Rylee: Creating a Deployable Human-Like Chess Engine to Enhance the Learning Experience

1st Ethan Gee
Computer Science
Utah State University
Logan, UT, USA
ethan.gee@usu.edu

2nd Nate Stott
Computer Science
Utah State University
Logan, UT, USA
nate.stott@usu.edu

***Abstract*—Artificial Intelligence (AI) has surpassed human performance in many domains; since 2005, chess engines have demonstrated consistent superhuman play. However, fields such as law, medicine, and human resources cannot directly deploy superhuman AI due to ethical constraints. These limitations, do not preclude the use of AI systems designed to complement—rather than exceed—human performance. Prior work shows that aligning AI with human-observable behavior, instead of explaining behavior, can yield desirable results. Chess provides a strong testbed for developing such systems. Rylee is a neural-network-based chess engine trained to mimic human move selection, aiming for lightweight computation and modest predictive accuracy. Traditional chess engines, optimized for perfect play, offer limited value as human training tools, and attempts to attenuate them fail to capture human-like behavior. Maia improves on this but requires substantial compute to train and run. This work builds on Maia by developing a more computationally efficient, human-aligned model—comparable in size to Stockfish and capable to run on standard hardware. Trained on large Lichess datasets across multiple rating levels, Rylee learns to predict human moves with modest success. Key results show that Rylee achieves 25% Top-1 accuracy and 51% Top-5 accuracy in predicting human moves using a model 30 times smaller than Maia (800,000 parameters vs. 25 million), and experiments demonstrate that the model generalizes well across training and validation sets with minimal overfitting while being trainable on edge devices in 2-3 days compared to Maia's 3-4 weeks on high-end GPUs. Furthermore, Rylee expands capabilities beyond Maia by including opening move predictions (first 10 moves), supporting a broader ELO range (700-2500 vs. 1100-1900), and incorporating diverse game types rather than filtering to only blitz games. The broader significance of this research lies in advancing human-aligned AI and enabling more accessible, engaging AI-based training systems across diverse domains.**

***Index Terms*—Chess, Neural Networks, Machine Learning, Artificial Intelligence, AI, Human-like AI, Deep Learning**

I. Introduction

Since the 1940s, chess-playing programs, or "chessbots," have evolved to surpass the capabilities of the most skilled human players. These bots are now integral to modern chess training, with players globally using them to refine their skills. However, a significant disparity exists between the strategies employed by traditional chessbots and human players. Human chess is characterized by the use of heuristics, intuition, and pattern recognition. In contrast, conventional chessbots rely on brute-force computation, analyzing vast numbers of potential future board states to select the move that maximizes their probability of winning.

This computational superiority creates an imbalance in human-computer matches, where the bot's expansive memory and processing power provide a decisive advantage. To mitigate this, bots are often programmed to introduce deliberate errors, resulting in an unnatural and often confusing experience for the human player. This presents a paradox: while chessbots dominate the game, their non-human-like mode of play limits their effectiveness as a training tool for humans.

This project addresses the challenge of creating a more "human-like" chessbot. Our work builds upon the research of McIlroy-Young et al. on the Maia models, which were trained using supervised learning to emulate human decision-making, a departure from the reinforcement learning approach common to engines like AlphaZero. While Maia demonstrated strong performance in predicting human moves, its reliance on the large and resource-intensive AlphaZero architecture limits its practical application. Conversely, Stockfish, a significantly smaller engine, achieves comparable performance to AlphaZero, indicating that model size does not scale linearly with performance.

The primary objective of this project is to develop a chessbot that not only plays like a human but is also computationally efficient enough to run on standard consumer hardware. We aim to achieve a level of human-like play comparable to the Maia models but with a model deployable to edge devices. To this end, we will utilize the Lichess Dataset, a comprehensive collection of human chess games and player ratings, which is the same dataset employed in the training of Maia. This work contributes to the growing field of human-aligned AI by exploring the trade-offs between model size, performance, and human-like behavior in the domain of chess.

II. Related Work

A. Chess Engines

While early chess engines relied on traditional search algorithms like minimax and alpha-beta pruning, the field has been transformed by neural network approaches. These early engines, including Deep Blue, used handcrafted evaluation functions that required extensive manual tuning of positional factors.

The introduction of deep neural networks marked a paradigm shift in computer chess. Modern engines employ sophisticated architectures: AlphaZero uses a deep residual network with 20 residual blocks, each containing convolutional layers and batch normalization. This network processes the $8{\times}8{\times}73$ board representation to output both a position evaluation and a policy distribution over possible moves. Leela Chess Zero (LC0), an open-source implementation of similar principles, utilizes a network with 24 residual blocks and generates 256 channels in its intermediate layers.

Maia, specifically designed for human-like play, employs a modified version of LC0's architecture. Instead of learning through self-play, Maia is trained on millions of human games using supervised learning. Its network consists of multiple convolutional layers followed by a policy head that predicts move probabilities and a value head that evaluates positions. Maia processes the board state through 12 input planes (6 piece types $\times$ 2 colors) and incorporates additional features like player ratings. Recent hybrid approaches, such as Stockfish's NNUE (Efficiently Updatable Neural Network), combine traditional search with a smaller neural network containing roughly 40,000 parameters that can be efficiently updated during search.

B. Human-Like Chess AI

While the pursuit of optimal play has been the primary focus of chess engine development, there is a growing interest in creating AI that plays in a more human-like manner. The Maia Project, by McIlroy-Young et al., represents a significant step in this direction. Instead of using reinforcement learning to find the best possible move, Maia is trained on a large dataset of human games to predict the move a human player would make in a given position. This supervised learning approach results in an engine that exhibits more human-like characteristics, including making mistakes that are typical of human players at a certain skill levels instead of the always the optimal move. Importantly, the model is never required to intentionally choose a sub-optimal move. Because the task is reframed as predicting the move a human would make, the notion of an "optimal" move in the traditional engine sense no longer applies. This leads to experiences humans can easily learn from.

C. Dataset and Training Data

The Lichess database has emerged as the standard dataset for human chess modeling research, providing unprecedented scale and diversity spanning a wide range of player skill levels from beginner to grandmaster.

Prior work has leveraged this dataset extensively. The Maia Project utilized 169 million Lichess games filtered to specific rating bands (1100-1900 ELO) and game types (blitz time controls only), training separate models for each 100-point rating interval. This approach enabled rating-specific modeling but required substantial computational resources for data processing and model training. Other researchers have used the Lichess dataset for opening theory analysis, playing style detection, and cheat detection, demonstrating its versatility beyond move prediction tasks.

III. Proposed Method

A. System Architecture

We created Rylee, a chess engine, which can process Lichess data, train with tunable parameters, and deploy into a playable environment. First, automating Lichess game file downloads, then processing to a consumption ready format stored in the database. Model training draws directly from this preprocessed dataset via a custom Pytorch Dataset class. Lastly, models can be manually evaluated using a custom chess game implementation; a human, Stockfish, or Leela player can be used as the opposing player.

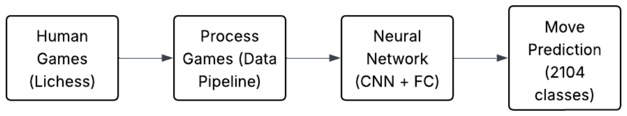
Human Games (Lichess)
Process Games (Data Pipeline)
Neural Network (CNN + FC)
Move Prediction (2104 classes)

Fig. 1. A High level system architecture diagram showcasing the full data flow from Lichess and processing to training and move prediction.

B. Data Collection and Processing

1) PGN File Processing: Lichess data is stored in Portable Game Notation (PGN) files. PGN is made up of two parts the tag pairs, which stores game metadata (e.g. game duration, white player elo, winner), and the move sequence (e.g. e2e4, e1g1) stored in Universal Chess Interface (UCI).

For each game useful tags are extracted including the players elos. Next we go though the UCI section creating "snapshots" of the board at every step, we combine the snapshots with the move the player made on their turn, and the elo of both players (we call this metadata). Finally each board snapshot and metadata pair are represented using three dimensional tensors that are 8 by 8 (due to the size of the board) with 12 channels (6 channels for the white pieces and 6 for black).

C. Neural Network Architecture

The neural network architecture consists of three main sections: convolutional layers for processing board piece positions (pieces are spatially related), a fully connected network for feature learning and dimensionality reduction, and two output heads for human move and auxiliary predictions (predicting the legal moves available to the player).

The convolutional section processes the $8{\times}8{\times}12$ encoded board through six convolutional layers. Each layer uses 64 filters with $8{\times}8$ kernels utilizing "same" padding. After each convolution, ReLU activation functions are applied to ensure non-linearity.

The fully connected section starts with a 4100-dimensional input (flattened board features + metadata) and progressively reduces dimensionality through six layers ($512 \rightarrow 32 \rightarrow 32 \rightarrow 32 \rightarrow 32 \rightarrow 32$) with ReLU activations. This

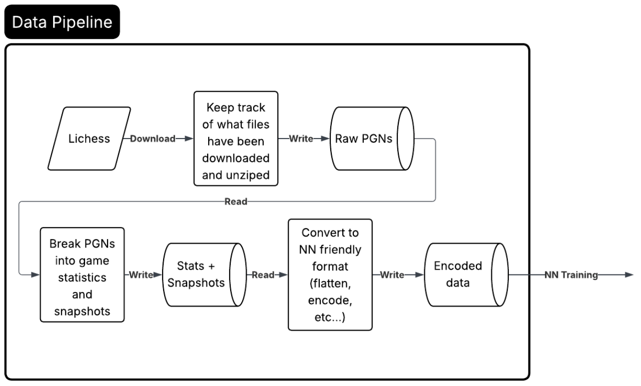
Data Pipeline
Lichess
Download
Keep track of what files have been downloaded and unziped
Write
Raw PGNs
Read
Break PGNs into game statistics and snapshots
Write
Stats + Snapshots
Read
Convert to NN friendly format (flatten, encode, etc...)
Write
Encoded data
NN Training

Fig. 2. A detailed data processing pipeline diagram showcasing the process starting from Lichess PGN files to neural network training data. Steps include file parsing, snapshot and metadata extraction, and encoding.

creates a 32-dimensional embedding of the board state and metadata.

This architecture balances efficiency and complexity to capture human chess playing patterns. The dual head design allows complemented learning of move prediction with the learning of chess rules.

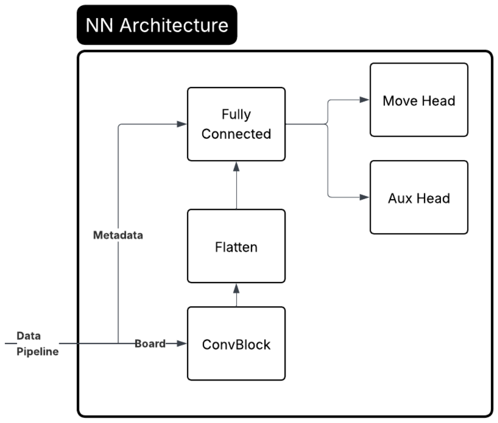

Fig. 3. A neural network architecture diagram showcasing the convolutional, flatten, fully connected layers, with the dual output heads for player move and legal move predictions.

D. Training Process

1) Training Parameters: Began training model with the following configurations:

- Batch size: 512
- Initial learning rate: 0.001
- Weight decay: 1e-4
- Beta parameters: (0.9, 0.999)
- Number of epochs: 50

2) Optimization Strategy: The explodable combinations used on the random search for hyperparameter optimization:

- Learning rates: [0.1, 0.01, 0.001, 0.0001]
- Weight decay rates: [1e-3, 1e-4, 1e-5]
- Beta values: [0.9, 0.95, 0.99]
- Momentum values: [0.9, 0.95, 0.99]

The best configuration was determined by the lowest validation loss.

3) Hardware Configuration: Training was done on embedded hardware (keep in mind a Jetson Orin Nano is pretty close to a Raspberry Pi):

- Platform: NVIDIA Jetson Orin Nano 8GB Developer Kit
- GPU: 1024-core NVIDIA Ampere architecture GPU
- CPU: 6-core Arm Cortex-A78AE v8.2 64-bit CPU
- RAM: 8GB 128-bit LPDDR5
- Storage: 64GB eMMC 5.1

IV. Experiments

A. Dataset Description

We trained our model off of 15,167 games from January of 2013. These games were played by a wide breadth of players in a variety of states. The original Maia model was split into 9 separate models split apart by elo ranges of 100 (e.g. 1100-1200) this limited the models generalizability and so we trained our model on one complete dataset to increase generalizability. Expanding from 1100-1900 to 700-2500 to further increase access.

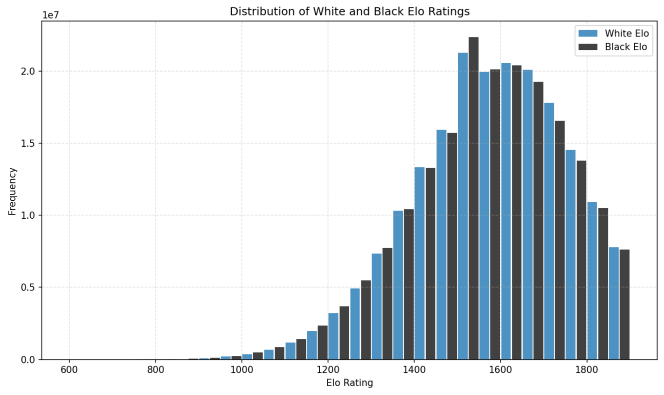

Fig. 4. Spread of the elos present in the training set.

One other key difference is the fact that both iterations of the Maia Paper are unable to play chess openings as they don't have any knowledge of the first 10 moves. This is a profound limitation as the opening is one of the key areas where players struggle within a game. Knowing this we choose to include the first 10 moves to increase the understanding of openings.

We also didn't limit the model to just the blitz games as we wanted to increase the overall variability in the games the model was capable of representing. We introduced this noise as it gives a far more holistic view of human players. This was useful as blitz games only show up a small portion of the data and we want to show all human play.

For training, we used 15,167 human-rated games from January of 2013 from players rated 700-2500 ELO, generating approximately 1 million board snapshots. The dataset was

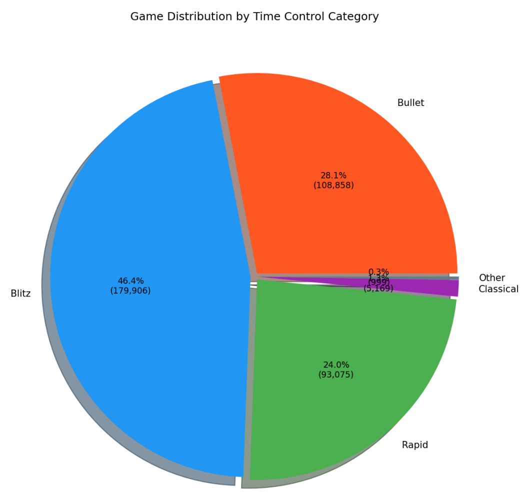

Fig. 5. Time Control Splits of the games from January of 2013

split into 80% training (800,000 snapshots), 10% validation (100,000 snapshots), and 10% test (100,000 snapshots). Games were validated to remove incomplete or corrupted entries.

B. Baseline Description

To evaluate Rylee's performance in predicting human-like moves, we compared against five baseline models representing different approaches to chess move prediction:

Random Move Selection: As a lower bound, we implemented a baseline that randomly selects from all legal moves in a given position, reflecting the average branching factor in chess positions.

Random Forest Classifier: We trained a Random Forest model on the same dataset to establish a classical machine learning baseline, demonstrating that simple pattern recognition can capture some human behavior but falls short of deep learning approaches.

Stockfish 15: A traditional minimax-based chess engine with alpha-beta pruning and hand-crafted evaluation functions. While Stockfish plays near-perfect chess, confirming that optimal play differs significantly from human play.

Leela Chess Zero (LC0) 4200: A neural network-based engine trained through self-play reinforcement learning, similar to AlphaZero. Despite using neural networks, LC0 optimizes for winning rather than human-like behavior.

Maia-1 1500: The primary comparison point for our work. Maia-1 is a supervised learning model specifically trained to predict human moves at the 1500 ELO level, requiring 25 million parameters and substantial computational resources for training.

Evaluation Metrics: We use Top-1 accuracy (percentage of positions where the predicted move exactly matches the human move) and Top-5 accuracy (percentage where the human move appears in the top 5 predictions) as our primary metrics. These directly measure alignment with human decision-making. We also report filtered accuracy, which excludes noisy board states

TABLE I
COMPARISON OF MODEL ACCURACY

Name	Accuracy
Random Moves	6%
Random Forest	13%
Stockfish	40%
Leela 4200	44%
Maia1 1500	51%
Rylee Fully Connected	3.5%
Rylee Conv	23.5%
Rylee Conv with Aux	25%
Rylee Conv with Aux Filtered	35%

(blitz games only, exclude first 10 moves, filter games between narrow ELO ranges).

All of the source code for this project is a available at https://github.com/EthanDGee/ryleeeeeeeeeeeee

C. Experimental Evaluation

Our initial approach focused on training a model that is similar in scale and architecture to the StockFish series of models. These models are extremely small, and currently is the second strongest chess bot in the market. Even better it can be deployed easily to edge devices due to it's relatively small size, as it only consists of 6 fully connected layers. It should be mentioned that it's neural engines deployability is due to it's more limited role of simply evaluating board states, and its NNUE (Efficiently Updatable Neural Network) architecture. Thus, it's strength is due to it's ability to effectively operate in a minimax search system. That being said we trained from that starting point. After training we achieved a meager score of around 3.5% accuracy.

Seeking to improve we added convolutional layers to board state evaluation. Adding these layers in in line with our humanistic approach has humans also use spacial features of the chess board to evaluate next moves (e.g. Knights are valued higher near the center of the board). This lead to a massive breakthrough in performance increasing accuracy to 23.5%.

One further problem a machine learning based approach faces is learning the rules of chess. To improve this Maia added an auxiliary head to improve the models performance by having it predict the legal moves possible from a given board state. This was shown to have a profound impact. Implementing the auxiliary head into Rylee lead to accuracy jumping to around 25%.

1) Training and Validation Performance: Our final model with convolutional layers and auxiliary head demonstrated strong performance and generalization. Table II shows the comprehensive metrics across training and validation sets.

The close alignment between training and validation metrics indicates minimal overfitting, suggesting that Rylee has successfully captured generalizable patterns in human chess decision-making rather than memorizing specific positions. The filtered accuracy metrics, which exclude noisy board states, show that Rylee achieves 35% Top-1 accuracy on non-noisy board states.

TABLE II
TRAINING AND VALIDATION METRICS FOR RYLEE CONV WITH AUX

Metric	Training	Validation
Loss	0.0152	0.0164
Top-1 Accuracy	27%	25%
Top-5 Accuracy	53%	51%
Top-1 Accuracy (Filtered)	36%	35%
Top-5 Accuracy (Filtered)	54%	53%

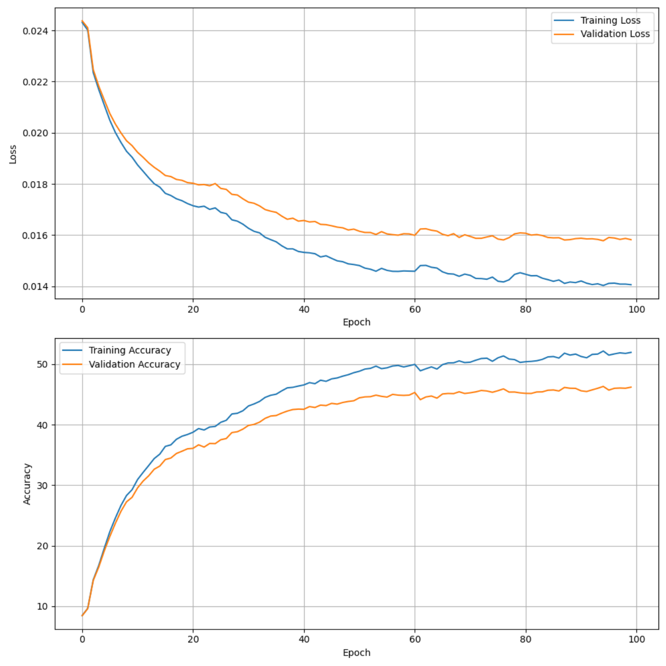

Fig. 6. Training and validation curves showing loss and accuracy over 100 epochs. The close alignment between training and validation metrics demonstrates strong generalization with minimal overfitting.

2) Computational Efficiency: A key contribution of this work is demonstrating that human-aligned chess AI can be trained and deployed on edge devices:

- **Training Time:** 2-3 days on NVIDIA Jetson Orin Nano (vs. 3-4 weeks for Maia on dual A100 GPUs)
- **Preprocessing Time:** 1.5 hours (vs. 8 days for Maia)
- **Model Size:** 800,000 parameters (30× smaller than Maia)
- **Inference Speed:** 10,000 positions/second on edge hardware
- **Memory Footprint:** ¡50MB for model weights (deployable on Chromebooks and Raspberry Pi)

3) Comparison to State-of-the-Art: While Rylee's Top-1 accuracy is lower than Maia's, this comparison must be contextualized by the substantial differences in model size, training data scale, and computational requirements detailed above. The performance gap is expected given these constraints, but Rylee's Top-5 accuracy demonstrates that it captures human-like thinking in over half of positions when considering the top 5 candidate moves.

V. CONCLUSION AND FUTURE WORK

A. Summary

We successfully developed Rylee, a compact neural network-based chess engine for human-aligned chess training. The key achievement is demonstrating that meaningful human-aligned behavior can be captured with far fewer resources than existing approaches, opening the door to accessible AI-based chess training tools. Limitations include the accuracy gap compared to larger models, and training on a limited time range of data (January 2013).

B. Future Work

1) Model Improvements: Several architectural enhancements could improve Rylee's accuracy while maintaining deployability:

Deeper Networks: Increasing the number of convolutional layers from 6 to 12-15 could capture more complex spatial patterns while remaining within the parameter budget for edge deployment.

2) Training Enhancements: **Larger Datasets:** Expanding beyond 2013 data to include multiple years (2013-2023) would provide 10-100x more training examples, likely improving generalization and accuracy.

Data Augmentation: Board reflections and rotations could artificially expand the dataset. Position-aware augmentation that maintains chess semantics (e.g., mirroring along the vertical axis) could improve robustness.

Time-Conditioned Training: Including remaining time as an input feature would allow the model to capture how humans play differently under time pressure, distinguishing between bullet, blitz, and classical games.

Cross-Validation: Implementing k-fold cross-validation would provide more robust performance estimates. Unlike Maia, Rylee's smaller dataset makes cross-validation computationally feasible.

3) Feature Additions: **ELO Prediction Head:** Adding an auxiliary head to estimate player rating from move patterns would enable dynamic difficulty adjustment, allowing Rylee to adapt its play style to match the user's skill level in real-time.

Human vs. Bot Detection: A discriminator head could identify engine-assisted play by detecting superhuman move sequences, useful for online chess platforms combating cheating.

Blunder Detection: Integrating with Stockfish to identify major mistakes and generate natural language explanations would create an interactive tutoring system. Rylee could suggest the human-like move while explaining why it's better than the played move.

4) Evaluation Extensions: **User Studies:** Conducting controlled experiments where chess students practice against Rylee vs. traditional engines would quantitatively measure educational effectiveness and user engagement.

Turing Test for Chess: Having experienced players review game transcripts and classify whether moves came from humans or Rylee would directly measure human-likeness beyond simple accuracy metrics.

Behavioral Stylometry: Following McIlroy-Young et al.'s work on detecting individual playing styles, Rylee could be extended to mimic specific players or playing styles (aggressive, defensive, tactical, positional).

Position Difficulty Analysis: Analyzing accuracy across position types (tactical, positional, endgame) would identify where the model excels or struggles, guiding targeted improvements.

C. Concluding Remarks

This work demonstrates that human-aligned AI systems can be made practical and accessible without requiring data center infrastructure. The key insight is that perfect alignment is unnecessary—capturing enough human-like behavior to create an engaging, educational experience requires far fewer resources than previously thought.

Beyond chess, this research has broader implications for democratizing AI development and deployment. Many applications benefit more from human-aligned behavior than superhuman performance: tutoring systems in education, collaborative decision support in medicine, and AI assistants in creative domains all prioritize complementing human thinking over replacing it.

Most importantly, this project validates a key principle in AI development: that constraint-driven design can yield innovations with broad impact. The requirement to run on edge devices forced architectural decisions that not only achieved deployability but also resulted in faster training, easier debugging, and more interpretable behavior. As AI systems grow larger and more resource-intensive, approaches like Rylee's remind us that smaller, focused models trained on curated data can deliver substantial value for specific applications—and that accessibility and alignment may matter more than raw performance for many real-world use cases.

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. on Knowledge Discovery and Data Mining (KDD), 2020.

[2] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, pp. 484–489, 2016.

[3] 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.

[4] Z. Tang, D. Jiao, R. McIlroy-Young, J. Kleinberg, S. Sen, and A. Anderson, "Maia-2: A Unified Model for Human-AI Alignment in Chess," in Proc. 38th Conf. on Neural Information Processing Systems (NeurIPS), 2024.

[5] 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. on Knowledge Discovery and Data Mining (KDD), 2020, pp. 1667–1677.

[6] T. McGrath et al., "Acquiring human-like concepts from a superhuman AI," arXiv:2210.13432, 2022.

[7] R. McIlroy-Young, R. Wang, A. Anderson, and J. Kleinberg, "Detecting Individual Decision-Making Style: Exploring Behavioral Stylometry in Chess," in Proc. 35th Conf. on Neural Information Processing Systems (NeurIPS), 2021.

[8] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[9] N. Moskopp, "python-chess: a chess library for Python," GitHub repository, 2014. [Online]. Available: https://github.com/niklasf/python-chess

[10] Doshi-Velez, F., & Kim, B. (2017). A Roadmap for a Rigorous Science of Interpretability. ArXiv, abs/1702.08608.

[11] Stadie, B. C., Abbeel, P., & Sutskever, I. (2017). Third-person imitation learning. arXiv preprint arXiv:1703.01703.

[12] Charness, N. The impact of chess research on cognitive science. Psychol. Res 54, 4–9 (1992). https://doi.org/10.1007/BF01359217