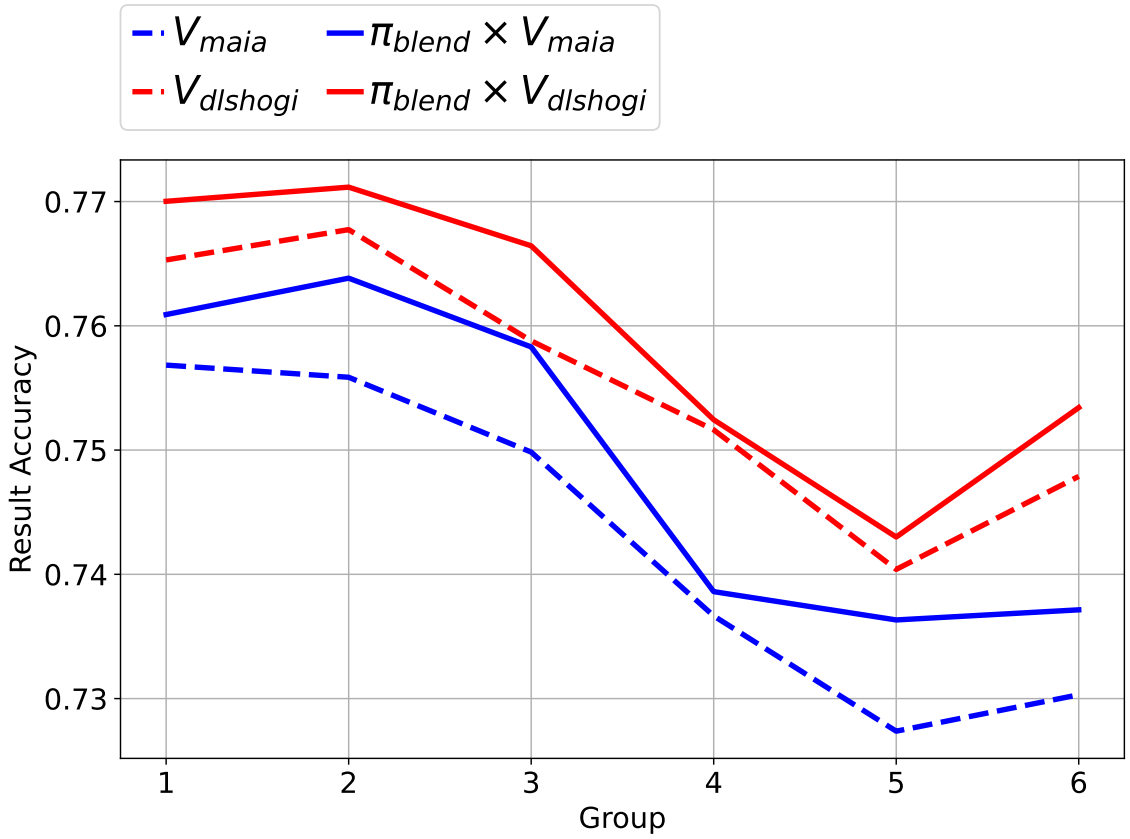


Figure 6.1: Result accuracy of the existing models and the inner product models using the Blend model for each rating group. Higher result accuracy indicates better win rate prediction performance.

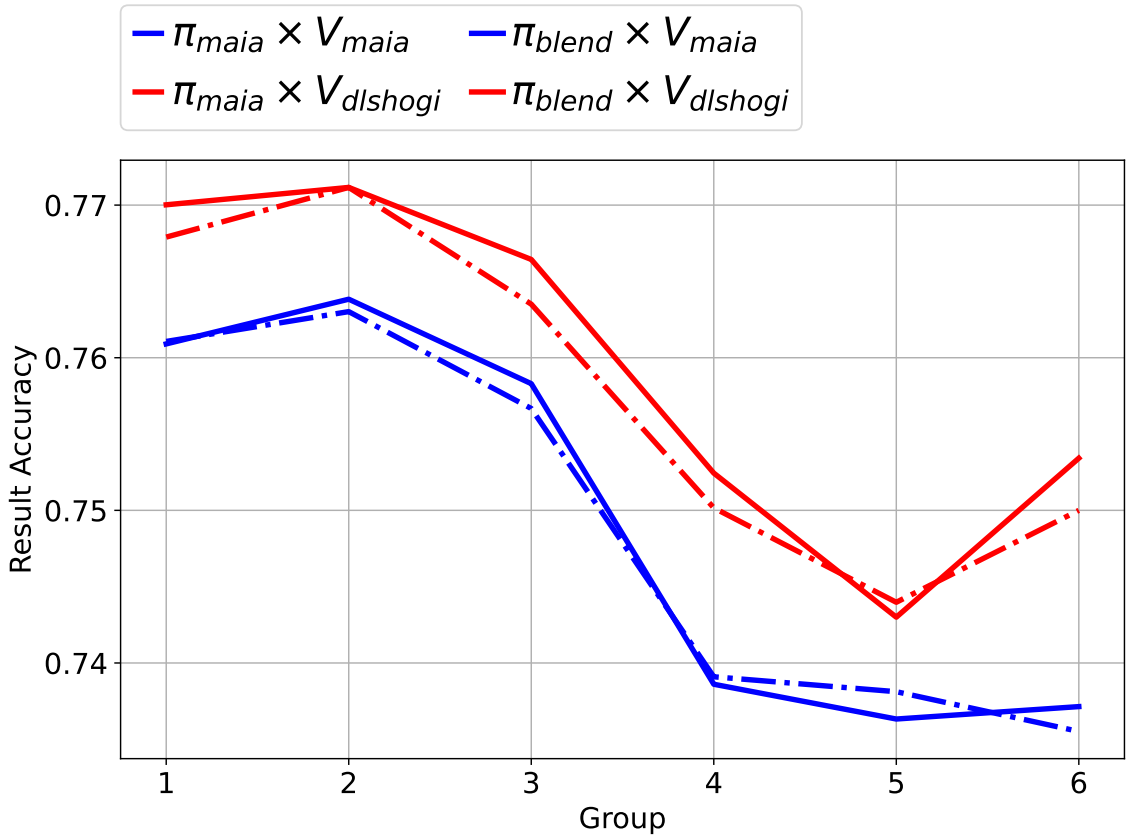


Figure 6.2: Result accuracy of the inner product models using Maia and those using the Blend model for each rating group.

Figure 6.2 shows the result accuracy of inner product models that use Maia-S-All as the move probability distribution and those that use the Blend model. The dash-dotted lines represent inner product models that use the Blend model as the move probability distribution. The tendencies that higher rating groups exhibit lower result accuracy, and that models using DLShogi win rate predictions outperform those using Maia-S-All win rate predictions, remain unchanged. Although the inner product model using the Blend model (solid lines) generally achieves higher result accuracy than the inner product model using Maia-S-All (dash-dotted lines), the difference is not statistically significant. The author guesses that this is because, while the Blend model has better move-prediction performance, there are relatively few moves for which the difference from Maia-S-All is large enough to change the game outcome.

### 6.4.3 Cross Entropy

Cross entropy is defined as the average of the following quantity with respect to the predicted win rate $\hat{V}$ of the side to move, and is often used as a loss function in binary classification:

$$\textbf{Cross Entropy} = \begin{cases} -\log \hat{V} & \textit{(side to move wins)} \\ -\log(1-\hat{V}) & \textit{(side to move loses).} \end{cases}$$

If the true win rate is $p$, cross entropy attains its minimum value only when $\hat{V} = p$. As discussed in Subsection 6.4.2, result accuracy counts a prediction as correct whenever the final winner is correctly predicted. Thus, even if the model predicts a win rate of 51% in a completely winning position, the prediction is still considered correct. In practice, however, it is more desirable that the predicted win rate be close to the actual human win rate. Cross entropy is an evaluation metric that rewards predicting 99% rather than 51% in such overwhelmingly winning positions.

As shown in Hiraoka's work [57], the win rates predicted by models obtained via reinforcement learning, such as AlphaZero, often take values close to 0 or 1, and DLShogi exhibits a similar tendency. In this study, aiming for a fairer comparison, we perform calibration for each model to either bring predicted win rates closer to 0.5 or closer to 0 or 1 (see the appendix for details). By the nature of the transformation, the result accuracy discussed in the previous subsection is unaffected.

Figure 6.3 shows the cross entropy of the existing models and the inner product models using the Blend model for each rating group. Note that, in contrast to result accuracy, lower cross entropy indicates better win rate prediction performance.

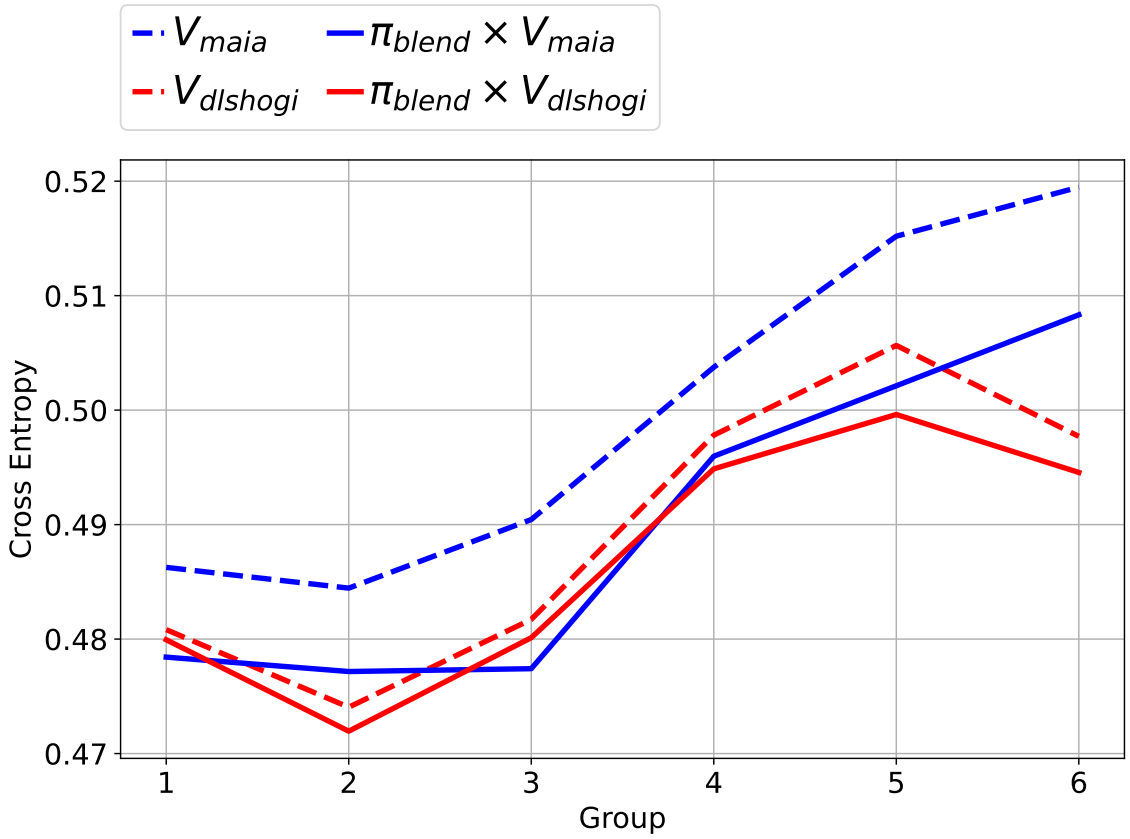


Figure 6.3: Cross entropy (after calibration) of the existing models and the inner product models using the Blend model for each rating group. Lower cross entropy indicates better win rate prediction performance.

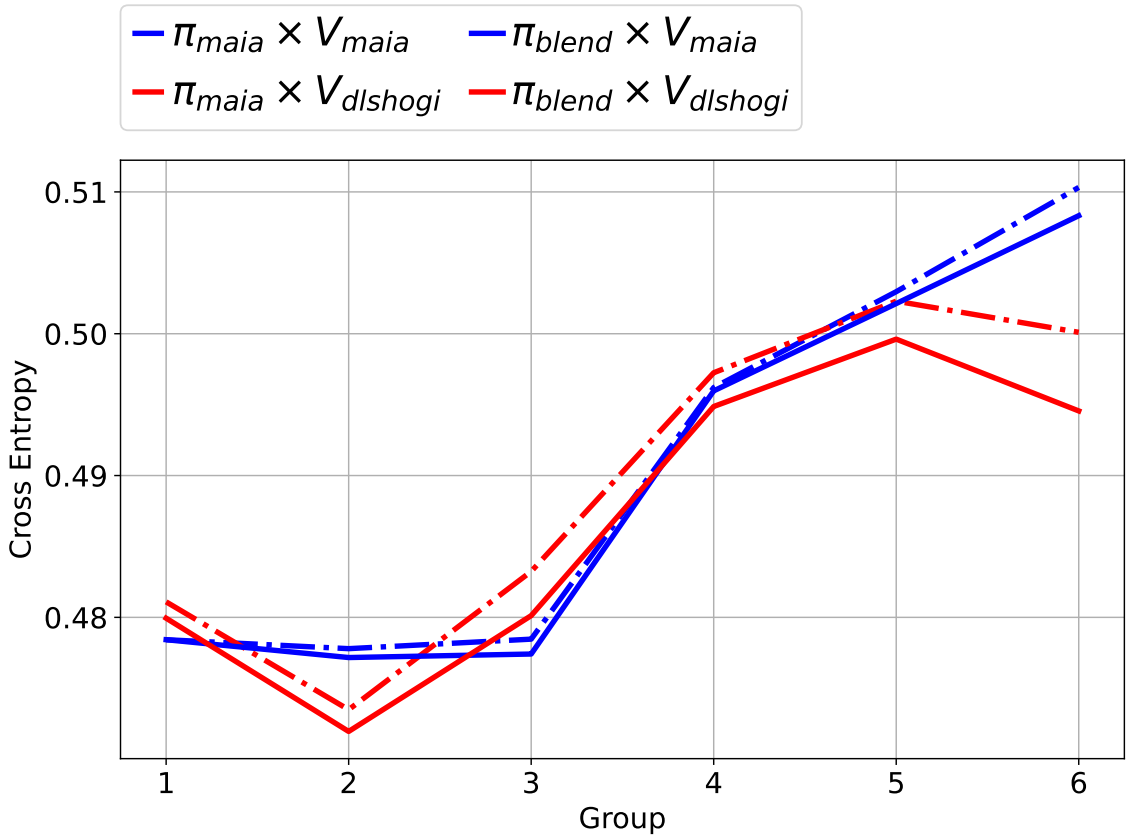


Figure 6.4: Cross entropy (after calibration) of the inner product models using Maia and those using the Blend model for each rating group.

Comparing the win rate predictions of the existing models (dashed lines) with those of the inner product models using the Blend model (solid lines), we find that for models using Maia-S-All win rate predictions (blue), the inner product models achieve lower cross entropy by about 0.7 to 1.3 points, and for models using DLShogi win rate predictions (red), the inner product models achieve lower cross entropy by about 0.1 to 0.6 points.

In contrast to the result accuracy, for inner product models using Maia-S-All win rate predictions (blue solid lines) and those using DLShogi win rate predictions (red solid lines), which model achieves lower cross entropy depends on the rating group. This indicates that, even when a model has inferior win rate prediction performance, applying the inner product model can sometimes reverse the relative performance between models.

Comparing the existing models, namely the win rate predictions of Maia-S-All (blue dashed line) and DLShogi (red dashed line), we observe that DLShogi achieves lower cross entropy by about 0.5 to 2.2 points. Overall, cross entropy tends to be higher for higher rating groups, which is consistent with the trend observed in result accuracy.

Figure 6.4 shows the cross entropy of inner product models that use Maia-S-All as the move probability distribution (dash-dotted lines) and those that use the Blend model (solid lines). The results largely mirror the tendencies seen in result accuracy: the inner product models using the Blend model perform slightly better overall.

### 6.4.4 Expected Calibration Error

Expected calibration error (ECE) is a widely used metric for evaluating calibration performance and is well aligned with the goal of this study, which is to adjust the raw win rate predictions of Maia-S-All and DLShogi so that they better reflect human win rates. To compute ECE, we divide the predicted win rates into bins, aggregate actual win rates within each bin, and measure the error between the predicted and empirical win rates. For example, if we collect 100 positions from game records for which the predicted win rate is approximately 80%, ECE measures the discrepancy between the predicted 80% and the actual fraction of wins in those positions. Figure 6.5 illustrates an example of ECE computation. In this study, we divide the range of predicted win rates into ten bins of width 10%.

Although smaller ECE values are more desirable, a model with ECE close to zero is not necessarily a good win rate prediction model. For example, a model that always outputs a predicted win rate of 0.5 for the side to move can achieve ECE close to zero, but such a model is clearly undesirable.

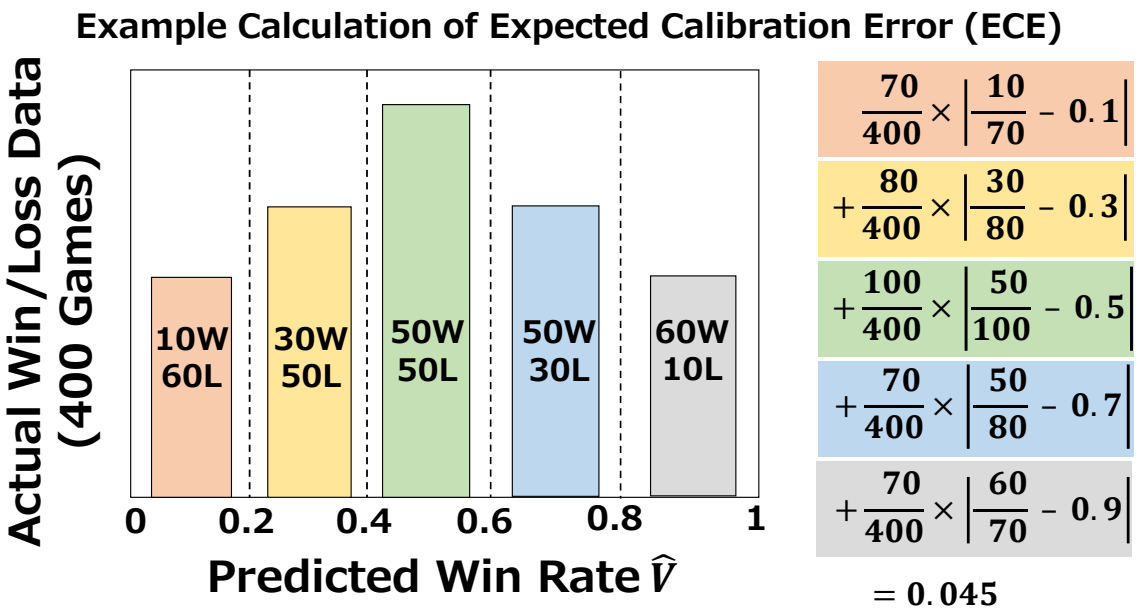


Figure 6.5: An example of expected calibration error calculation.

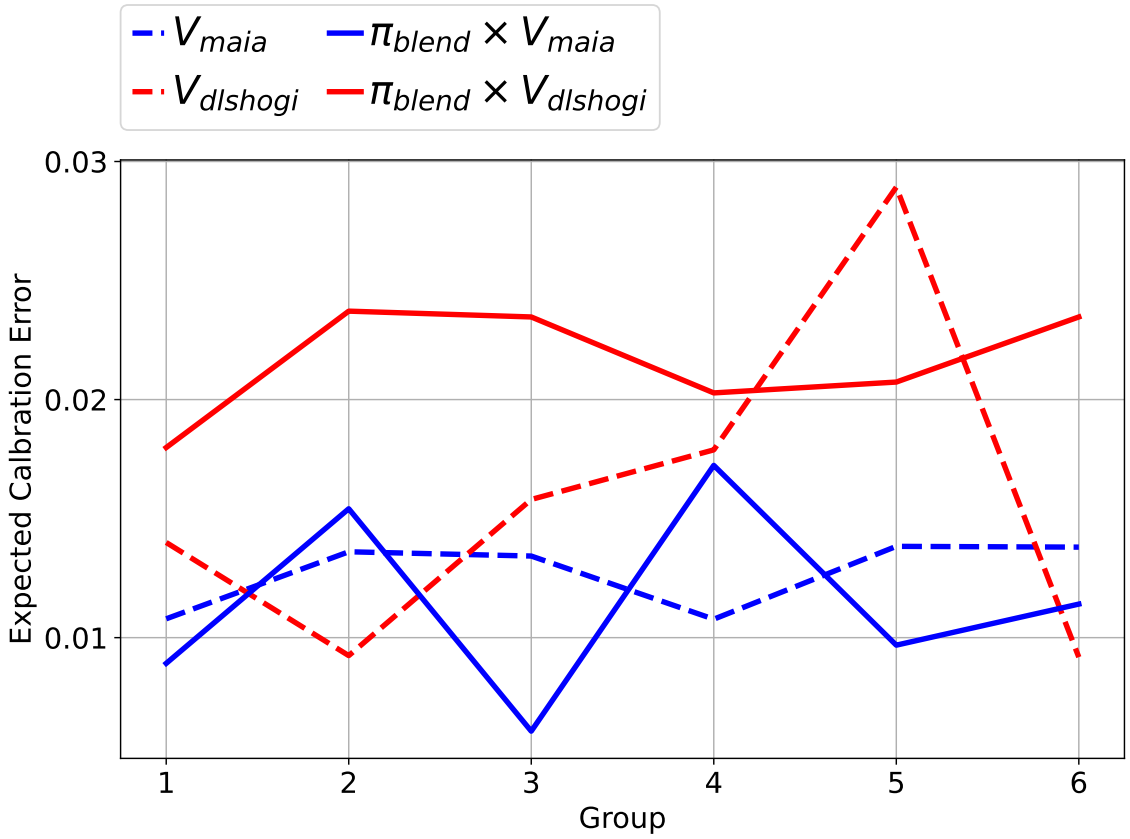


Figure 6.6: Expected calibration error (after calibration) of the existing and proposed models for each rating group. Lower expected calibration error indicates better win rate prediction performance.

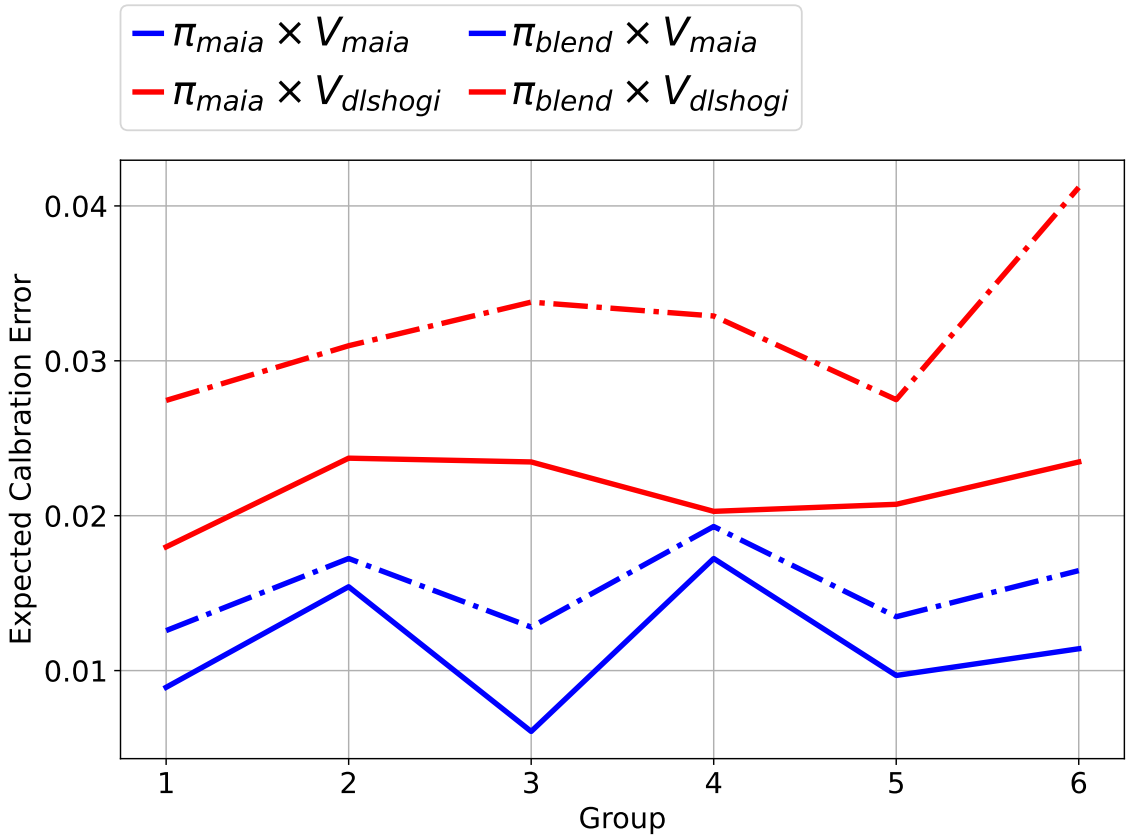


Figure 6.7: Expected calibration error (after calibration) of the existing and proposed models for each rating group.

Figure 6.6 shows the expected calibration error of the existing models and the inner product models using the Blend model for each rating group.

Comparing models that use DLShogi win rate predictions (red) with those that use Maia-S-All win rate predictions (blue), we see that models using DLShogi exhibit larger variance. This may be because the calibration used in these models is tuned for cross entropy rather than for ECE, and thus may not be optimal for reducing ECE.

Figure 6.7 shows the expected calibration error of inner product models that use Maia-S-All as the move probability distribution (dash-dotted lines) and those that use the Blend model (solid lines). In contrast to the previous evaluation metrics, it is clear from the figure that inner product models using the Blend model achieve substantially lower ECE than those using Maia-S-All.

To visualize the magnitude of ECE in this study, Figure 6.8 shows the relationship between predicted and actual win rates, together with the histogram of predicted win rates. The model is the inner product model using the Blend model, and the target rating group is Group 1. The black line indicates the line $y = x$, and ECE (about 0.01) corresponds to the error between the blue polyline and the black line, weighted by the frequencies in the histogram. As can be seen from the figure, the win rates predicted by the inner product model using the Blend model are very close to the actual win rates.

As mentioned earlier, even if the model does not understand the position well, it can make ECE close to zero simply by always predicting a win rate near 0.5. Thus, given the same ECE, a model that produces extreme values near 0 or 1 is considered to have a deeper understanding of the positions. Therefore, to evaluate the accuracy of win rate prediction, we desire an evaluation method that balances ECE with result accuracy and cross entropy.

## 6.5 Conclusion

In this study, to evaluate the accuracy of human win rate prediction, we compared existing models and the proposed models using three metrics: result accuracy, cross entropy, and expected calibration error. As existing models, we used the win rate predictions of DLShogi, trained via reinforcement learning in a manner similar to AlphaZero, and those of Maia-S-All, trained via supervised learning. As proposed models, we introduced inner product models that combine these win rate predictions with move probability distributions.

We propose an inner product model that considers only one move ahead from the root position. However, we believe that methods that perform deeper search based

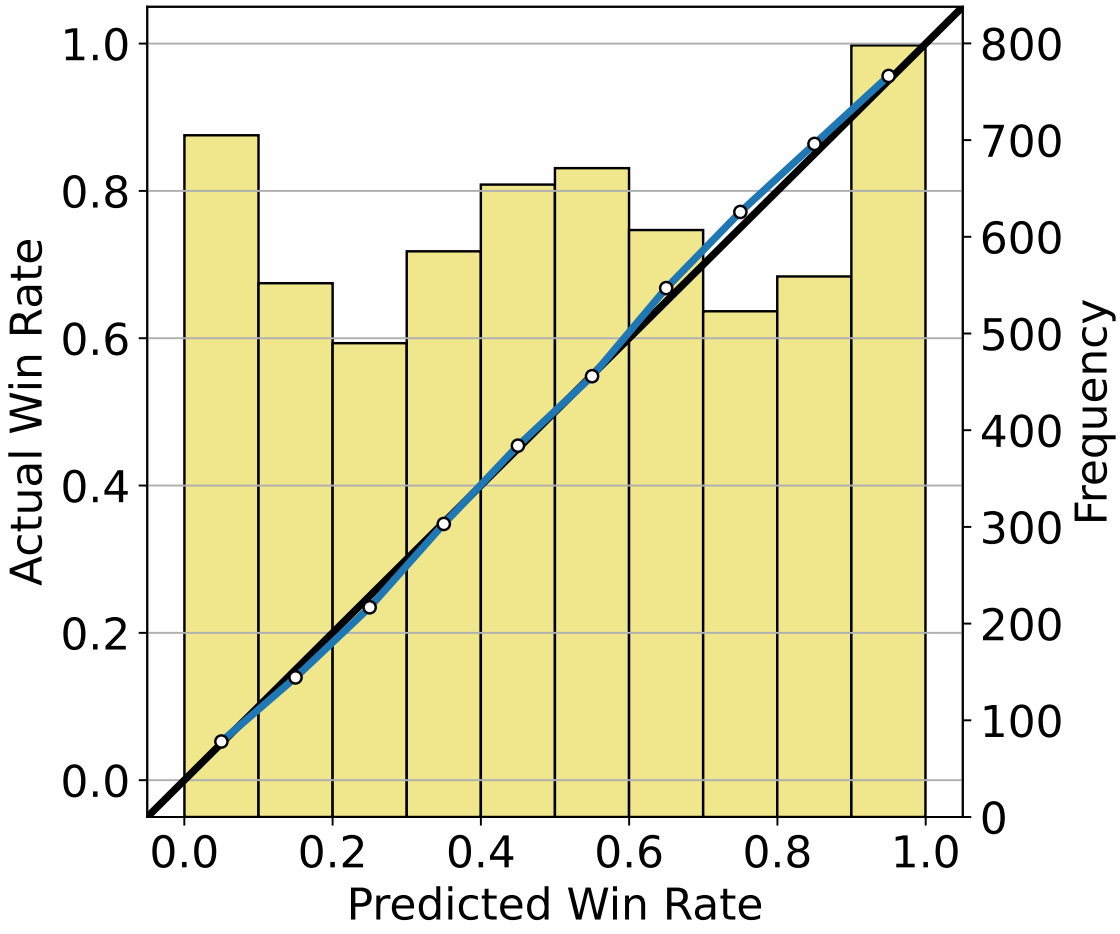


Figure 6.8: Relationship between predicted and actual win rates, and histogram of predicted win rates. The model is the inner product model using the Blend model, and the target rating group is Group 1.

on move probability distributions, such as realization probability search [48], could also be effective. In addition, it may be beneficial to consider methods that mix the win rate predictions of Maia-S-All and DLShogi.

# Chapter 7

# Conclusion

## 7.1 Summary of Findings

This study explored methods for learning human-like policies from limited data, along with their evaluation metrics and applications. The main contributions of this research are summarized as follows:

- Human-like policies at the group level: In the previous study, Maia trained each rating group using 12 million records via supervised learning. We found that when training with only about 1% of that data, the model tended to make poor moves in complex positions that require search. To address this, we proposed a Blend Model that combines the policy of a supervised model — human-like but weak in difficult positions — with that of a reinforcement learning model — strong in difficult positions but less human-like. The Blend Model improved move-matching accuracy in Chess, Go, and Shogi, and was shown to be robust with respect to the blending parameter $\alpha$. In Chess, even when compared to Maia trained on 100 times more data, the Blend Model achieved higher move-matching accuracy. In Go, we demonstrated that the proposed method outperformed KL-Regularized Search. In Shogi, we analyzed why the Blend Model was effective and showed that it was particularly beneficial in positions where the supervised model tended to make poor moves.

- Human-like policies at the individual level: The previous study, Transfer Maia, reported that fine-tuning with 1,000 game records per player degraded move-matching accuracy, suggesting more game records per player were required. However, in most games other than Chess, few players have more than 1,000 records, making it important to develop methods that can learn effectively from

smaller datasets. We proposed a two-stage fine-tuning approach that leverages not only the target player's records but also similar game records. To measure similarity, we introduced two approaches: one based on embedding vectors learned from player identification models, and another based on game-specific feature vectors. Experiments showed that two-stage fine-tuning — first using data similar to the target player, then fine-tuning on the target player's data — led to moderate improvements in move-matching accuracy.

- **Applications of human-like policies:** We explored applications of human-like policies and proposed an inner product-based model that improves human player's win rate prediction accuracy by incorporating human-like policies. We also introduced three evaluation metrics for human win rate prediction: result accuracy, cross-entropy loss, and expected calibration error. Result matching accuracy measures whether the predicted win rate correctly reflects the actual game outcome relative to the 50% threshold. It is particularly important because it aligns well with the intuitive understanding of human spectators. Cross-entropy loss penalizes incorrect predictions more severely when the confidence is high and is minimized only when predicted and actual win rates match exactly. Expected calibration error measures how well predicted win rates align with empirical outcomes (e.g., whether positions predicted to have an 80% win rate actually lead to wins about 80% of the time). Using these three evaluation metrics, we demonstrated that incorporating human-like policies improves win rate prediction performance.

## 7.2 Future Directions

Possible future directions include applying the proposed methods to games with different properties. The games considered in this study — Shogi, Go, and Chess — are two-player perfect-information games with discrete state and action spaces. An interesting extension would be to apply these methods to games or domains with continuous states and actions, or to tasks with even more limited data. Beyond games, similar approaches could also be applied to robotics, where reinforcement learning and imitation learning are widely used.

Another potential direction concerns moving beyond imitation of existing players. While this study focused on mimicking real players at the group and individual levels, future work could explore creative synthesis – for example, generating new play styles by blending data from players with different characteristics, or specifying desired traits to create a model that embodies them.

# Appendix A

# Appendix: Human-Like Policies at Group Level

## A.1 Experiment of Multiple Trials on Chess

In this section, we provide the results of the move-matching accuracy obtained from multiple supervised learning trials and the Blend method using each supervised learning model. In more detail, we separated the collected game records into training, validation, and test data, where the ratios were 90%, 5%, and 5%, respectively, as described in subsection 4.4.1. In this experiment, we prepared three different separations for each set of the collected game records (Maia-S-1100, Maia-S-1900, and Maia-S-Wide) to investigate the robustness of the supervised learning models and the Blend method.

Figure A.1 shows the move-matching accuracy of the supervised learning models and the Blend models tested on rating 1100 data for each trial. The x-axis represents trials and the y-axis represents the move-matching accuracy. The error bars represent the 95% confidence intervals calculated from the test data for each trial. The types of lines represent the types of the models. Specifically, the chain lines represent supervised learning policies such as Maia-S and Maia-S-Wide and the solid lines represent their Blend models. The colors of the bars represent the rating of the human games used in the training data for the corresponding policy. Specifically, yellow represents using the training data of rating 1100, and red represents using the training data of rating 1100 to 1900. The results showed that different trials obtained similar move-matching accuracy, though the deviations of the Maia-S-1100 models were relatively big. Nevertheless, the Blend model in each trial outperformed the corresponding Maia-S model.

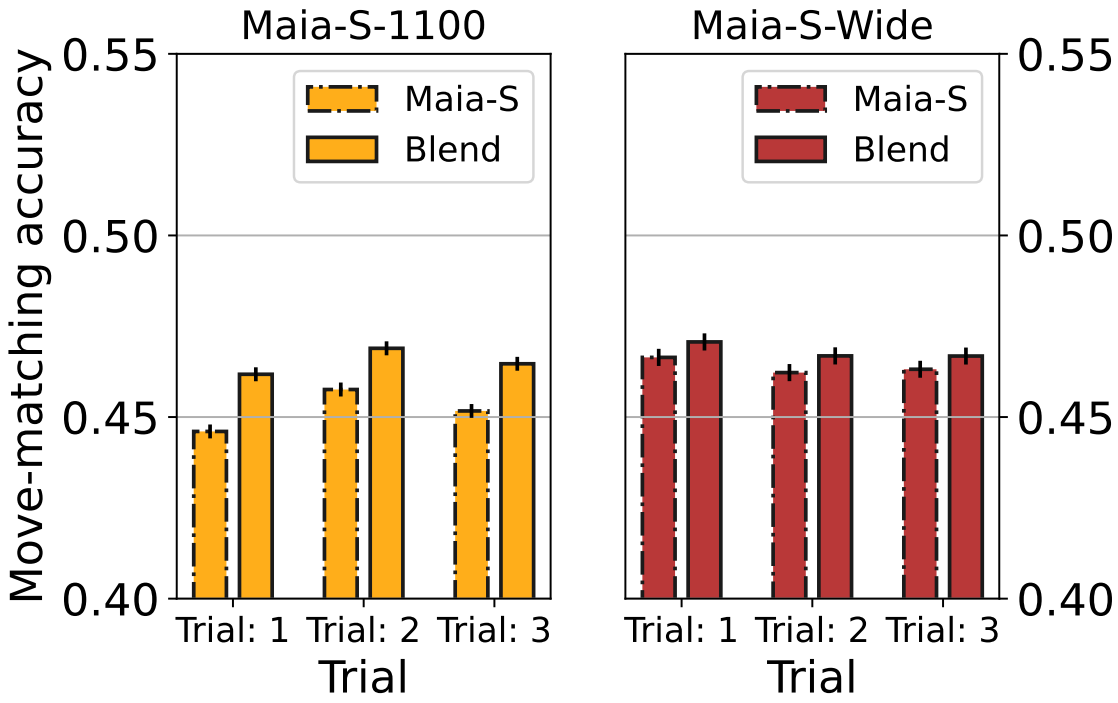


Figure A.1: Move-matching accuracy of Maia-S-1100, Maia-S-Wide, and their Blend models on rating 1100 data for each trial.

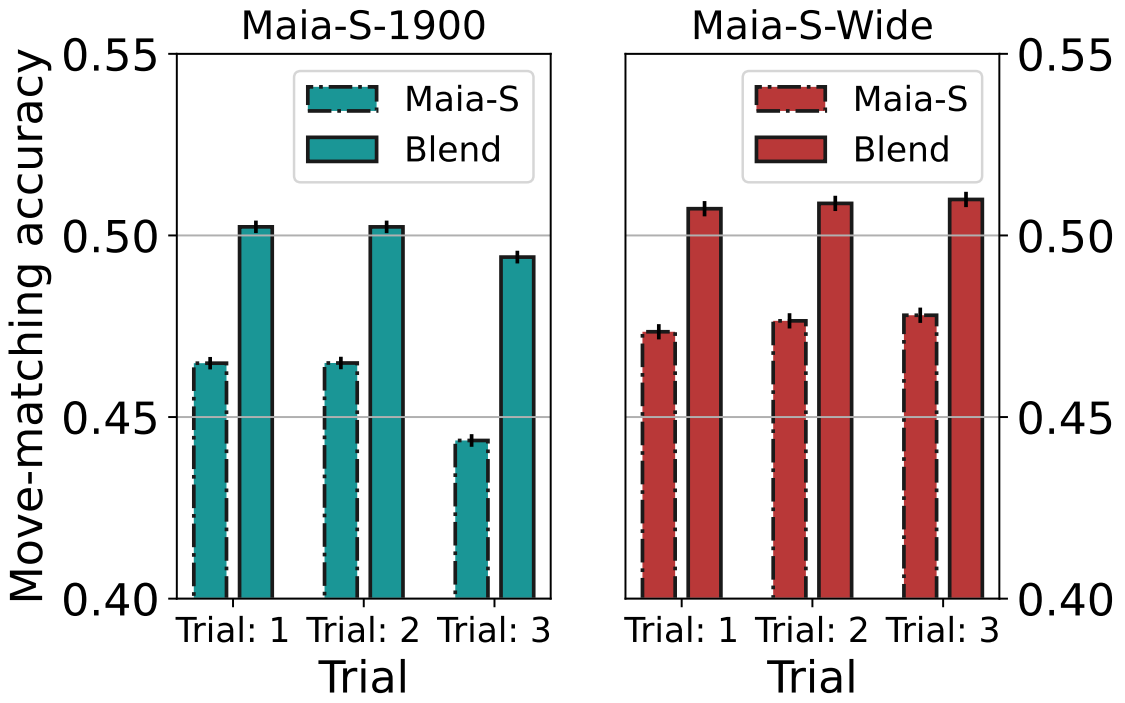


Figure A.2: Move-matching accuracy of Maia-S-1900, Maia-S-Wide, and their Blend models on rating 1900 data for each trial.

We also conducted experiments on rating 1900 data, and the results are shown in Figure A.2. The only difference in representation between Figures A.1 and A.2 is that yellow bars becomes blue bars, which means using the training data of rating 1100 and 1900, respectively. Similar to Figure A.1, the move-matching accuracy of different trials was similar, except that the deviations of the Maia-S-1900 models were relatively big. Also, the Blend model in each trial was clearly better than the corresponding Maia-S model. From these results, we concluded the results of one trial to be reliable enough.

## A.2 The Definition of Loss

In subsection 4.6.3, we made an assumption that the Blend model can eliminate supervised learning models' bad moves that require a bit of searching (thinking) because AlphaZero-like policies are incorporated. To verify our assumption, we define the loss of a move to evaluate how bad the move is.

For a given position, let $move_{Human}$ be the human move, $move_{Maia}$ be the move that Maia-S-Wide assigns the highest probability to, $move_{DLShogi}$ be the move that DLShogi assigns the highest probability to, and $move_{Blend}$ be the move that the Blend model assigns the highest probability to. Also, let the evaluation of each move (cp: centipawn) be $CP_{Human}, CP_{Maia}$, $CP_{DLShogi}$, $CP_{Blend}$. We then define the best evaluation at the position $CP_{Best}$ and move $m$'s loss $Loss_m$ as follows.

$$CP_{Best} = max(\{CP_{Human}, CP_{Maia}, CP_{DLShogi}, CP_{Blend}\})$$

$$Loss_m = CP_{Best} - CP_m$$

In other words, the larger the loss is, the worse the move is, and the best move' loss is 0. The main reason for focusing on these four kinds of moves instead of all legal moves is to reduce the computation time. In Shogi, the average number of legal moves is 92 [21], and evaluating all legal moves is time-consuming. In addition, we consider that the sets of these four kinds of moves are likely to contain promising (the best) moves.

The remaining task to calculate the loss of a move is to obtain the cp evaluations, which we need a strong Shogi engine. DLShogi is strong but is used in our proposed method, so it would be unfair to evaluate the proposed method on that basis. YaneuraOu + Suisho [52] is a Shogi engine that has strength similar to DLShogi. This Shogi engine uses YaneuraOu's $\alpha\beta$ search and an evaluation function called Suisho. To obtain the cp of each move, we employed YaneuraOu + Suisho with 100,000 search nodes and default values for the rest of the settings.

# Appendix B

# Appendix: Human-Like Policies At Individual Level

Tables B.1, B.2, and B.3 present the configurations of the models and networks.

## B.1 Configurations of Models and Networks

Table B.1: Base Model Configuration

| Model Type | Residual CNN |
|---|---|
| Blocks | 6 |
| Channels | 192 |
| Batch Size | 1024 |
| Epoch | 30 |
| Learning Rate | 0.01 |
| Optimizer | SGD with Momentum |
| Momentum | 0.9 |
| Weight Decay | 0.0001 |
| Loss Function | Cross-Entropy (Policy) + BCEWithLogits (Value) |

Table B.2: Embedding Network Configuration

| Model Type | Residual CNN + Transformer |
|---|---|
| Blocks of Residual CNN | 6 |
| Channels of Residual CNN | 192 |
| Epoch | 6 |
| Learning Rate | 0.01 |
| Optimizer | SGD with momentum |
| Momentum | 0.9 |
| Loss Function | Generalized End-to-End Loss (G2E2) |
| Input Dimension to Transformer | 160 |
| Embedding Size | 512 |
| Players per Batch | 15 |
| Games per Player | 10 |

Table B.3: SGFT Model Configuration

| Model Type | Residual CNN |
|---|---|
| Blocks | 6 |
| Channels | 192 |
| Batch Size | 256 |
| Epoch (Stage 1) | 2 |
| Epoch (Stage 2) | 10 |
| Learning Rate | 0.00025 |
| Optimizer | SGD with momentum |
| Momentum | 0.9 |
| Weight Decay | 0.0001 |
| Loss Function | Cross-Entropy (Policy) + BCEWithLogits (Value) |

# Appendix C

# Appendix: Applications of Human-Like Policies

## C.1 Calibration of DLShogi's Predicted Win Rates

| Group | $V_{maia}$ | $V_{dlshogi}$ | $\pi_{maia} \times V_{maia}$ | $\pi_{maia} \times V_{dlshogi}$ | $\pi_{blend} \times V_{maia}$ | $\pi_{blend} \times V_{dlshogi}$ |
|---|---|---|---|---|---|---|
| Group 1 | 1.03 | 2.61 | 0.98 | 2.43 | 0.98 | 2.46 |
| Group 2 | 0.98 | 2.45 | 0.93 | 2.30 | 0.93 | 2.31 |
| Group 3 | 0.87 | 2.21 | 0.85 | 2.11 | 0.85 | 2.12 |
| Group 4 | 0.89 | 2.24 | 0.85 | 2.10 | 0.85 | 2.13 |
| Group 5 | 0.88 | 2.23 | 0.88 | 2.08 | 0.88 | 2.11 |
| Group 6 | 0.80 | 2.04 | 0.78 | 1.89 | 0.78 | 1.93 |

Table C.1: Adjusted temperature parameters for each group and model.

As shown in the study by Hiraoka [57], models obtained through reinforcement learning such as AlphaZero tend to produce extreme win rate predictions close to 0 or 1. Figure C.2 shows a histogram of DLShogi's predicted win rates for the validation data in Group 1. It can be seen that for many positions, the predicted win rate is either between 0.0–0.1 or 0.9–1.0.

To address this issue, we perform win-rate calibration using the following formula, which was also used in Hiraoka's study:

$$V_{cal} = \frac{V^{1/T}}{V^{1/T} + (1-V)^{1/T}}. \tag{C.1}$$

Figure C.1 shows an example of win rate calibration with different temperature values using Eq. (C.1). Equation (C.1) is a monotonically increasing function that passes through the points (0, 0), (0.5, 0.5), and (1, 1). When the temperature parameter $T \in [0, \infty)$ decreases, most values are transformed closer to 0 or 1, while larger $T$ values push most predictions toward 0.5.

In this study, we determine the temperature parameter $T$ for calibration using validation data. However, adjusting the parameter to minimize expected calibration error tends to produce excessively large $T$ values, resulting in nearly all predictions being close to 0.5. Since result accuracy is unaffected by calibration, we use the temperature parameter $T$ optimized for cross-entropy across all models. Specifically, we perform an exhaustive search in the range of 0.5 to 3.0 at intervals of 0.01 and adopt the parameter that yields the best performance on the validation data.

Figure C.3 shows the cross entropy of DLShogi for the validation data in Group 1, after tuning the temperature parameter $T$. When the temperature parameter is 1 (i.e., using DLShogi's raw predictions), the cross entropy is around 0.7. In contrast, with the optimal temperature parameter $T = 2.61$, the cross entropy decreases to around 0.5. Similar convex (U-shaped) relationships were observed for other models and groups, with the minimum typically located between $T = 0.5$ and $T = 3.0$. Table C.1 summarizes the final adjusted temperature parameters for each group and model.

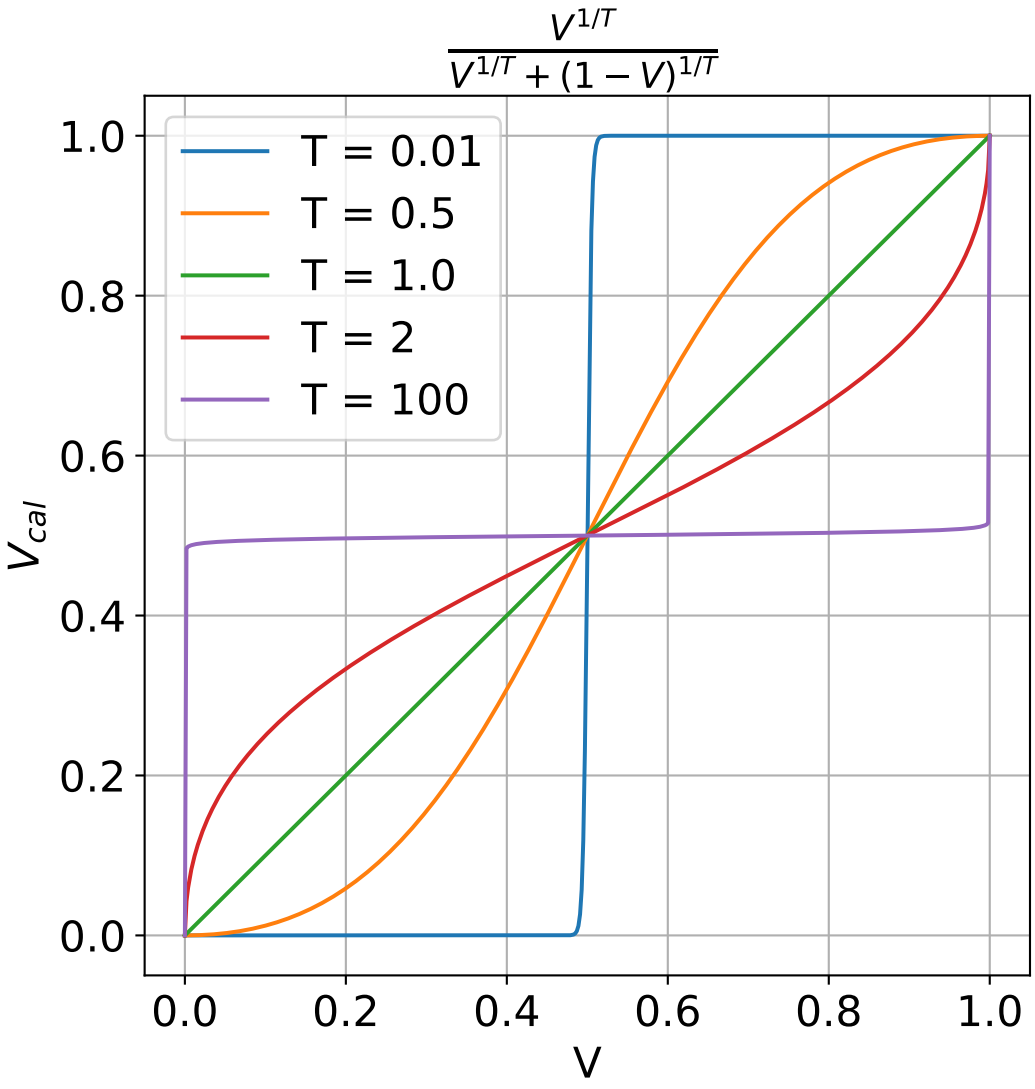


Figure C.1: Example of win-rate calibration with different temperature values using Eq. (C.1).

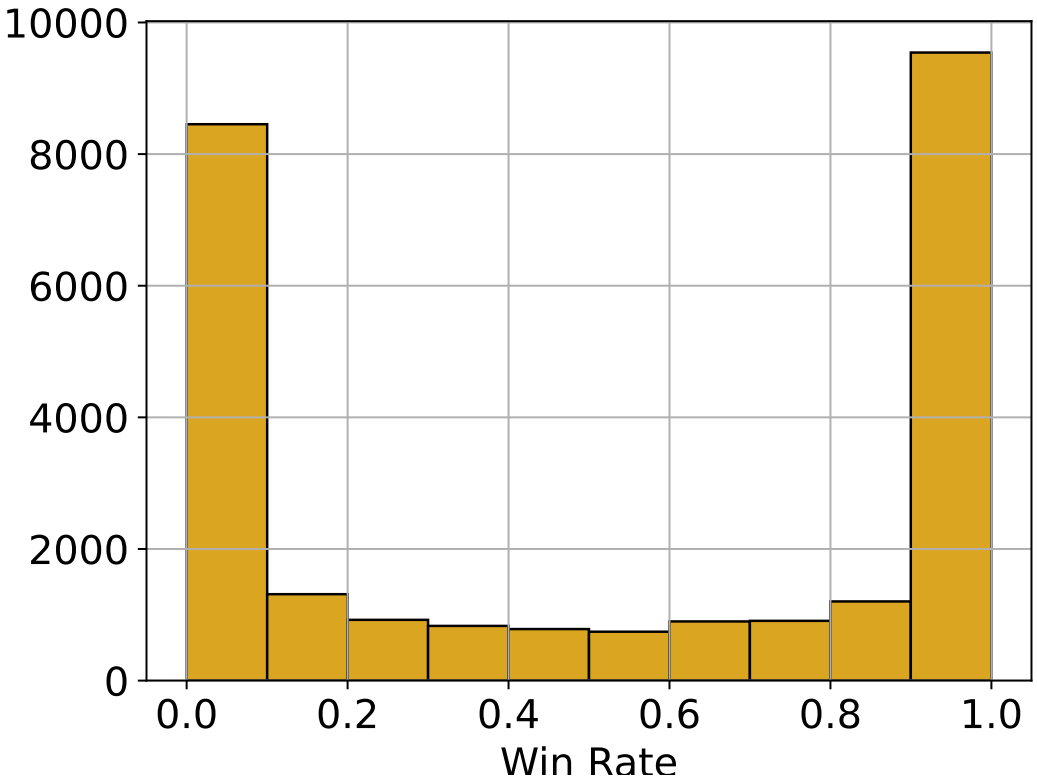


Figure C.2: Histogram of DLShogi's predicted win rates (Group 1).

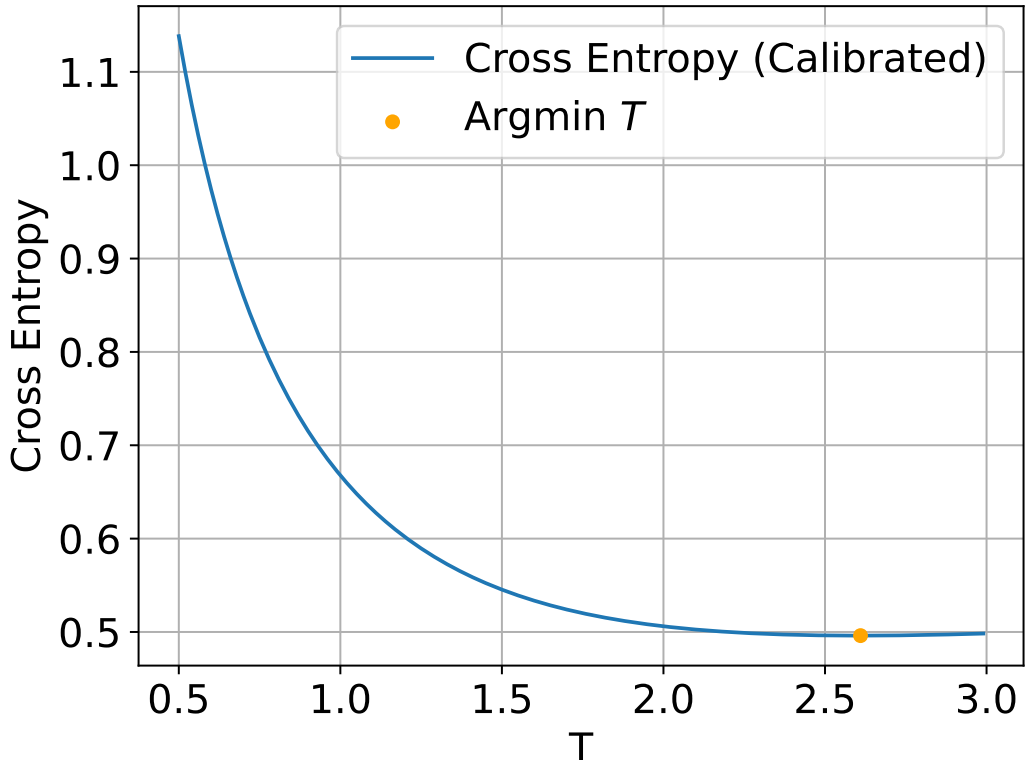


Figure C.3: Example of cross-entropy calibration for DLShogi (Group 1).

# Acknowledgements

I would like to express my sincere and deepest gratitude to Prof. Kokolo Ikeda and Prof. Chu-Hsuan Hsueh for their invaluable guidance throughout the completion of this dissertation. I am also deeply grateful to all members of the Ikeda Laboratory for their support in both my research and daily life. I would like to thank my friends for their continuous friendship and encouragement. Finally, I would like to express my heartfelt gratitude to my family for their unwavering support and for raising me to who I am today.

# List of Publications

## International Journals

1. Chester, S. C., Rogers, J. C., **Ogawa, T.**, Terao, M., Nakai, R., Abe, N., & De Brito, S. A. (2025). White Matter Correlates of Psychopathic Traits in a Japanese Community Sample: Sex Matters!. European Journal of Neuroscience, 62(1), e70177.

2. **Ogawa, T.**, Hsueh, C. H., & Ikeda, K. (2024). More Human-Like Gameplay by Blending Policies From Supervised and Reinforcement Learning. IEEE Transactions on Games, 16(4), 831-843.

3. Chester, S. C., **Ogawa, T.**, Terao, M., Nakai, R., Abe, N., & De Brito, S. A. (2023). Cortical and subcortical grey matter correlates of psychopathic traits in a Japanese community sample of young adults: sex and configurations of factors' level matter!. Cerebral Cortex, 33(9), 5043-5054.

## International Conferences

1. Kuboki, K., **Ogawa, T**., Hsueh, C. H., Yen, S. J., & Ikeda, K. (2025). Policies of Multiple Skill Levels for Better Strength Estimation in Games. In 2025 21st AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2025).

2. **Ogawa, T.**, Nakagawa, K., & Ikeda, K. (2024). Optimal execution strategy using Deep Q-Network with heuristics policy. In 2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI) .

3. **Ogawa, T.**, Hsueh, C. H., & Ikeda, K. (2023). Improving the human-likeness of game AI's moves by combining multiple prediction models. In 2023 15th International Conference on Agents and Artificial Intelligence (ICAART 2023).

# Domestic Journals

1. **小川 竜欣**, 河野 慎, 謝 志杰, 岸 尚希, ... & 落合 桂一. (2025). 手書き漢字の埋め込み表現による反復学習頻度最適化のための半減期予測. 情報処理学会論文誌, 66(9), 1346-1356.

# Domestic Conferences

1. **小川竜欣**, & 池田心. (2024). 着手確率を用いた人間の勝率予測の改善. 第 29 回ゲームプログラミングワークショップ（GPW-24）.

2. **小川竜欣**, 中川慧, & 池田心. (2024). ヒューリスティック方策を組み合わせた Deep Q-Network による最適執行戦略. 人工知能学会全国大会論文集 第 38 回. 一般社団法人 人工知能学会.

3. **小川竜欣**, & 池田心. (2022). 着手予測モデルが予測しづらい局面の考察・分類と確信度を利用した一致率の向上. 第 28 回ゲームプログラミングワークショップ（GPW-22）.

4. **小川竜欣**, & 池田心. (2021). 対局状況をより正確に表現するための盤面評価値. 第 27 回ゲームプログラミングワークショップ（GPW-21）.

# Bibliography


[1] Nolan Bard et al. "The Hanabi challenge: A new frontier for AI research". In: *Artificial Intelligence* 280 (2020), p. 103216. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2019.103216. URL: https://www.sciencedirect.com/science/article/pii/S0004370219300116.

[2] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[3] Murray Campbell. "Knowledge discovery in deep blue". In: *Commun. ACM* 42.11 (Nov. 1999), pp. 65–67. ISSN: 0001-0782. DOI: 10.1145/319382.319396. URL: https://doi.org/10.1145/319382.319396.

[4] Micah Carroll et al. *On the Utility of Learning about Humans for Human-AI Coordination*. 2020. arXiv: 1910.05789 [cs.LG]. URL: https://arxiv.org/abs/1910.05789.

[5] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[6] Rémi Coulom. "Computing "Elo ratings" of move patterns in the game of Go". In: *ICGA journal* 30.4 (2007), pp. 198–208.

[7] Maximilian Croissant et al. "An appraisal-based chain-of-emotion architecture for affective language model game agents". In: *Plos one* 19.5 (2024), e0301033.

[8] Ekin D Cubuk et al. "Autoaugment: Learning augmentation strategies from data". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 113–123.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[10] Nobuto Fujii et al. "Evaluating human-like behaviors of video-game agents autonomously acquired with biological constraints". In: *International Conference on Advances in Computer Entertainment Technology*. 2013, pp. 61–76.

[11] Tobias Gessler et al. *OvercookedV2: Rethinking Overcooked for Zero-Shot Coordination*. 2025. arXiv: 2503.17821 [cs.AI]. URL: https://arxiv.org/abs/2503.17821.

[12] Camilo Gordillo et al. "Improving playtesting coverage via curiosity driven reinforcement learning agents". In: *2021 IEEE Conference on Games (CoG)*. IEEE. 2021, pp. 1–8.

[13] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[14] Philip Hingston. "A new design for a Turing Test for Bots". In: *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. 2010, pp. 345–350. DOI: 10.1109/ITW.2010.5593336.

[15] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in neural information processing systems* 29 (2016), pp. 4572–4580.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[17] Kunihito Hoki and Tomoyuki Kaneko. "Large-scale optimization for evaluation functions with minimax search". In: *Journal of Artificial Intelligence Research* 49 (2014), pp. 527–568.

[18] Chu-Hsuan Hsueh and Kokolo Ikeda. "Improvement of move naturalness for playing good-quality games with middle-level players". In: *Applied Intelligence* 54.2 (2024), pp. 1637–1655. DOI: doi.org/10.1007/s10489-023-05210-2.

[19] Chu-Hsuan Hsueh and Kokolo Ikeda. "Playing good-quality games with weak players by combining programs with different roles". In: *2022 IEEE Conference on Games (CoG)*. IEEE. 2022, pp. 612–615.

[20] HumanCompatibleAI. *Overcooked-AI: A benchmark environment for cooperative human-AI task performance*. https://github.com/HumanCompatibleAI/overcooked_ai. 2019.

[21] Hiroyuki Iida, Makoto Sakuta, and Jeff Rollason. "Computer shogi". In: *Artificial Intelligence* 134.1 (2002), pp. 121–144. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(01)00157-6. URL: https://www.sciencedirect.com/science/article/pii/S0004370201001576.

[22] Kokolo Ikeda, Simon Viennot, and Naoyuki Sato. "Detection and labeling of bad moves for coaching Go". In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. 2016, pp. 1–8. DOI: 10.1109/CIG.2016.7860441.

[23] Athul Paul Jacob et al. "Modeling strong and human-like gameplay with KL-regularized search". In: *International Conference on Machine Learning*. 2022, pp. 9695–9728.

[24] Jaechang Kim et al. *Bridging the Gap between Expert and Language Models: Concept-guided Chess Commentary Generation and Evaluation*. 2025. arXiv: 2410.20811 [cs.LG]. URL: https://arxiv.org/abs/2410.20811.

[25] Tetsuhiko Kinebuchi and Takeshi Ito. "Shogi Program That Selects Natural Moves by Considering the Flow of Preceding Moves". In: *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*. IEEE. 2015, pp. 79–84.

[26] Joel Lehman and Risto Miikkulainen. "Neuroevolution". In: *Scholarpedia* 8.6 (2013), p. 30977.

[27] Reid McIlroy-Young et al. "Learning models of individual behavior in chess". In: *SIGKDD*. 2022, pp. 1253–1263.

[28] Reid McIlroy-Young et al. "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*. 2020, pp. 1677–1687.

[29] Reid McIlroy-Young et al. "Detecting individual decision-making style: Exploring behavioral stylometry in chess". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24482–24497.

[30] Tatsuyoshi Ogawa et al. "More Human-Like Gameplay by Blending Policies from Supervised and Reinforcement Learning". In: *IEEE ToG* 16.4 (2024), pp. 831–843.

[31] Tatsuyoshi Ogawa., Chu-Hsuan Hsueh., and Kokolo Ikeda. "Improving the Human-Likeness of Game AI's Moves by Combining Multiple Prediction Models". In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*. INSTICC. SciTePress, 2023, pp. 931–939. ISBN: 978-989-758-623-1. DOI: 10.5220/0011804200003393.

[32] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[33] Joon Sung Park et al. "Generative agents: Interactive simulacra of human behavior". In: *Proceedings of the 36th annual acm symposium on user interface software and technology*. 2023, pp. 1–22.

[34] Luong Huu Phuc, Kanazawa Naoto, and Ikeda Kokolo. "Learning human-like behaviors using neuroevolution with statistical penalties". In: *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. 2017, pp. 207–214. DOI: 10.1109/CIG.2017.8080437.

[35] Mihai Polceanu. "MirrorBot: Using human-inspired mirroring behavior to pass a turing test". In: *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. 2013, pp. 1–8. DOI: 10.1109/CIG.2013.6633618.

[36] Nikaash Puri et al. *Explain Your Move: Understanding Agent Actions Using Specific and Relevant Feature Attribution*. 2020. arXiv: 1912.12191 [cs.CV]. URL: https://arxiv.org/abs/1912.12191.

[37] Constantin Ruhdorfer et al. "The Overcooked Generalisation Challenge: Evaluating Cooperation with Novel Partners in Unknown Environments Using Unsupervised Environment Design". In: *Transactions on Machine Learning Research* (2025). URL: https://www.collaborative-ai.org/publications/ruhdorfer25_tmlr.pdf.

[38] Jacob Schrum, Igor V. Karpov, and Risto Miikkulainen. "UT^2: Human-like behavior via neuroevolution of combat behavior and replay of human traces". In: *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. 2011, pp. 329–336. DOI: 10.1109/CIG.2011.6032024.

[39] Lisa Schut et al. *Bridging the Human-AI Knowledge Gap: Concept Discovery and Transfer in AlphaZero*. 2023. arXiv: 2310.16410 [cs.AI]. URL: https://arxiv.org/abs/2310.16410.

[40] Claude E. Shannon. "XXII. Programming a computer for playing chess". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41.314 (1950), pp. 256–275. DOI: 10.1080/14786445008521796. eprint: https://doi.org/10.1080/14786445008521796. URL: https://doi.org/10.1080/14786445008521796.

[41] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144. DOI: 10.1126/science.aar6404. eprint: https://www.science.org/doi/pdf/10.1126/science.aar6404. URL: https://www.science.org/doi/abs/10.1126/science.aar6404.

[42] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[43] David Silver et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[44] Ho Chit Siu et al. "Evaluation of Human-AI Teams for Learned and Rule-Based Agents in Hanabi". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 16183–16195. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[45] DJ Strouse et al. *Collaborating with Humans without Human Data*. 2022. arXiv: 2110.08176 [cs.LG]. URL: https://arxiv.org/abs/2110.08176.

[46] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[47] Julian Togelius et al. "Assessing believability". In: *Believable bots*. Springer, 2013, pp. 215–230.

[48] Yoshimasa Tsuruoka, Daisaku Yokoyama, and Takashi Chikayama. "Game-tree search algorithm based on realization probability". In: *Icga Journal* 25.3 (2002), pp. 145–152.

[49] Alan Mathison Turing. "Computing Machinery and Intelligence". In: *Mind* 49 (1950), pp. 433–460.

[50] Erik C.D. van der Werf and Mark H.M. Winands. "Solving Go for rectangular boards". In: *ICGA Journal* 32.2 (2009), pp. 77–88. DOI: 10.3233/ICG-2009-32203.

[51] Chang Xiao and Brenda Z. Yang. *LLMs May Not Be Human-Level Players, But They Can Be Testers: Measuring Game Difficulty with LLM Agents*. 2024. arXiv: 2410.02829 [cs.AI]. URL: https://arxiv.org/abs/2410.02829.

[52] *YaneuraOu + Suisho5 Shogi engine*. https://github.com/mizar/YaneuraOu/releases/tag/v7.5.0. (Accessed on 03/15/2023).

[53] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*. Vol. 2. Springer, 2018.

[54] Chao Yu et al. *Learning Zero-Shot Cooperation with Humans, Assuming Humans Are Biased*. 2023. arXiv: 2302.01605 [cs.AI]. URL: https://arxiv.org/abs/2302.01605.

[55] Sangdoo Yun et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6023–6032.

[56] Hongyi Zhang et al. "mixup: Beyond empirical risk minimization". In: *arXiv preprint arXiv:1710.09412* (2017).

[57] 平岡 和幸. **囲碁対局における推定勝率のキャリブレーション：*KataGo* の勝率 *8* 割は人間なら何割か**. Tech. rep. 4. 亜細亜大学経営学部データサイエンス学科, 2024.