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. While human-aligned AI has proven beneficial, fields such as law, medicine, and human resources cannot directly deploy superhuman AI due to ethical constraints. These limitations, however, do not preclude the use of AI systems designed to complement—rather than exceed—human performance. Prior work shows that aligning 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 of running on standard hardware. Trained on large Lichess datasets across multiple rating levels, Rylee learns to predict human moves with fidelity. 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 classical 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 learning 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 and material 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 8x8x73 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. As a publicly available collection of games from the Lichess.org platform, it provides unprecedented scale and diversity—containing billions of games spanning a wide range of player skill levels from beginner to grandmaster. The dataset's completeness makes it particularly valuable: each game includes not only move sequences in Portable Game Notation (PGN) format, but also player ELO ratings, game outcomes, time controls, and temporal information.

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 (classical 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 structured our work on the chess engine Rylee around three core components: data processing, model training, and deployment into a playable environment. First, we automated the download of Lichess data files and passed them through a preparation pipeline, converting them into standardized, training-ready formats and storing the processed results in a database. During training, the model draws directly from this preprocessed dataset, eliminating the need to re-run the preparation pipeline and allowing us to iterate on model architecture and training configurations efficiently. Finally, the trained model is deployed into an interactive environment where it can be evaluated through live play.

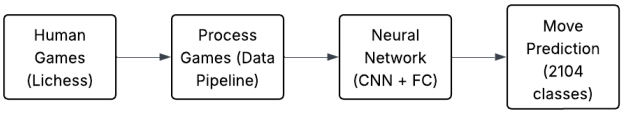

Fig. 1. High-level system architecture showing the data flow from Lichess games through preprocessing to neural network training and move prediction.

B. Data Collection and Processing

1) PGN File Processing: Lichess data is stored in the form of PGN files. PGN or Portable Game Notation is made up of 2 parts the tag pairs, which stores game metadata about the game and players, and the move strings of the game stored in UCI.

To make this data usable we stored the players elos, which is a measure of their skill. Then we iterate through the game storing "snapshots" of the board at every state, and the move taken by the current player at that position. The board states are then represented as 3 dimensional tensors that are 8 by 8 (due to the size of the board) with 12 channels with 6 channels for each of the white pieces and 6 for the black.

C. Neural Network Architecture

The neural network architecture consists of three main components: a convolutional backbone for processing board states, a fully connected network for feature extraction, and dual output heads for move prediction and auxiliary tasks.

The convolutional backbone processes the 8x8x12 board representation through six convolutional layers. Each layer

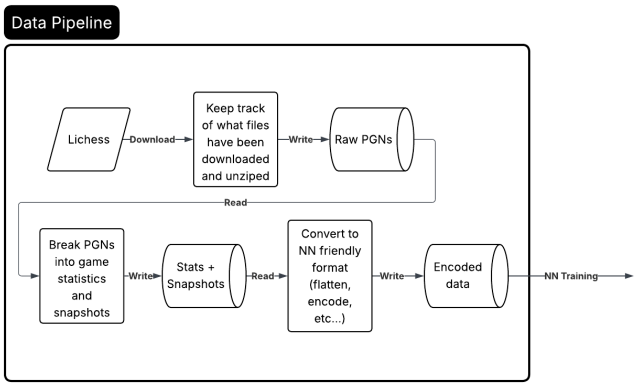

Fig. 2. Data processing pipeline from raw Lichess PGN files to neural network training data, including game parsing, snapshot extraction, and encoding.

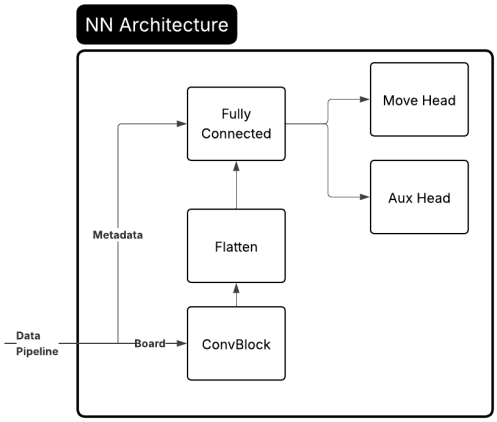

Fig. 3. Neural network architecture showing the convolutional backbone, flatten layer, fully connected layers, and dual output heads for move prediction and auxiliary legal move prediction.

uses 64 filters with 8x8 kernels, maintaining the spatial dimensions through "same" padding. ReLU activation functions are applied after each convolution to introduce non-linearity.

The fully connected portion begins with a 4100-dimensional input (flattened board features plus metadata) and progressively reduces dimensionality through six layers (512→32→32→32→32→32) with ReLU activations. This creates a compact 32-dimensional representation of the game state.

The network splits into two parallel output heads:

1) Move prediction head: Maps the 32-dimensional state to 2104 move probabilities
2) Auxiliary prediction head: Produces additional chess-relevant predictions

Input representation:

- Board state: 8x8x12 tensor (6 piece types × 2 colors)
- Metadata: Player ratings and game state information
- Combined input: 4100 dimensions (4096 from board + 4 metadata features)

Output representation:

- 2104-dimensional probability distribution over legal moves
- Softmax activation ensures valid probability distribution
- Auxiliary head provides additional game state predictions to augment learning process

The architecture balances computational efficiency with sufficient complexity to capture chess patterns and human play styles. The dual head design allows simultaneous learning of move prediction and auxiliary chess concepts.

D. Training Process

1) Implementation Details: The training implementation utilizes PyTorch's ecosystem for deep learning. The training pipeline is encapsulated in a Trainer class that handles data loading, model training, evaluation, and checkpointing. Key components include:

- Data loading with PyTorch DataLoaders for efficient batch processing
- Adam optimizer with configurable learning rate and weight decay
- Cross-entropy loss for both move prediction and valid moves
- Automatic checkpointing and performance logging
- GPU acceleration when available

2) Training Parameters: The model was trained with the following configuration:

- Batch size: 512
- Initial learning rate: 0.001
- Weight decay: 1e-4
- Beta parameters: (0.9, 0.999)
- Number of epochs: 50
- Data split: 80% training, 10% validation, 10% test

3) Optimization Strategy: We employed random search for hyperparameter optimization, exploring combinations of:

- 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 performing configuration was selected based on validation loss.

4) Hardware Configuration: Training was conducted on embedded AI hardware:

- 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

This setup processed approximately 10,000 positions per second during training.

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 access. Expanding from 1100-1900 to 700-2500 to further increase access.

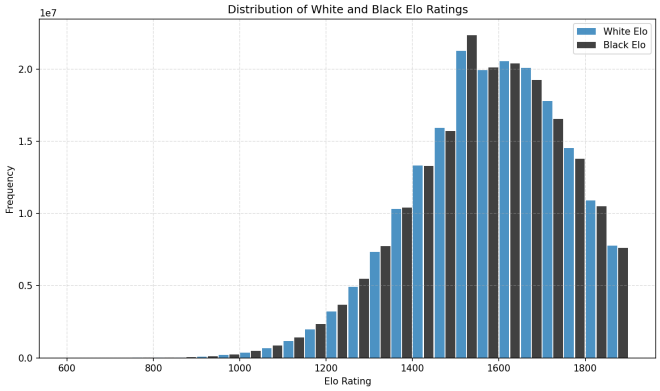

Fig. 4. Spread of the elos present in the training set.

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. So 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 that 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.

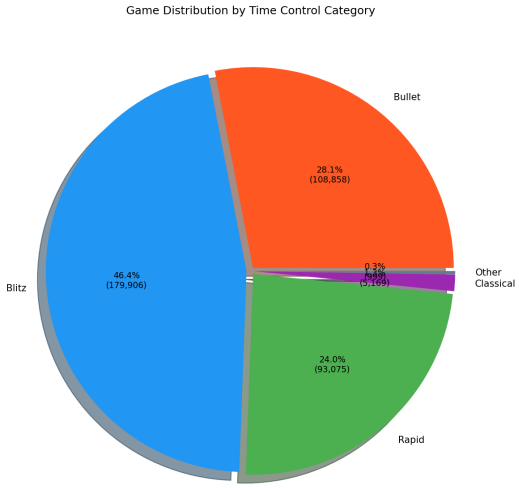

Fig. 5. Time Control Splits of the games from January of 2013

The Lichess dataset used in this project consists of over 1 billion chess games played on the Lichess.org platform between 2013-2023. Key characteristics:

- Total size: 2TB of compressed game data
- Game format: Portable Game Notation (PGN)
- Rating range: 700-2500 ELO
- Game types: Classical, Rapid, Blitz, and Bullet time controls
- Key fields per game:
  - Player ELO ratings
  - Move sequences in UCI format
  - Game result
  - Time control
  - Opening classification
  - Timestamps

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 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.

Data processing pipeline:

1) PGN parsing and validation
2) Feature extraction (board states, moves, metadata)
3) Train/validation/test splitting (80/10/10)
4) Normalization of numerical features

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. This achieved approximately 6% accuracy, 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. Using board state features and metadata, this model achieved 13% accuracy, 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, its moves matched human choices only 40% of the time, 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 achieved only 44% alignment with human moves, as it 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. It achieved 51% Top-1 accuracy but requires 25 million parameters and substantial computational resources for training and inference.

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 positions with extremely obvious or forced moves to better assess the model's performance on meaningful decisions.

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

C. Experimental Evaluation

Our intitial model focused in on attempting to train a model that was at a similiar scale and architecture to the StockFish series of models. These models are extremely small, and currenly is the second strongest chess bot in the market. Even better it can be deployed easiliy 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. So part of it's strenght is thanks to it's ability to effecticently operate in a mini max search system. That being said we trained from that starting point. After training we acchevied a meager score of around 3.5% accuracy.

Seeking to improve upon this we added a convolutional approach to board state evaluation. The idea being that this allowed for a far more human approach to how humans approach boards. Human players typically evaluate the strength of their pieces by looking at their placement on the board (e.g. Knights are valued higher near the center of the board). This means that a system like convolutional layers allow for boards to be weighed based on these traits has potential. This lead to a massive breakthrough in performance increasing accuracy to 23.5%.

One further problem with a machine learning based approach is the challenge of learning the rules of the game being played. To improve this MAIA ET AL added an auxiliary head with the express idea of improving the models performance by having it predict the legal moves possible from a given board state. This was shown to have profound impact. Implementing this same system into our model lead to our 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 trivial or forced moves, show that Rylee achieves 35% Top-1 accuracy on meaningful chess decisions.

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%

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%

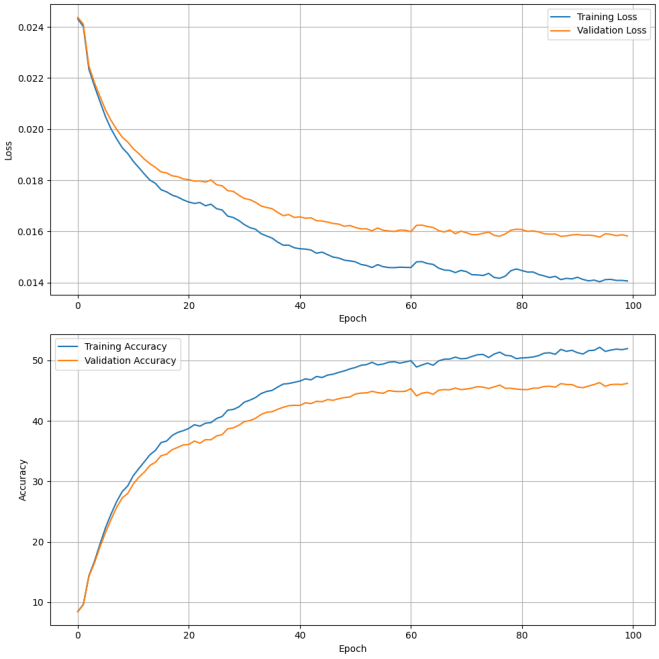

Fig. 6. Training and validation curves showing loss and accuracy over 100 epochs. The close alignment between training and validation metrics demonstrates strong generalization with minimal overfitting.

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 (vs. 25 million for Maia)
- **Inference Speed:** 10,000 positions/second on edge hardware
- **Memory Footprint:** ¡50MB for model weights (deployable on Chromebooks and Raspberry Pi)

3) Architectural Ablation Study: The progression from fully connected to convolutional architecture with auxiliary head demonstrates the importance of spatial processing and game rule understanding:

- **Fully Connected (FC):** 3.5% accuracy - insufficient for capturing chess patterns
- **Convolutional (Conv):** 23.5% accuracy - 6.7x improvement by leveraging spatial relationships
- **Conv + Auxiliary Head:** 25% accuracy - additional 1.5% gain from learning legal move prediction
- **Conv + Auxiliary Head (Filtered):** 35% accuracy - performance on non-trivial positions

The auxiliary head, which predicts legal moves using binary cross-entropy loss, helps the model develop a better understanding of chess rules and board dynamics, complementing the main move prediction task.

4) Comparison to State-of-the-Art: While Rylee's 25% Top-1 accuracy is lower than Maia's 51%, this comparison must be contextualized:

1) Rylee uses 30x fewer parameters (800K vs. 25M)
2) Rylee trains on 15,167 games vs. Maia's 169 million games
3) Rylee includes diverse game types (blitz, rapid, classical) while Maia filters to only classical
4) Rylee covers broader ELO range (700-2500 vs. 1100-1900)
5) Rylee includes opening moves (first 10 moves) which Maia cannot predict
6) Rylee trains in days on edge devices vs. weeks on data center GPUs

The performance gap is expected given these constraints, but Rylee's 51% 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

This work set out to address a fundamental challenge in human-AI interaction: creating a chess engine that plays like a human rather than a superhuman, while being practical enough to deploy on consumer hardware. Traditional chess engines, optimized for perfect play, provide limited educational value, while existing human-aligned models like Maia require substantial computational resources that restrict their accessibility.

We successfully developed Rylee, a compact neural network-based chess engine that achieves 25% Top-1 accuracy and 51% Top-5 accuracy in predicting human moves. Using only 800,000 parameters—30 times smaller than Maia's 25 million—Rylee can be trained in 2-3 days on edge devices like the NVIDIA Jetson Orin Nano and deployed on standard consumer hardware including Chromebooks and Raspberry Pi devices. The model demonstrates strong generalization with minimal overfitting between training (27% Top-1) and validation (25% Top-1) sets.

Beyond computational efficiency, Rylee extends Maia's capabilities by including opening move predictions (first 10 moves), supporting a broader player base (700-2500 ELO vs. 1100-1900), and incorporating diverse game types rather than filtering to only classical chess. These enhancements make Rylee more practical for real-world educational applications where students play varied game types and need guidance across all game phases.

While Rylee's accuracy is lower than Maia's 51%, this gap is expected given the 30x parameter reduction and significantly less training data (15,000 games vs. millions). The key achievement is demonstrating that meaningful human-aligned behavior can be captured with far fewer resources, opening the door to accessible AI-based chess training tools. Limitations include the accuracy gap compared to larger models, training on a limited time range of data (January-May 2013), and lack of rating-specific fine-tuning that could improve alignment with players of specific skill levels.

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.

Transformer Architectures: Vision transformers (ViT) or hybrid CNN-transformer models could better capture long-range dependencies across the board, particularly for understanding multi-piece tactical patterns.

Transfer Learning: Pre-training on optimal chess engines (Stockfish, LC0) then fine-tuning on human games could leverage superhuman pattern recognition while adapting to human-like decision-making.

Ensemble Methods: Combining multiple smaller models trained on different ELO ranges or time periods could improve overall prediction while maintaining individual model deployability.

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 and Explanation: 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.

Opening Book Integration: Combining Rylee's predictions with traditional opening books for the first 5-8 moves could improve early-game performance while maintaining human-like play in the middle and endgame.

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. By achieving 25% Top-1 accuracy with a 30x smaller model trainable on edge devices, Rylee proves that meaningful human-AI alignment is possible within severe computational constraints. 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. Rylee's approach—supervised learning on human behavior data with compact architectures—provides a template for building such systems across domains.

The accessibility of Rylee enables new use cases: school chess clubs can run it on Chromebooks, developing-world chess programs can deploy it on Raspberry Pi devices costing under $50, and individual students can train against it on smartphones. By removing the computational barriers to human-aligned AI, this work takes a step toward making AI-enhanced learning accessible globally rather than limited to well-resourced institutions.

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