JAIST Repository
https://dspace.jaist.ac.jp/

Title	More Human-Like Gameplay by Blending Policies From Supervised and Reinforcement Learning
Author(s)	Ogawa, Tatsuyoshi; Hsueh, Chu-Hsuan; Ikeda, Kokolo
Citation	IEEE Transactions on Games, 16(4): 831-843
Issue Date	2024-07-11
Type	Journal Article
Text version	publisher
URL	http://hdl.handle.net/10119/20010
Rights	Copyright (c) 2024 Author(s). Tatsuyoshi Ogawa, Chu-Hsuan Hsueh, and Kokolo Ikeda, IEEE Transactions on Games, vol. 16, no. 4, pp. 831-843. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/). Original publication is available on IEEE Xplore via https://doi.org/10.1109/TG.2024.3424668.
Description	

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

More Human-Like Gameplay by Blending Policies From Supervised and Reinforcement Learning

Tatsuyoshi Ogawa, Chu-Hsuan Hsueh, *Member, IEEE*, and Kokolo Ikeda

***Abstract*—Modeling human players' behaviors in games is a key challenge for making natural computer players, evaluating games, and generating content. To achieve better human–computer interaction, researchers have tried various methods to create human-like artificial intelligence. In chess and *Go*, supervised learning with deep neural networks is known as one of the most effective ways to predict human moves. However, for many other games (e.g., *Shogi*), it is hard to collect a similar amount of game records, resulting in poor move-matching accuracy of the supervised learning. We propose a method to compensate for the weakness of the supervised learning policy by Blending it with an AlphaZero-like reinforcement learning policy. Experiments on *Shogi* showed that the Blend method significantly improved the move-matching accuracy over supervised learning models. Experiments on chess and *Go* with a limited number of game records also showed similar results. The Blend method was effective with both medium and large numbers of games, particularly the medium case. We confirmed the robustness of the Blend model to the parameter and discussed the mechanism why the move-matching accuracy improves. In addition, we showed that the Blend model performed better than existing work that tried to improve the move-matching accuracy.**

***Index Terms*—Board game, human-likeness, player modeling, reinforcement learning, supervised learning.**

I. Introduction

As 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. [1] have shown that the policies of strong game AI are very different from those of humans. This becomes problems when humans try to learn from or enjoy playing against game AIs. In addition, Yannakakis and Togelius [2] stated that human-like agents are important when designing games and generating game content. The reason is that nonhuman-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. [3] 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. [1]. *Maia* used neural networks with similar structures to those used in AlphaZero [4], 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. [5] 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

Manuscript received 1 November 2023; revised 26 March 2024 and 23 May 2024; accepted 27 June 2024. Date of publication 11 July 2024; date of current version 17 December 2024. This work was supported by JST SPRING under Grant JPMJSP2102. Recommended by Associate Editor C. B. Browne.*(Corresponding author: Tatsuyoshi Ogawa.)*

The authors are with the Graduate School of Advanced Science and Technology, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan (e-mail: ogawa.tatsuyoshi@jaist.ac.jp; hsuehch@jaist.ac.jp; kokolo@jaist.ac.jp).

Digital Object Identifier 10.1109/TG.2024.3424668

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

model with 1% 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 model for chess, *Go*, and *Shogi* (Japanese chess-like game) that blends *Maia*-like and AlphaZero-like game AI policies to create more human-like game AI.

This article is extended from a preliminary version studying *Shogi* [6] by applying the Blend model to chess (Section IV) and *Go* (Section V) to test the generalization ability. In addition, we propose Maia-S-Wide in Section III and conduct experiments on all three games, chess, *Go*, and *Shogi* (Section IV to VI). We also compare the original *Maia* with the Blend model using *Maia* in Section IV, compare kullback-leibler (KL)-regularized search [5] with the Blend model using Maia-S-Wide in Section V, and analyze the Blend model using Maia-S-Wide in detail in Section VI.

The rest of this article is organized as follows. Section II introduces related research. Section III proposes Maia-S, Maia-S-Wide, and the Blend model. Sections IV–VI show the experimental results of applying these proposed methods to the target games, chess, *Go*, and *Shogi*. Finally, Section VII concludes this article and discusses future research directions.

II. Related Research

In the field of games, researchers have tried to create 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 [7] 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 G*o* 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 [8] or evaluation functions trained using human players' games [9].

Reinforcement learning is another effective way to create strong game-playing programs, with the advantage of requiring no human game records. AlphaZero [4] 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 cleverly, but the strategies or behaviors are sometimes not like humans. With respect to human-likeness, Togelius et al. [10] 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 [3], [11], [12].

For chess, a program called *Maia* [1] was explicitly designed to imitate human players' moves. This chess AI used deep neural networks for supervised learning. Human players were divided into nine 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% [1]. McIlroy-Young et al. [1] 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. [5] 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 [13] 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 [9] using a transition probability function [8]. Linear combination was used and the weight was trained using human moves. Their proposed method predicted human moves significantly better than each function alone.

III. 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 [1], [5] or 73 000 professional game records in *Go* [5] 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 Section VI-C, 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-likeness.[1] For a given set of positions with humans' moves, move-matching accuracy of a model is the ratio that the model's

[1]Other measures of human-likeness also exist. For example, we used likelihood in addition to move-matching accuracy in our previous work [6]. Since the

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.

1) State space $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$.
2) Action space $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$.
3) Transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$.
4) Reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

Then, the conditional probability $Pr(A = a \mid S = s)$ 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 π_1 and π_2 to create a human-like policy that takes advantage of each. The new policy π_{blend} is calculated as follows:

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

and

$$\pi_{\text{blend}}(a|s) = \frac{P_{s,a}}{\sum_{i=1}^{|A|} P_{s,a_i}} \tag{2}$$

where $\alpha \in [0, 1]$ is used to represent the importance of π_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

$$P_{s,a} = \alpha \times \pi_1(a|s) + (1 - \alpha) \times \pi_2(a|s) \tag{3}$$

or a more general form from both (1) and (3) like

$$P_{s,a} = (\alpha \times \pi_1(a|s)^{\beta} + (1 - \alpha) \times \pi_2(a|s)^{\beta})^{1/\beta}. \tag{4}$$

In this article, we use formula (1) because preliminary experiments showed that (1)'s move-matching accuracy was higher than (3) and was about the same as (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 nonhuman-like mistakes and create policies that are basically human-like.

Maia used 12 million game records to train each of their nine 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 α, i.e., the importance of π_1, on a group or individual basis, it is possible to create π_{blend} suitable for that group or individual. In (1), consider the case where π_1 is a *Maia*-like policy and π_2 is an AlphaZero-like policy. We expect a larger α fits players with lower ratings and a smaller α 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.

1) How many differences are in move-matching accuracy between Maia-S and Maia-S-Wide?
2) How much improvement does the move-matching accuracy of the Blend model have over each individual policy?
3) How does the move-matching accuracy change according to players' ratings?
4) 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.

IV. 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-likeness. 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 (Lc0) 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 six times as many games as each Maia-S policy but with a wider rating range). Section IV-A describes the data and model settings. In Section IV-B, 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 IV-C, we compare the original *Maia* with the Blend model using *Maia*.

A. Data and Model Settings

We used games played on Lichess 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

general tendency of the two measures is similar, we present only move-matching accuracy in this paper to make the results easier to understand.

this platform, players can choose HyperBullet (30 s), Bullet (1 min), Blitz (3–8 min), Rapid (8–15 min), 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

1) we ignored Bullet and HyperBullet games since the move quality was generally lower,
2) we discarded the first 10 plies to ignore frequent patterns in the opening,
3) we discarded moves played when the remaining time was less than 30 s to ensure better move quality,
4) we separated players into nine groups according to their ratings, which were 1100–1199, 1200–1299,..., 1900–1999, and
5) 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. In total, 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.

1) For Maia-S, the total steps of training was 50 000, i.e., 1/8 of *Maia*.
2) 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.
3) 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 π_1) in our Blend model. For the other policy π_2, we employed an AlphaZero-based program, Lc0.[4] More specifically, we used the neural network's policy and did not incorporate the network into tree search. As (1) shows, the Blend model includes the parameter, $\alpha \in [0, 1]$, that determines the influence of π_1. The larger the value of α, the greater the influence of π_1. We performed preliminary experiments to find promising α settings using a grid search with 0.01 increments. In the following sections, each of the Blend model used a setting of α with the highest move-matching accuracy in each rating range's validation data.

[2][Online]. Available: 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 article, we present the results of a single trial for each model, which we consider to be reliable enough.

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

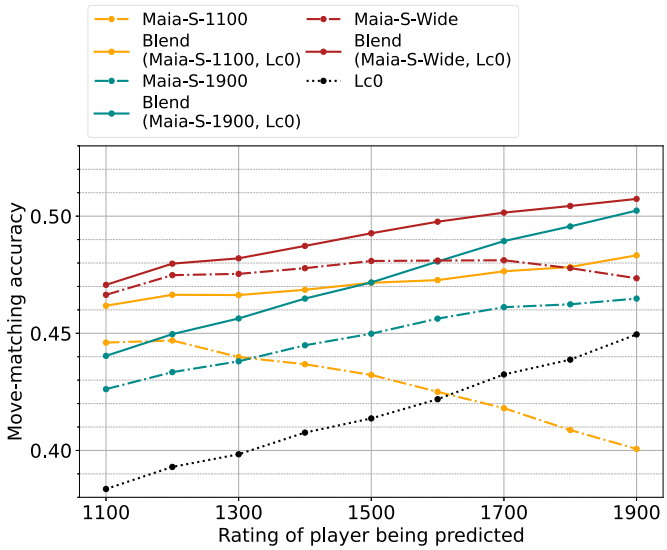

Fig. 1. Move-matching accuracy of Maia-S models, Lc0, and Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rating range) is approximately ±0.002, calculated based on the test data.

B. Fundamental Analysis

In this section, we provide fundamental analyses of the experimental results. Fig. 1 shows the move-matching accuracy of Maia-S-1100, Maia-S-1900, Maia-S-Wide, Lc0, 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 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 Lc0, 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.

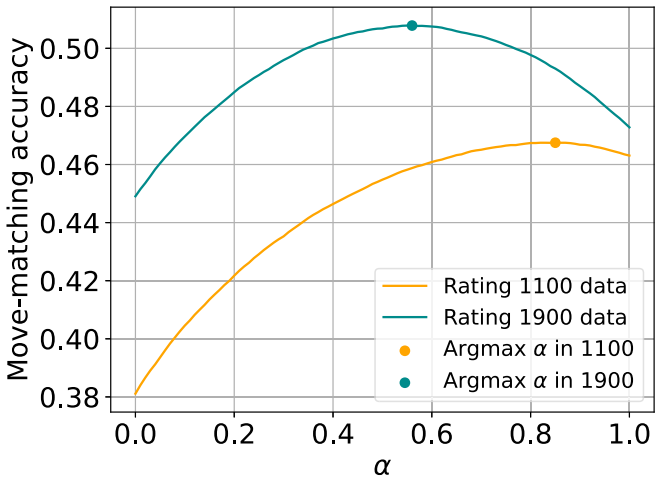

Fig. 2. Move-matching accuracy of Blend (Maia-S-Wide, Lc0) to the rating 1100 and 1900 data for different α using validation data.

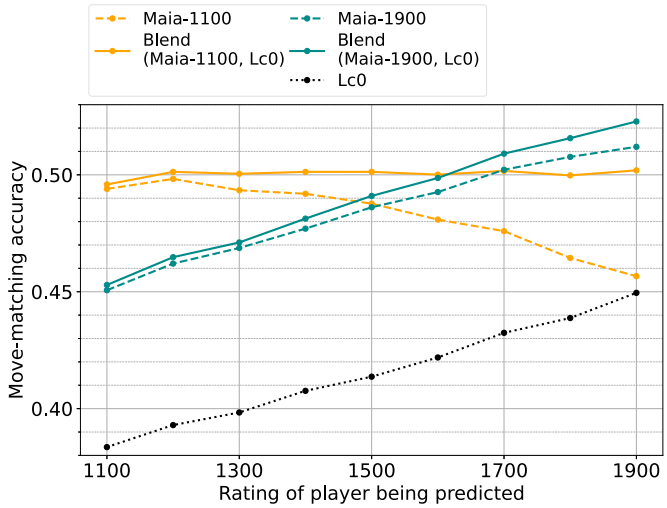

Fig. 3. Move-matching accuracy of *Maia* models, Lc0, and Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rating range) is approximately ± 0.002, calculated based on the test data.

The move-matching accuracy of Lc0 (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 Lc0, 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 α 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 α 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 Lc0. 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 α of 0.85 for the rating 1100 data, and was 0.034 higher than Maia-S-Wide with the best α 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.

We investigated the robustness of the Blend model to α, i.e., the influence of π_1. We targeted the Blend (Maia-S-Wide, Lc0) model, which had the highest move-matching accuracy over the validation data in chess. Fig. 2 shows the move-matching accuracy of the Blend (Maia-S-Wide, Lc0) model for the rating 1100 and 1900 data with different α. The x-axis represents α 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 α for the validation data. The best α for the rating 1100 was higher than the rating 1900. Namely, π_1, i.e., Maia-S-Wide, should be weighted more than Lc0 when predicting low-rated players' moves compared to high-rated players'.

Generally speaking, the appropriate α depends on the target of imitation. When we want to imitate stronger players, the reinforcement learning policy (Lc0) should be weighted higher (i.e., smaller α). Regarding the robustness, even if α deviates from the optimal value by 0.1, the move-matching accuracy only drops about 0.001 to 0.002. Therefore, α is not a very sensitive parameter, although it needs to be optimized depending on the target data.

C. *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. Fig. 3 shows the move-matching accuracy of Maia, Lc0, 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 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 Lc0, 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 α 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 α 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 Lc0. 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.

V. 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 [4], [7], solving smaller boards [14], and teaching human players [15], [16].

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 (siox times as many games as each Maia-S policy but with a wider rank range). Section V-A describes the data and model settings. In Section V-B, 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 Section V-C, 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.

A. 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 18 k) to experts (such as 9-dan or 9 d).[6] The board size of 19 $\times$ 19 is the most popular, and players can choose the thinking time settings of 20 min 60 s, 5 min 30 s, and 1 min 20 s. In Fox Go, 20 min 60 s means a main time of 20 min and three times of 60-s byo-yomi. In this rule, each player is first given 20 min to play, and once the 20 min are used up, the player must play each move in 60 s; if the player goes over 60 s 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-S and Maia-S-Wide policies. We separated players into twelve groups according to their rank, which were, 6 k, 5 k,..., 2 k, 1 k, 1 d, 2 d,..., 8 d, and 9 d. 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-3 k and Maia-S-6 d using 120 000 games of 3 k players and 6 d players, respectively. For Maia-S-Wide, we used 80 000 game records from each of the ranks between 3 k and 6 d, 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 [17], 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 π_1) in our Blend model. For the other policy π_2, we employed an AlphaZero-based program, KataGo.[9] Similar to the experiments on chess in Section IV, we used the policy of KataGo's neural network and tuned the parameter α for the Blend model using each rank's validation data.

B. Fundamental Analysis

In this section, we provide fundamental analyses of the experimental results. Fig. 4 shows the move-matching accuracy of Maia-S-3 k, Maia-S-6 d, Maia-S-Wide, KataGo, and the Blend models tested on the test data of all the 15 rank groups 6 k to 9 d. 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 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 3 k players, blue represents using the training data for 6 d players, red represents using the training data for 3 k to 6 d players, and

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

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

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

[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][Online]. Available: https://github.com/lightvector/KataGo

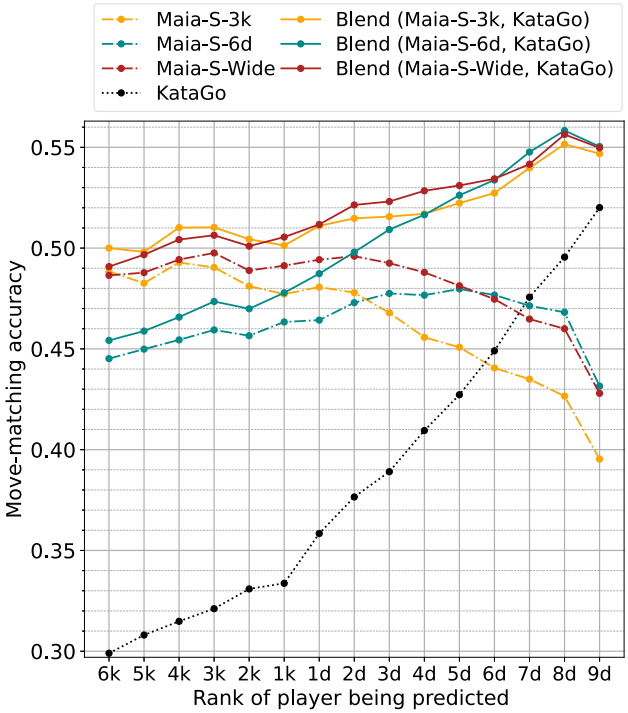

Fig. 4. Move-matching accuracy of Maia-S models, KataGo, and the Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rank) is approximately ±0.004, calculated based on the test data.

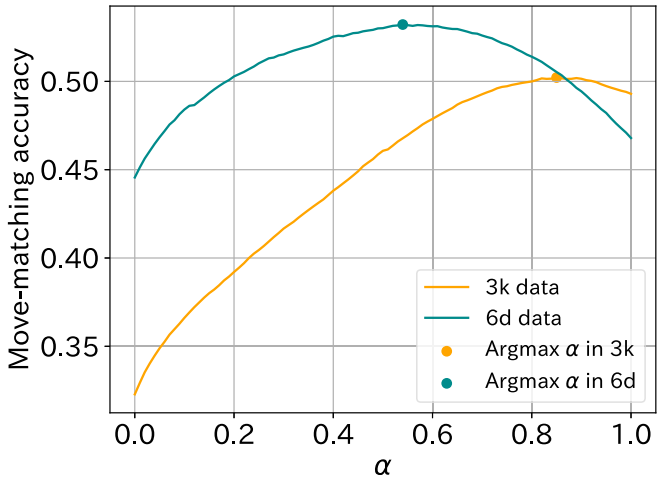

Fig. 5. Move-matching accuracy of Blend (Maia-S-Wide, KataGo) to 3 k and 6 d data for different α using validation data.

black represents KataGo, which did not use human game records for its training data.

From the move-matching accuracy of Maia-S-3 k (yellow chain curve) and Maia-S-6 d (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-3 k for the 3 k data but 0.002 lower than Maia-S-6 d for the 6 d 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-6 d even for the 6 d 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-3 k, KataGo) model (the yellow solid curve) was 0.020 higher than Maia-S-3 k with the best α of 0.78 for the 3 k data. The move-matching accuracy of the Blend (Maia-S-6 d, KataGo) model (the blue solid curve) was 0.057 higher than Maia-S-6 d with the best α of 0.58 for the 6 d 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 α of 0.85 for the 3 k data, and was 0.060 higher than Maia-S-Wide with the best α of 0.54 for the 6 d 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 α, i.e., the influence of π_1. We targeted the Blend (Maia-S-Wide, KataGo) model, which had the highest move-matching accuracy over the validation data in Go. Fig. 5 shows the move-matching accuracy of the Blend (Maia-S-Wide, KataGo) model for 3 k and 6 d data with different α. The x-axis represents α 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 3 k players (low-rank players), and blue represents the validation data for 6 d players (high-rank players). The dots represent the best α for the validation data. The best α for 3 k was higher than 6 d. Namely, π_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 α 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 α). Regarding the robustness, even if α deviates from the optimal value by 0.1, the move-matching accuracy only drops about 0.001 to 0.002. Therefore, α is not a very sensitive parameter, although it needs to be optimized depending on the target data.

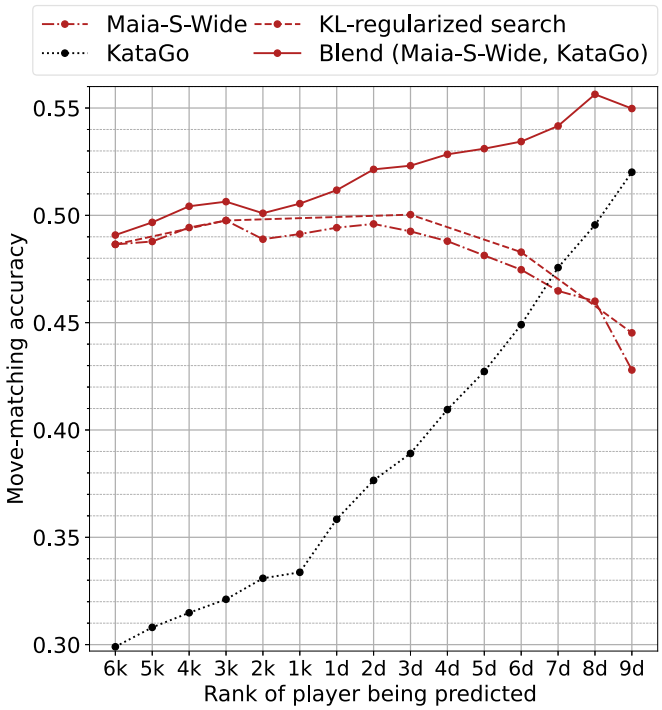

Fig. 6. Move-matching accuracy of Maia-S-Wide model, KataGo, Blend model, and KL-regularized search. The 95% confidence interval of each data point (i.e., move-matching accuracy for a rank) is approximately ±0.004, calculated based on the test data.

C. Comparison With KL-Regularized Search

In addition to our Blend model, the KL-regularized search [5] 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 ∞, 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 ∞. 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 6 k, 3 k, 3 d, 6 d, and 9 d.

In the following, we compare our Blend model with the KL-regularized search, where both used the Maia-S-Wide neural network. Fig. 6 shows the move-matching accuracy of Maia-S-Wide, KataGo, the Blend model, and KL-regularized search tested on the test data of 6 k to 9 d. 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 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 3 k to 6 d 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 3 k data and was 0.008 higher than Maia-S-Wide with the best c_{puct} of 1 for the 6 d 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.

VI. 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 [18], which is also one of the three targets of AlphaZero [4].

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 six times as many games as each Maia-S policy but with a wider rating range). Section VI-A describes the data and model settings. In Section VI-B, 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 Section VI-C, we investigate the mechanism why the Blend modes can better predict human moves.

A. 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, 5, or 10-min 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-min games and did some filtering on the game records, similar to *Maia* in chess. The three conditions are explained as follows.

1) 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.
2) 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.

3) 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.

1) *Group 1:* R1433–R1591.
2) *Group 2:* R1592–R1655.
3) *Group 3:* R1656–R1708.
4) *Group 4:* R1709–R1768.
5) *Group 5:* R1769–R1855.
6) *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. In total, 90% of data in each group were training data, 5% were validation data, and the remaining 5% were test data for evaluation. We performed multitask supervised learning, such as *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 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 ten epochs of training for each group. This is because we observed that the loss often converged at around ten epochs.

These Maia-S policies served as one of the two policies (say π_1) in our Blend model. For the other policy π_2, we employed an AlphaZero-based program, DLshogi.[11] Similar to the experiments on chess in Section IV, we used the policy of DLshogi's neural network and tuned the parameter α for the Blend model using each group's validation data.

B. Fundamental Analysis

In this section, we provide fundamental analyses of the experimental results. Fig. 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.

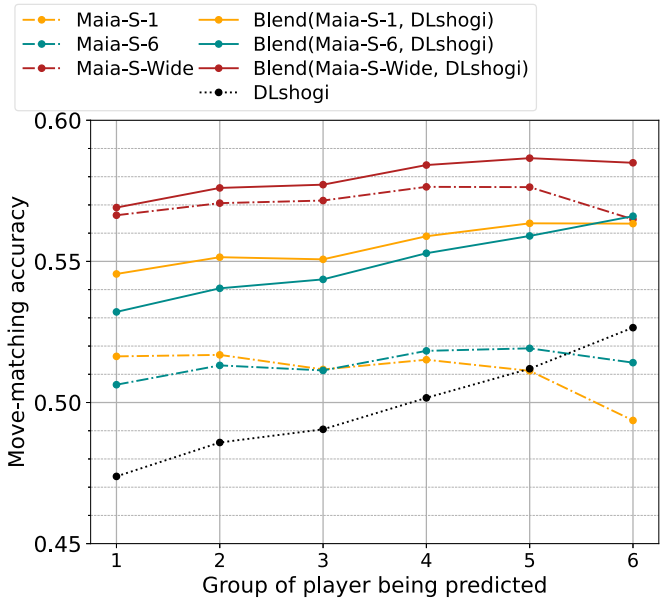

Fig. 7. Move-matching accuracy of Maia-S models, DLshogi, and Blend models. The 95% confidence interval of each data point (i.e., move-matching accuracy for a group) is approximately ±0.002, calculated based on the test data.

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 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.

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

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

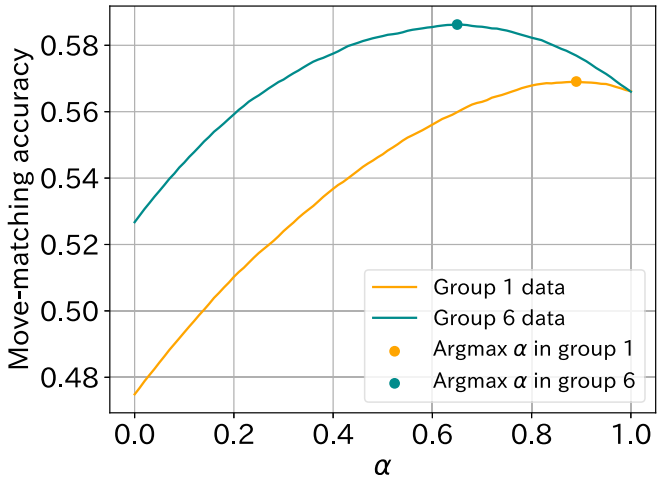

Fig. 8. Move-matching accuracy of Blend (Maia-S-Wide, DLshogi) to group 1 and group 6 data for different α using validation data.

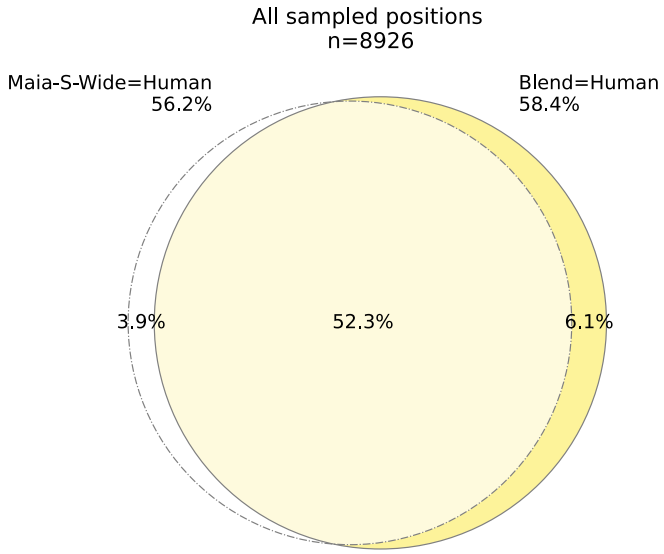

Fig. 9. Venn diagram of Maia-S-Wide and the Blend model's prediction results of the all experimented positions.

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 α 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 α 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 α of 0.89 for group 1 data, and was 0.020 higher than Maia-S-Wide with the best α 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 α, i.e., the influence of π_1. We targeted the Blend (Maia-S-Wide, DLshogi) model, which had the highest move-matching accuracy over the validation data in Shogi. Fig. 8 shows the move-matching accuracy of the Blend (Maia-S-Wide, DLshogi) model for group 1 and group 6 data with different α. The x-axis represents α 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 α for the validation data. The best α for Group 1 was higher than Group 6. Namely, π_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 α 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 α). Regarding the robustness, even if α deviates from the optimal value by 0.1, the move-matching accuracy only drops about 0.001 to 0.002. Therefore, α is not a very sensitive parameter, although it needs to be optimized depending on the target data.

C. Analysis of Mechanisms of Blend Model

We investigate the mechanism why the Blend model improves move-matching accuracy. We used test data from Group 6 in Section VI-A, which had the highest move-matching accuracy and better blend effects. For our verification experiments, we randomly sampled approximately 9000 positions where the final wins/losses were unknown.[12]

Fig. 9 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 positions, the Blend model improved move-matching accuracy by 2.2% (58.4% minus 56.2%). The amount of improvement matched the results in Fig. 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

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

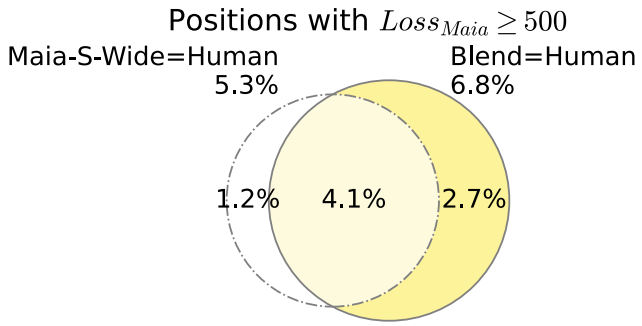

Fig. 10. Venn diagram of Maia-S-Wide and the Blend model's prediction results of the experimented positions with $\text{Loss}_{\text{Maia}} \geq 500$.

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 (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 B. 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., $\text{Loss}_{\text{Maia}} \geq 500$. Fig. 10 shows the Venn diagram of the prediction results of the experimented positions with $\text{Loss}_{\text{Maia}} \geq 500$ ($n = 1466$). Similar to Fig. 9, the chain-line and solid-line circles contain positions that Maia-S-Wide's and Blend's moves matched humans', respectively. The 1466 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 Fig. 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.

VII. 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 3 k), 0.534 for *Go* advanced players (rank 6 d), 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 α 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.

1) 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.
2) 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.
3) 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 nongame 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

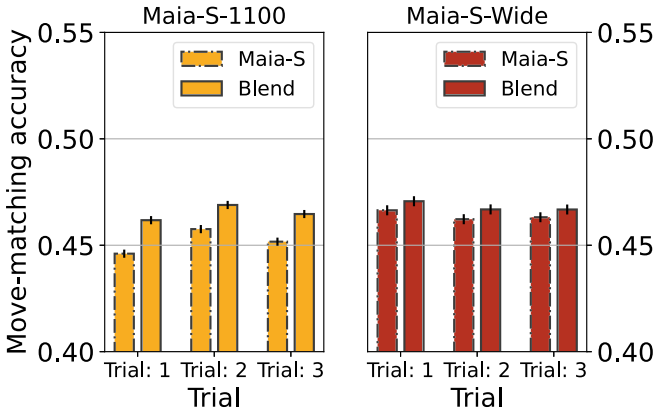

Fig. 11. Move-matching accuracy of Maia-S-1100, Maia-S-Wide, and their Blend models on rating 1100 data for each trial.

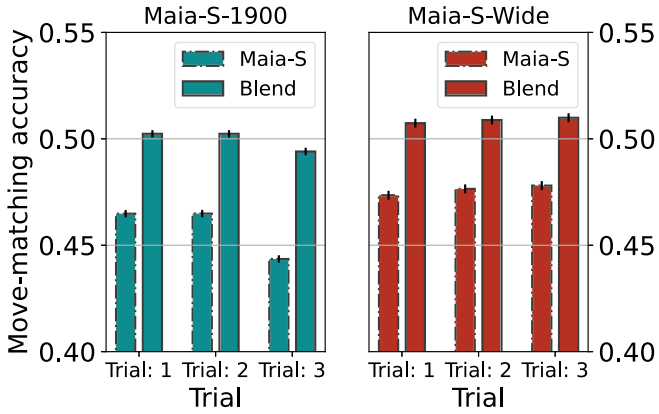

Fig. 12. Move-matching accuracy of Maia-S-1900, Maia-S-Wide, and their Blend models on rating 1900 data for each trial.

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.

Appendix A
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 Section IV-A. 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.

Fig. 11 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.

We also conducted experiments on rating 1900 data, and the results are shown in Fig. 12. The only difference in representation between Figs. 11 and 12 is that yellow bars becomes blue bars, which means using the training data of rating 1100 and 1900, respectively. Similar to Fig. 11, 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.

Appendix B
The Definition of Loss

In Section VI-C, 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 $\text{move}_{\text{Human}}$ be the human move, $\text{move}_{\text{Maia}}$ be the move that Maia-S-Wide assigns the highest probability to, $\text{move}_{\text{DLshogi}}$ be the move that DLshogi assigns the highest probability to, and $\text{move}_{\text{Blend}}$ be the move that the Blend model assigns the highest probability to. Also, let the evaluation of each move (CP: centipawn) be $\text{CP}_{\text{Human}}, \text{CP}_{\text{Maia}}$, $\text{CP}_{\text{DLshogi}}$, and CP_{Blend}. We then define the best evaluation at the position CP_{Best} and move m's loss Loss_m as follows:

$$\text{CP}_{\text{Best}} = \max(\{\text{CP}_{\text{Human}}, \text{CP}_{\text{Maia}}, \text{CP}_{\text{DLshogi}}, \text{CP}_{\text{Blend}}\})$$
$$\text{Loss}_m = \text{CP}_{\text{Best}} - \text{CP}_m.$$

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 [18], 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 [19] 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.

REFERENCES

[1] R. McIlroy-Young, S. Sen, J. Kleinberg, and A. Anderson, "Aligning superhuman AI with human behavior: Chess as a model system," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1677–1687.

[2] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Berlin, Germany: Springer, 2018, vol. 2.

[3] N. Fujii, Y. Sato, H. Wakama, K. Kazai, and H. Katayose, "Evaluating human-like behaviors of video-game agents autonomously acquired with biological constraints," in *Proc. Int. Conf. Adv. Comput. Entertainment Technol.*, Springer, 2013, pp. 61–76.

[4] D. Silver et al., "A general reinforcement learning algorithm that masters chess, Shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aar6404

[5] A. P. Jacob et al., "Modeling strong and human-like gameplay with KL-regularized search," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 9695–9728.

[6] T. Ogawa, C. Hsueh, and K. Ikeda, "Improving the human-likeness of game AI's moves by combining multiple prediction models," in *Proc. 15th Int. Conf. Agents Artif. Intell.-Volume 3: ICAART, INSTICC*, SciTePress, 2023, pp. 931–939.

[7] R. Coulom, "Computing "Elo ratings" of move patterns in the game of go," *ICGA J.*, vol. 30, no. 4, pp. 198–208, 2007.

[8] Y. Tsuruoka, D. Yokoyama, and T. Chikayama, "Game-tree search algorithm based on realization probability," *ICGA J.*, vol. 25, no. 3, pp. 145–152, 2002.

[9] K. Hoki and T. Kaneko, "Large-scale optimization for evaluation functions with minimax search," *J. Artif. Intell. Res.*, vol. 49, pp. 527–568, 2014.

[10] J. Togelius, G. N. Yannakakis, S. Karakovskiy, and N. Shaker, "Assessing believability," in *Believable Bots*. Berlin, Germany: Springer, 2013, pp. 215–230.

[11] P. Hingston, "A new design for a turing test for bots," in *Proc. 2010 IEEE Conf. Comput. Intell. Games*, 2010, pp. 345–350.

[12] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Adv. Neural Inf. Process. Syst.*, vol. 29, pp. 4572–4580, 2016.

[13] T. Kinebuchi and T. Ito, "Shogi program that selects natural moves by considering the flow of preceding moves," in *Proc. IEEE 2015 3rd Int. Conf. Appl. Comput. Inf. Technol./2nd Int. Conf. Comput. Sci. Intell.*, 2015, pp. 79–84.

[14] E. C. van der Werf and M. H. Winands, "Solving go for rectangular boards," *ICGA J.*, vol. 32, no. 2, pp. 77–88, 2009.

[15] K. Ikeda, S. Viennot, and N. Sato, "Detection and labeling of bad moves for coaching Go," in *Proc. 2016 IEEE Conf. Comput. Intell. Games*, 2016, pp. 1–8.

[16] C.-H. Hsueh and K. Ikeda, "Improvement of move naturalness for playing good-quality games with middle-level players," *Appl. Intell.*, vol. 54, no. 2, pp. 1637–1655, 2024.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. 2016 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[18] H. Iida, M. Sakuta, and J. Rollason, "Computer Shogi," *Artif. Intell.*, vol. 134, no. 1, pp. 121–144, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370201001576

[19] "YaneuraOu Suisho5 Shogi engine," Accessed: Mar. 15, 2023. [Online]. Available: https://github.com/mizar/YaneuraOu/releases/tag/v7.5.0,