Can Language Models Reason Through Chess?

Lucas Dionisopoulos*
University of California, San Diego
ldionisopoulos@ucsd.edu

Nicklas Majamaki*
University of California, San Diego
nmajamaki@ucsd.edu

Luke Mitbo*
University of California, San Diego
lmitbo@ucsd.edu

Manas Mulpuri*
University of California, San Diego
mmulpuri@ucsd.edu

Sree Nadella*
University of California, San Diego
srnadella@ucsd.edu

Abstract

The recent advancement of reasoning capabilities in language models has led to the saturation of previously difficult evaluation tasks for frontier models. The release of DeepSeek-R1 publicly demonstrated that verifiable domains such as code or math naturally lend themselves toward eliciting strong reasoning capabilities. In this paper, we attempt to extend reasoning to a novel verifiable domain: Chess. We successfully test the efficacy of various board representations and attempt to train a 1.5B and 7B parameter model through reinforcement learning to improve in chess via reasoning. Following unsuccessful results from reinforcement learning, we diagnose and identify weaknesses in our base models potentially serving as a bottleneck to reasoning ability. Finally, we evaluate the capability of frontier language models on chess to discover this domain remains difficult for even the most capable models; we conclude that this remains a promising, interesting, and plausible domain to further pursue.

1 Introduction

Chess has been a recurring grand challenge in the field of artificial intelligence, with notable milestones such as IBM Deep Blue forcing reigning world champion Gary Kasparov to resign in 1996 and the release of AlphaZero in 2017 Silver et al. [2017] – a neural architecture that achieved super-human chess performance purely through self-play.

The importance of chess as a target can be attributable to the difficult nature of the game coupled with clear, definable objectives; a hard, interesting problem well-suited for techniques developed in artificial intelligence. Additionally, new techniques for 'solving' chess remain interesting areas of study. Despite superhuman capability from chess engines, AlphaZero learned a unique style of play different from the flavor of leading chess engines – allowing AlphaZero to push human ability to new heights and evolve the frontier of the game. During an interview following his victory in a 2019 chess tournament in Norway, world champion Magnus Carlsen attributed his evolution as a player to AlphaZero:

> "In essence, I have become a very different player in terms of style than I was a bit earlier, and it has been a great ride." *Magnus Carlsen*

*Authors listed in alphabetical order by last name.

For these reasons, chess has been and will continue to be an interesting domain of study in the development of artificial intelligence. New techniques that lend themselves to achieving superhuman performance may teach humans to further evolve a game that has been studied for thousands of years.

The latest advancements in language models and reasoning models present an interesting new direction to approach the study of chess from. Models that achieve superhuman performance in coding OpenAI [2024] have only cracked the surface of chess with the capability of beating a player making random moves Saplin [2024]. However, the application of reasoning models to chess – a verifiable domain with clear rewards – presents a promising approach to this recurring grand challenge. Their success could offer a new lens to study the game of chess through, potentially opening the black box and granting us access to the reasoning behind the madness.

2 Related Work

Traditional chess engines like Stockfish operate using a variation of minimax search that involves alpha-beta pruning of trajectories unreachable by players choosing to play optimally Maharaj et al. [2022]. Per the Computer Chess 40/15 Rating List on March 14, 2025, Stockfish is currently the best performing chess engine in the world with an ELO of 3,642 CCRL Team [2025].

AlphaZero takes a different approach, utilizing a bespoke neural architecture leveraging a stack of grids representing the board state processed using both convolutional and linear layers Silver et al. [2017]. At time of publication, AlphaZero was tested against the 2016 TCEC world-champion version of Stockfish, winning or drawing every game played. MuZero further generalized the AlphaZero architecture to remove requiring game rules and was able to match AlphaZero in ELO by the end of training Schrittwieser et al. [2020].

Recent work from researchers at Google DeepMind tested the ability of a decoder-only transformer architecture to learn to predict high-quality moves provided by top chess engine play. This transformer was able to reach an ELO of 2,299 – validating the ability for large neural networks to amortize the task of search within latent layers Ruoss et al. [2024].

These represent the state of the art of different techniques to chess: Chess engine, RL-guided search, and neural-amortized search using transformers. However, a new technique has risen in the last year that represents a novel way to scale performance of pretrained language models – reasoning – notably popularized by OpenAI's release of o1 OpenAI [2024] and DeepSeek's release of R1 DeepSeek-AI et al. [2025]. Additionally, Kimi released Kimi K1.5 further contributing to public understanding of reasoning models that was notably advanced through R1 KimiTeam et al. [2025].

The releases from DeepSeek and Kimi unveiled the process of training a state of the art reasoning model. Given a strong base model with capabilities in a verifiable domain (e.g., code, math), a simple application of reinforcement learning via algorithms such as PPO Schulman et al. [2017] or GRPO Shao et al. [2024] allows the underlying language model to learn long context reasoning on harder tasks. Included in this is the ability to self-correct and backtrack, as well as stronger reasoning patterns. Results from these techniques show an ability for these models to generalize reasoning to new domains, with Anthropic showing improved performance of reasoning models on novel domains such as Pokemon Anthropic Team [2025].

3 Dataset

Training an LLM to provide consistent high-quality moves in any setting requires a dataset that comprises of a variety of board configurations, ranging from early game to late game. For this reason, we selected to train on a subset of the Google DeepMind chess dataset Ruoss et al. [2024]. This dataset sources from random human-played games on lichess, a popular chess website, in order to construct a diverse collection of board configurations. Specifically, the boards are given in Forsyth–Edwards Notation (FEN), and each board comes paired with a list of legal moves and associated win-probabilities of each move. These win probabilities are derived from Stockfish evaluations of the board that results from playing each of the legal moves. The dataset contained far more boards than was required for GRPO training, so we selected a random subset of 5000 entries for training and 2000 entries for validation.

(a) Original Chess Board

```
Black to move
r . b . . . . .
. p p r . p k p
p . n . p n p .
. . . . . . . .
. . P . . . . .
. . N . . N . .
P P B . . P P P
R . . . R . K .
```

(b) Grid Representation

```
Black to move.
Black Bishop on c8.
Black King on g7.
Black Knights on c6, f6.
Black Pawns on b7, c7, f7, h7, a6, e6, g6.
Black Rooks on a8, d7.
White Bishop on c2.
White King on g1.
White Knights on c3, f3.
White Pawns on c4, a2, b2, f2, g2, h2.
White Rooks on a1, e1.
```

(c) Desc. Representation

```
r1b5/1ppr1pkp/p1n1pnp1/8/2P5/2N2N2/PPB2PPP/R3R1K1 b - - 1 17
```

(d) FEN Representation

Figure 1: Visualization of Chess Representations

4 Approach

4.1 Prompting

Many works such as Xu et al. [2025] have demonstrated that including information in the prompt telling an LLM that it is an expert in the domain pertaining to the problem increased the overall performance on that problem. We adopt this approach, contextualizing the LLM as a smart and strategic player in a chess tournament. Due to the wide array of chess notation possible in the model's training dataset, we provide a one-shot example to aid in the model's ability to provide moves in the format our reward model expects. The full system prompts used, including the one-shot example, can be found in the Appendix, but all system prompts followed the same basic format of:

```
You are a smart, strategic, and wise chess reasoning model
currently in a chess tournament.
Given a chess board and a list of legal moves you...

<One-shot example>

Make sure that your chosen move is provided in standard chess
bot notation
(e.g., g8f7, c2e1, g3g4) which means move the piece on the
first square to the second square
(so g8f7 means move the piece on g8 to f7).
Please use English for your thought process and make sure
your thinking isn't too long, and if your final answer is not
enclosed in <answer> </answer> tags you will lose.
```

In testing we found that giving a brief explanation of the formatting expected for the moves in the system prompt aided in the model's ability to provide legal moves.

Given our limited resources and the model's tendency to occasionally generate unreasonably long reasoning traces, we induced a token generation limit during training. In order to help the model spend more tokens reasoning through the strength of the moves rather than determining which move is legal, we made the decision to provide a list of legal moves after the board representation. We later discuss how this contributed towards reward hacking, so we later removed the list of legal moves from the prompt in further training runs. Each training data point used the same system prompt, with the user prompt containing only the board from the dataset and, depending on the training run, a list of the legal moves available for the board.

Category	Number of Legal Moves	Number of Illegal Moves	Extraction Errors
FEN	36	29	33
Desc.	32	55	11
Grid	22	63	14

Table 1: Board Configuration Experimental Results

4.2 Board Respresentation

Again due to our limited computational budget, we designed and executed experiments in order to determine the most promising board format to provide the model. The primary concern with boards in FEN was that the tokenization of the LLM would often group adjacent characters, which could have an adverse impact on the model's ability to comprehend the board state properly (see Appendix E). We tested FEN, a text grid representation, and an English descriptive representation. Each representation can be seen in Figure 1.

All experiments to determine viability of board representations were tested on a DeepSeek-R1-Distill-Qwen-1.5B model for 100 samples each, used the same system prompt, and contained a list of legal moves in UCI notation. Samples where the model exceeded 100 seconds to provide an answer were rare and discarded from the results, and in our experiments an *Extraction Error* is defined to be an instance where the model failed to produce a move-like string between $< answer >< /answer >$ tags.

The results from these experiments in Table 1 allowed us to rule out the grid representation, as it was outperformed by other representations in every metric. However, while the results seemed to indicate that the FEN representation was the most comprehendible representation for the LLM, some subjective analysis of the reasoning traces indicated that the model generally understood the dynamics of the board better in the Desc. representation. For example, below is a sample from a reasoning trace where the model was given the board in FEN and eventually suggested the best-available move:

> *First, let's analyze each move: 1. **g3g2**: This means moving a pawn from g4 to g3 and then capturing the rook on e6 or d5 with its own pawn…*

Upon inspection, it is clear that this reasoning is invalid and inconsistent with the rules of chess. Meanwhile, below is a sample from a reasoning trace where the model was given a board in Desc. format yet failed to produce valid answer tags:

> *3. The bishop from e7 to d8 or f6: This seems strategic as it can potentially capture White's pawns…*

Here, the squares listed are valid squares the bishop can move to and the reasoning is also generally valid.

While we acknowledge that these are only single examples and that generating concrete experiments to quantify this behavior is difficult, inspection of the reasoning traces warranted further experimentation in training on both FEN and Desc. board representations. Additionally, even with specifying the desired format for representing a move, we found that on occasion the model would provide its move in PGN notatation instead of UCI. For this reason, some training runs used PGN for the moves as opposed to UCI in order to lean into this tendency. All training runs where the list of legal moves was excluded from the prompt used PGN for the moves.

4.3 Reward Modeling

Given that the dataset provides a diverse set of board configurations, the win probability associated with the best move for each board ranged from 0 to 1. For this reason, the rewards could not simply use the win probabilities, for they must properly consider the *relative* win probabilities of each legal move. We identified three possible reward functions to consider this context.

First, we can simply normalize all of the win probabilities into a $[0, 1]$ range by applying the following formula to each move:

$$R(move_i) = \frac{w_i - w_{min}}{w_{max} - w_{min}}$$

Here, w_{min} is the minimum win probability of all legal moves for the given board, and w_{max} is the maximum win probability of all legal moves for the given board.

The second configuration takes the z-score of all of the win probabilities and then normalize these z-scores into a $[0, 1]$ range by applying a similar normalization formula. For the sake of completeness, the formula for this reward model is given by:

$$R(move_i) = \frac{\text{zscore}(w_i) - \text{zscore}(w_{min})}{\text{zscore}(w_{max}) - \text{zscore}(w_{min})}$$

Finally, the third reward configuration takes the z-score of each of the win probabilities and normalize to fit in the same $[0, 1]$ range, followed by setting all values less than a predetermined threshold to 0. This configuration has the added benefit of only giving rewards for "strong" moves should the threshold hyperparameter be set appropriately. The larger the threshold, the stronger the move must be in order to receive a reward. This comes with the added cost of more sparse rewards, and after experimentation we found a threshold of 0.4 to be a reasonable middle-ground of extracting strong moves yet still giving rewards to a not insignificant subset of the legal moves.

It should be noted that a select few entries of the dataset were such that all legal moves had a win probability of 0. Although these entries were sparse, in order to limit the model seeing strong rewards for irrelevant moves we somewhat arbitrarily assigned a reward of 0.5 for all legal moves in these cases.

In some training runs, we also introduced a reward of 0.1 for the model providing $< answer >< /answer >$ tags, and a reward of -0.1 for not including such tags. This was done primarily in an attempt to alleviate the issue of sparse rewards in the absence of having a list of legal moves provided in the prompt.

The distribution of rewards over the dataset for the first and second configurations were nearly identical, so unless otherwise specified the second reward configuration was the default setting we chose.

4.4 Reasoning Loop via GRPO

Given the strong baseline ability of the DeepSeek-Distilled family of reasoning models DeepSeek-AI et al. [2025] – notably the 1.5B and 7B variants – coupled with distillation on a proven set of high-quality reasoning data, we decided to apply the GRPO-based DeepSeek-R1 reasoning loop using our previously defined rewards.

$$\begin{aligned}
\mathcal{J}_{GRPO}(\theta) &= \mathbb{E}\big[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)\big] \\
&\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta D_{KL}\big(\pi_\theta||\pi_{ref}\big)\right), \\
D_{KL}\big(\pi_\theta||\pi_{ref}\big) &= \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1.
\end{aligned}$$

In the experiments, we followed the hardware-efficient parameter choice of $G = 1$ as was shown to be effective in the TinyZero implementation Pan et al. [2025]. The reference policy, $\pi_{\theta_{old}}$ is the initial model checkpoint downloaded from the Hugging Face Hub. Given a choice of $G = 1$, the advantage A_i is simply our reward metric mentioned previously. We also set the KL-loss coefficient $\beta = 0.001$. The clipping variable was set to $\epsilon = 0.2$. We used default optimization settings for the training engine we leveraged. All experiments used 8 as a default batch size except for the batch size stability training experiment.

4.5 Training Stack

We conducted full fine-tune experiments on DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B, leveraging Volcano Engine Reinforcement Learning ("verl"). verl is an open-source version of HybridFlow Sheng et al. [2024].

For hardware we utilized rented GPUs from a combination of providers including Lambda Labs, RunPod, and Vast ai. For the 1.5B experiments we utilized several different types of nodes: 2xA6000, 2xL40, 2xA10, and 1xA100. For the 7B experiments we utilized a node of 2xH100. Each experiment hereafter discussed ran in less than 3 hours (wall-clock speed).

5 Results

5.1 1.5B Model

We conducted several experiments on the 1.5B model, with new experiments guided by shortcomings of prior tests. Notably, we conducted the following 3 experiments – listing in chronological order:

- **1.5B Initial Prompt**: 2-shot, FEN board with legal moves using normalized reward.
- **1.5B Clipped Reward**: 1-shot, descriptive board with legal moves using clipped z-score reward.
- **1.5B No Legal Moves**: 1-shot, descriptive board without legal moves using regular z-score reward. Additionally, for this we asked the model to return its desired move in PGN notation as this format is likely most represented in pretraining data.

For the `1.5B Clipped Reward` and `1.5B No Legal Moves` experiments, we also included a 0.1 score penalty if the model failed to generate answer tags and a 0.1 positive reward if the model provided a move in the desired format (not required to be legal).

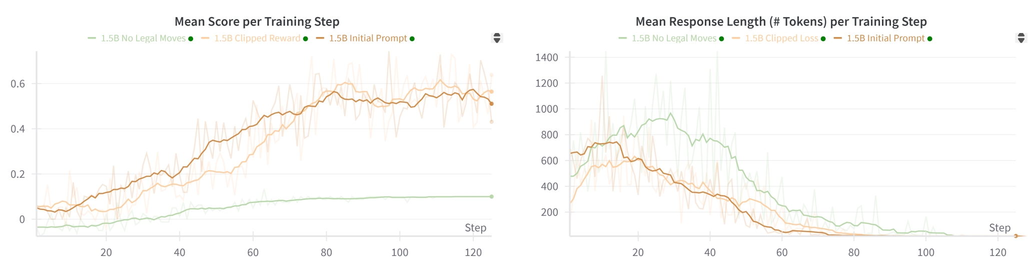

Figure 2: Results from various tests on DeepSeek-R1-Distill-Qwen-1.5B. Solid line is computed as running average over the prior 10 training steps. Results display clear sign of reward hacking as the model learns to receive trivial formatting rewards and fails to reason through the board state.

Figure 2 displays the results of our various 1.5B tests. These results clearly portray reward hacking as the model learns to trivially maximize return by following the desired formatting and does not learn to reason over the board state. For the `1.5B Initial Prompt` test, the model learns to predict a single move from the set of legal moves – this provides an expected value of approximately 0.5 which reflects the asymptote of the graph. For the `1.5B Clipped Reward` test, the model similarly optimized to choosing a random move, which we hoped using the clipped z-score would prevent. For the final `1.5B No Legal Moves` test, the model learned to provide a move in the desired PGN format to receive the guaranteed 0.1 reward. Appendix A further analyzes the reward hacking of the 1.5B model.

5.2 7B Model

Given results training on the 1.5B model, we decided to scale our experiment and test performance on the 7B model. These experiments required much more expensive hardware, so we chose to only test the `No Legal Moves` experiment. This utilized the same setup and reward as the 1.5B model.

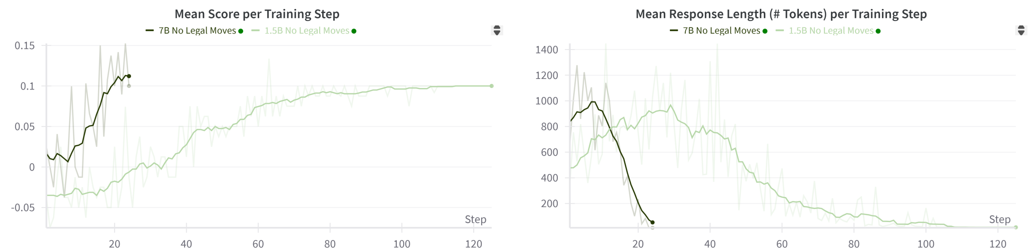

Figure 3: Results on scaling up to the 7B model. Solid line is computed as running average over the prior 10 training steps. Results show the 7B model struggles with the same issues observed with the 1.5B model.

Unfortunately, Figure 3 displays the same issues in the 7B model as we observed with the 1.5B model. However, the 7B model was able to learn this faster than the 1.5B model and was slightly better at predicting some legal moves. This can be seen by the spikes above 0.1 that the 7B model achieved – these spikes required choosing a legal move in the correct format.

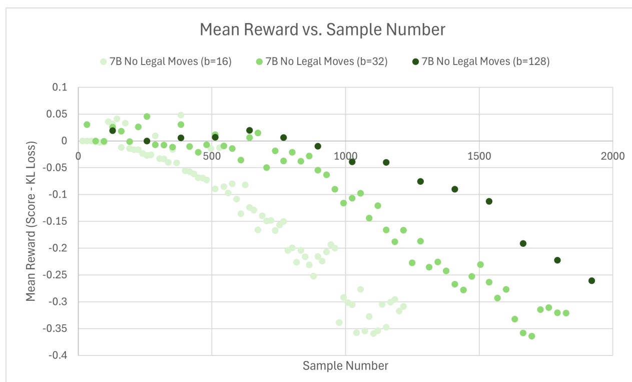

Figure 4: Results on experimenting with batch size and training stability. The larger batch sizes displayed more stable training as was predicted.

Given the 7B model was slightly more effective than the 1.5B model in these tests, we decided to run one final experiment to test the impact of varying batch sizes with training stability. Figure 4 displays the results of this experiment – the 7B model displayed more stable training and held a positive reward for more training steps compared to the smaller batch sizes. We hoped that given the model's poor performance resulting in sparse rewards, larger batch sizes may allow the model to learn to reason by avoiding local optima presented by the suboptimal reward hacked rewards. However, this was not enough to allow the model to learn to reason properly.

6 Post-Mortem Analysis

6.1 Initial Qualitative Analysis

To better understand our results, we took a step back to consider the possible explanations for the negative results. From a reward perspective, chess provides a verifiable domain assuming we define Stockfish win probability as ground truth; given Stockfish displays superhuman performance in chess, we believe this is a valid assumption not limiting our experiment.

Qualitative analysis of various responses from the base models displayed that the underlying cause is likely tied to the ability of the base models – the base models were prone to hallucinating moves and often failed to understand the full board positioning. Specific cases of model hallucinations are covered in Appendix B.

6.2 Ability of Our Base Models in Chess

Given the results of our initial qualitative analysis, we conducted an experiment to assess the capability of our base models in predicting legal moves in chess. This experiment was designed to evaluate the models' fundamental understanding of chess rules and board representations.

Our methodology involved selecting a random piece from a chessboard in the test set and prompting various base models to predict its available legal moves. We used the descriptive board format, as preliminary tests showed it to be more comprehensible to the models. The evaluation loop was designed to test multiple model variants (DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Llama-8B) with this format, expecting move predictions in Algebraic Notation (AN).

The results of our experiment are summarized in the following table:

Model	% Legal Moves	% Illegal Moves
DeepSeek-R1-Distill-Qwen-1.5B	17%	83%
DeepSeek-R1-Distill-Qwen-7B	38%	46%
Deepseek-R1-Distill-Llama-8b	54%	61%

Table 2: Performance of base models in predicting legal chess moves

The results in Table 2 indicate a clear trend of improvement with model size, yet even the largest model struggles significantly with legal move prediction. Notably, the percentage of illegal moves remains high across all models, suggesting persistent difficulties in understanding chess rules.

Our findings revealed that all the baseline models we tested demonstrated a significant lack of capability in predicting legal moves. This deficiency was observed across all model sizes, indicating that the issue is not solely related to model capacity. The primary reasons for these failures appear to be:

- **Hallucination**: Models occasionally forgot the given board representation midway through processing, leading to moves that were inconsistent with the board state or chess rules.
- **Board representation errors**: Models sometimes misinterpreted the initial board setup, leading to incorrect understanding of piece positions and relationships, even before move generation.

These results suggest that our base models, despite their general language understanding capabilities, lack the specialized knowledge required for chess gameplay. This experiment highlights the need for targeted training or fine-tuning approaches to improve base model performance in chess-related tasks.

6.3 Evaluation of Frontier Models

We also evaluated the chess-playing capabilities of state-of-the-art large language models by playing them against Stockfish. The Stockfish engine was configured with an Elo rating of 1350 with a move time of 10 milliseconds per turn. Each model played a full game as black against Stockfish, with the board state provided to the model in the descriptive format. The performance of the models was assessed using the centipawn evaluation score, which quantifies positional advantage or disadvantage relative to the model after it makes a move. For example, a score of +100 centipawns relative to black indicates that black is ahead by the equivalent of one pawn.

The following models were used in our evaluation:

- **OpenAI GPT Series**: 4o-mini, 4o, o3-mini, o1
- **DeepSeek**: DeepSeek-R1
- **Anthropic Claude Series**: Claude 3.5 Haiku, Claude 3.7 Sonnet, Claude 3.7 Sonnet with Extended Thinking.

Figure 5 shows how the frontier models performed against Stockfish. The graph is noisy, because each plot represents a single game between Stockfish and the LLM.

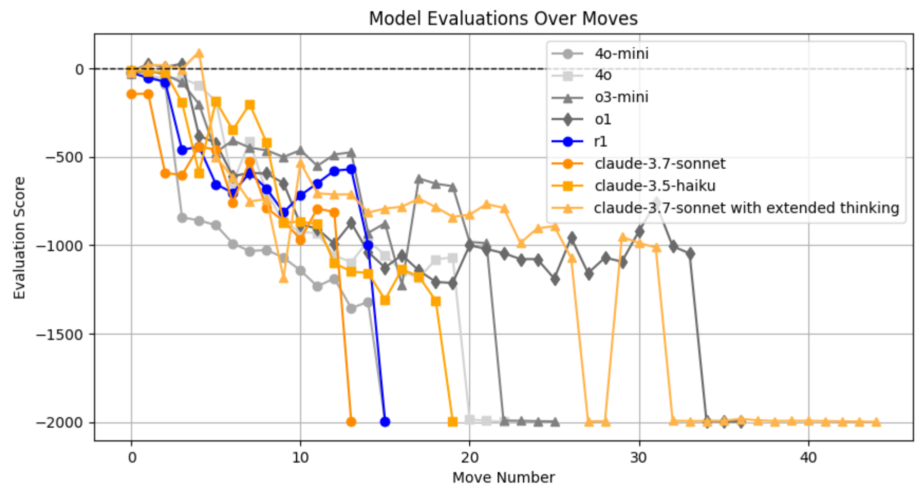

Figure 5: Performance against Stockfish (ELO of 1350 with 10ms move time).

We observed that all models exhibited significant weaknesses in playing chess and often blunder within the first few moves. However, models with explicit reasoning capabilities, such as o1 and Claude 3.7 Sonnet with Extended Thinking, were able to prolong their play even though they made critical mistakes early on in their games.

A recurring pattern observed across multiple models was an incomplete understanding of the position of all pieces on the chessboard. Often, models would make moves that would be effective in a standard board (e.g., develop your bishop to an open part of the board) – but the models fail to fully contextualize the opposing pieces and blunder their piece. An example is provided in Appendix C of a blunder made by o1.

This suggests that while frontier models grasp the general strategic principles of chess, they struggle to integrate these strategies with concrete positional evaluation.

7 Conclusion

Chess remains a difficult domain for language models despite being a popular game with strong representation online. We believe several factors are at play causing the poor results from language models:

- **Board Representation**: Chess boards are 2-D which makes them difficult for textual representation. Current formats such as FEN may lack efficacy due to tokenization issues, and other common formats such as PGN notation do not capture the full state of the board. As a result, chess data observed in pretraining may not be effectively compressed into latent space.
- **Difficulty**: Effective chess playing requires search and extrapolation of several move trajectories; while the results from Ruoss et al. [2024] showed the ability of a transformer to reach grandmaster-level chess play, eliciting this level of performance from a general language model requires breakthroughs in training reasoning models.
- **Grid-Based Problems**: Language models struggle in other grid-based domains such as the ARC Prize Chollet [2019]. One argument for the poor performance of autoregressive language models on this task is the large grid-based format not being well-suited for these architectures.

Despite the negative results from this study, we believe that this remains an interesting and plausible domain to further study. Despite the limitations of our base models, the 7B model did display superior performance compared to the 1.5B model and generated non-trivial rewards more often.

However, we believe what is required to bootstrap reasoning in chess is a stronger base model. This likely requires tailored fine-tuning on a chess-reasoning textual dataset. Through this, we believe that the increase in reward density may allow for a successful application of the reasoning feedback loop.

Understanding the capabilities of reasoning models in the domain of chess will unfold in due time. If history repeats itself, these breakthroughs will only elevate human performance in this storied domain, using artificial intelligence to augment and advance our own abilities.

References

Anthropic Team. Claude's extended thinking, February 2025. URL `https://www.anthropic.com/news/visible-extended-thinking?ref=nerdschalk.com`.

CCRL Team. Computer chess 40/15 ratings list, March 2025. URL `https://computerchess.org.uk/ccrl/4040/rating_list_all.html`. Accessed: March 15, 2025.

François Chollet. On the measure of inteligence. *ArXiv*, 2019. URL `https://arxiv.org/abs/1911.01547`.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

KimiTeam, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL `https://arxiv.org/abs/2501.12599`.

Shiva Maharaj, Nick Polson, and Alex Turk. Chess ai: Competing paradigms for machine intelligence. *Entropy*, 24(4):550, April 2022. ISSN 1099-4300. doi: 10.3390/e24040550. URL `http://dx.doi.org/10.3390/e24040550`.

OpenAI. Learning to reason with llms, September 2024. URL `https://openai.com/index/learning-to-reason-with-llms/`. Accessed: March 13, 2025.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, Cannada A. Lewis, Joel Veness, and Tim Genewein. Amortized planning with large-scale transformers: A case study on chess, 2024. URL `https://arxiv.org/abs/2402.04494`.

Maxim Saplin. Deepseek r1 vs openai o1, January 2024. URL `https://dev.to/maximsaplin/deepseek-r1-vs-openai-o1-1ijm`. Posted on Dev.to, edited on Jan 29.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL `http://dx.doi.org/10.1038/s41586-020-03051-4`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL `https://arxiv.org/abs/1712.01815`.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. Expertprompting: Instructing large language models to be distinguished experts, 2025. URL `https://arxiv.org/abs/2305.14688`.

Appendix

A Additional Training Results: 1.5B

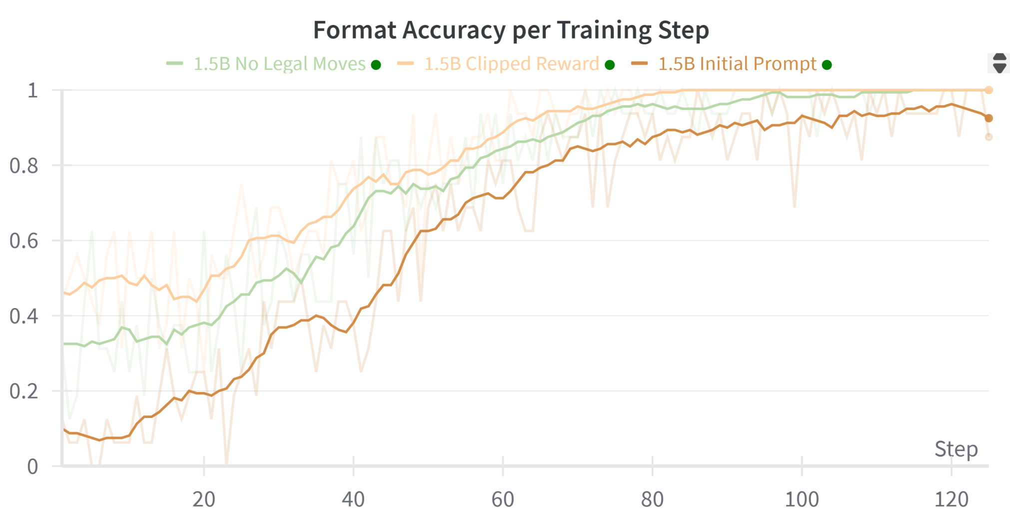

Figure 6: Format accuracy for the various 1.5B model tests.

Figure 6 highlights the reward hacking observed in the 1.5B experiments. Format accuracy is defined differently for each test but similarly involves the model providing valid answer tags (<answer> followed by </answer>) and, for the tests providing legal moves, providing a legal move. Combined with the expected values of random guesses explains the asymptotes of the mean score chart in Figure 2.

B Model Hallucinations

```
Model: deepseek-r1:7b

```Prompt:
<board>
Black to move.
Black King on e6.
Black Pawns on b7, c7, g7, b6, h6.
Black Rook on e1.
...
</board>
<piece> p </piece> <position> b7 </position>
...

Response:
...
Black has their King on e6 and pawns on b7, c7, g7, which are all on the seventh rank. The
White pieces include a Rook on e1, another on e2, Kings on f3, and White pawns on c3, h3, a2,
b2, g2.
...
<answer> b6 </answer>
```

Figure 7: Example from the 7B model displaying inaccuracy to fully contextualize the board state.

Figure 7 highlights the inability of the 7B model to correctly understand that it already had a black pawn on `b6`, making its proposed move illegal.

C Model Blunders

Figure 8: Example Blunder by o1

Figure 8, shows an example blunder made by state-of-the-art LLMs. Here o1 decides to move its bishop to c5, however it doesn't recognise that the bishop can easily be captured by White's pawn on d4. Here is o1's response while it made the blunder:

```
<think> Black's b7 pawn is eyed by White's queen on b3, but simply
defending the pawn passively may yield White an edge in development.
A more dynamic continuation is to develop the dark-squared bishop
to c5.  This pressures White's center (particularly d4) and sets
```
```

up tactical ideas if White becomes greedy and takes on b7 with the queen, for instance by responding with ...Nxd4. So 1...Bf8c5 is a solid developing move that addresses White's threats and prepares Black's own play in the center. </think> <answer>f8c5</answer>

D Full System Prompt

Legal moves provided, FEN prompt

You are a smart, strategic, and wise chess reasoning model currently in a chess tournament.

Given a chess board in Forsyth-Edwards Notation (FEN) and a list of legal moves, you think through the various moves you can make and reason about which move is the best. Then you provide your final move back to the user based on your reasoning analysis.

The reasoning process and answer must be enclosed within <think> </think> and <answer> </answer> tags, respectively. For example, when given an input prefixed with "user:", your response should be in the format "assistant: <think> [your reasoning] </think> <answer> [chosen move] </answer>".

Below is an example of your desired behavior:

Example 1: user: <FEN> 7R/4n1k1/4P3/1pp2B2/8/6P1/2r4P/6K1 w - - 3 50 </FEN> <legal moves> [f5h7, h8h6, h2h4, h8g8, h2h3, g1f1, f5d3, f5h3, h8b8, h8h4, h8c8, h8f8, h8a8, h8d8, f5g4, h8h3, g3g4, g1h1, f5e4, h8h5, f5c2, h8e8, f5g6, h8h7] </legal moves> assistant: <think> Playing as white, I'm in the offensive here. My rook is currently in at risk of being taken by their king and my bishop is at risk of being taken by their knight. I could take their rook with their bishop but they would take my rook. However, if I move my rook to h7, I'll put their king in check while saving my rook and bishop and continue pressure. Moving rook h8 to h7 is a wise move. </think> <answer> h8h7 </answer>

Make sure that your chosen move is provided in standard chess bot notation (e.g., g8f7, c2e1, g3g4) which means move the piece on the first square to the second square (so g8f7 means move the piece on g8 to f7). Please use English for your thought process and make sure your thinking isn't too long, and if your final answer is not enclosed in <answer> </answer> tags you will lose.

Legal moves provided, Desc. prompt

You are a smart, strategic, and wise chess reasoning model currently in a chess tournament.

Given a chess board and a list of legal moves, you think through the various moves you can make and reason about which move is the best. Then you provide your final move back to the user based on your reasoning analysis.

The reasoning process and answer must be enclosed within <think> </think> and <answer> </answer> tags, respectively. For example, when given an input prefixed with "user:", your response should be in the format "assistant: <think> [your reasoning] </think> <answer> [chosen move] </answer>".

Below is an example of your desired behavior:

Example 1:
user: <board> White to move.
Black King on g7
Black Knight on e7
Black Pawns on b5, c5
Black Rook on c2
White Bishop on f5
White King on g1
White Pawns on e6, g3, h2

```
White Rook on h8 </board>
<legal moves> [f5h7, h8h6, h2h4, h8g8, h2h3, g1f1, f5d3, f5h3, h8b8,
h8h4, h8c8, h8f8, h8a8, h8d8, f5g4, h8h3, g3g4, g1h1, f5e4, h8h5,
f5c2, h8e8, f5g6, h8h7] </legal moves>
assistant:  <think> Playing as white, I'm in the offensive here.
My rook is currently in at risk of being taken by their king and
my bishop is at risk of being taken by their knight.  I could take
their rook with their bishop but they would take my rook.  However,
if I move my rook to h7, I'll put their king in check while saving
my rook and bishop and continue pressure.  Moving rook h8 to h7 is a
wise move.  </think> <answer> h8h7 </answer>
Make sure that your chosen move is provided in standard chess bot
notation (e.g., g8f7, c2e1, g3g4) which means move the piece on the
first square to the second square (so g8f7 means move the piece on
g8 to f7).  Please use English for your thought process and make
sure your thinking isn't too long, and if your final answer is not
enclosed in <answer> </answer> tags you will lose.
```

No legal moves provided, Desc. prompt

```
You are a smart, strategic, and wise chess reasoning model currently
in a chess tournament.
Given a chess board, you think through the various moves you can
make and reason about which move is the best.  Then you provide your
final move back to the user based on your reasoning analysis.
The reasoning process and answer must be enclosed within <think>
</think> and <answer> </answer> tags, respectively.  For example,
when given an input prefixed with "user:", your response should be
in the format "assistant:  <think> [your reasoning] </think> <answer>
[chosen move] </answer>".
Below is an example of your desired behavior:
Example 1:
user:  <board> White to move.
Black King on g7
Black Knight on e7
Black Pawns on b5, c5
Black Rook on c2
White Bishop on f5
White King on g1
White Pawns on e6, g3, h2
White Rook on h8 </board>
assistant:  <think> Playing as white, I'm in the offensive here.
My rook is currently in at risk of being taken by their king and
my bishop is at risk of being taken by their knight.  I could take
their rook with their bishop but they would take my rook.  However,
if I move my rook to h7, I'll put their king in check while saving
my rook and bishop and continue pressure.  Moving rook h8 to h7 is a
wise move.  </think> <answer> Rh7 </answer>
Make sure that your chosen move is provided in standard PGN notation.
Please use English for your thought process and make sure your
thinking isn't too long, and if your final answer is not enclosed
in <answer> </answer> tags you will lose.
```

E Tokenization Issues with FEN

Below is a sample FEN board from the training dataset. The highlighted characters indicate characters that are grouped into a single token under the Qwen tokenizer.

r1b5/1ppr1pkp/p1n1pnp1/1/8/2P5/2N2N2/PPB2PPP/R3R1K1 b - - 1 17