LLM Chess: Benchmarking Reasoning and Instruction-Following in LLMs through Chess

Sai Kolasani[1], Maxim Saplin[2], Nicholas Crispino[3], Kyle Montgomery[3], Jared Davis[4], Matei Zaharia[1], Chi Wang[5], Chenguang Wang[3]
[1]UC Berkeley, [2]Independent Researcher, [3]UC Santa Cruz,
[4]Stanford University, [5] Google Deepmind
saikolasani@berkeley.edu, tutehabre@gmail.com

Abstract

We introduce LLM Chess, an evaluation framework designed to probe the generalization of reasoning and instruction-following abilities in large language models (LLMs) through extended agentic interaction in the domain of chess. We rank over 50 open and closed source models by playing against a random opponent using a range of behavioral metrics, including win and loss rates, move quality, move legality, hallucinated actions, and game duration. For a subset of models, we derive an Elo estimate by playing against a chess engine with variably configured skill. Despite the simplicity of the instruction-following task and the weakness of the opponent, many state-of-the-art models struggle to complete games or achieve consistent wins. Similar to other benchmarks on complex reasoning tasks, our experiments reveal a clear separation between reasoning and non-reasoning models. However, unlike existing static benchmarks, the stochastic and dynamic nature of LLM Chess uniquely reduces overfitting and memorization while preventing benchmark saturation. To support future work on evaluating reasoning and instruction-following in LLMs, we release our experimental framework, a public leaderboard, and a dataset of associated games. Our code is available at https://github.com/LLM-CHESS/llm_chess.

1 Introduction

Chess has long been viewed as an application for artificial intelligence (AI) since its inception, often being one of the first domains in which new technologies are used [Prost, 2012]. The idea of computer chess was pursued by the founders of AI, who viewed it as an exciting application in which advances could spur developments in other fields [Turing, 1988, Wiener, 2019, Shannon, 1950]. In fact, chess is often referred to as the 'drosophilia of AI', in that it both is a worthy testbed for experiments and also has guided the field's development [Simon and Schaeffer, 1992, McCarthy, 1990, Ensmenger, 2012]. As such, chess also has often been used to study cognitive abilities and decision making in humans [Groot, 1978, Simon and Chase, 1988, Sala et al., 2017, Sala and Gobet, 2017, Burgoyne et al., 2016, Blanch, 2022, Rosholm et al., 2017, Jankovic and Novak, 2019].

Since the 1950s, chess engines have been created with the hopes of beating humans, achieving various levels of success along the way. As time progressed, these engines advanced both through hardware and algorithmically, until reaching their current most powerful form with neural networks [Bernstein and de V. Roberts, 1958, Adel'son-Vel'skii et al., 1970, Newborn, 1979, Condon and Thompson, 1983, Campbell et al., 2002, Newborn, 2012, Silver et al., 2017]. While certain architectures and algorithms applied to chess have seen success elsewhere, these chess engines are explicitly tailored to chess games, unable to generalize.

Recently, large language models (LLMs) have shown incredibly competent performance in many diverse fields [Brown et al., 2020, Touvron et al., 2023, Thirunavukarasu et al., 2023, Liu et al., 2023, Wu et al., 2023b, Wei et al., 2022, OpenAI et al., 2024, DeepSeek-AI et al., 2025], leading many to wonder whether they may play an important role in achieving artificial general intelligence [Bubeck et al., 2023, Feng et al., 2024, Mumuni and Mumuni, 2025]. Additionally, tools like reinforcement learning and test-time scaling approaches have been shown to greatly increase reasoning abilities, accelerating the promise of a general reasoner [Chen et al., 2024, Shao et al., 2024, DeepSeek-AI et al., 2025]. While chess engines can now regularly beat humans, the game has not yet sufficiently been tested on LLMs, which ideally would possess such general characteristics that they could excel at any complex reasoning task, whether it be math, coding, or gameplaying like chess. As we start to design models with more general capabilities, what is old becomes new again: the large combinatorial spaces, long-horizon planning, and dynamic nature of chess all present thorough challenges for LLMs. Continuing the tradition of using chess to test and gain insights into current model capabilities, we present three main contributions:

1. We introduce LLM CHESS, a benchmark assessing both reasoning and instruction-following in the context of chess. Central to our benchmark is agentic interaction: by having LLMs play chess through autonomously selecting actions within a conversation, the difficulty comes not only in reasoning about the board and choosing the best move, but also how to formulate these choices. Unlike other reasoning benchmarks that can be contaminated or easily saturated, LLM CHESS is extensible by scaling the difficulty of the opponents and is not reliant on static board positions that can be included in training data.
2. We formulate a wide suite of per-model, per-game, and per-ply metrics to comprehensively evaluate the quality of each LLM's play, leveraging the depth of the chess domain to improve our analysis.
3. We evaluate over 50 models on LLM CHESS, showing that the domain of chess continues to present a challenging and informative reasoning task when applied to LLMs. We find that currently only the most powerful reasoning-enhanced LLMs can consistently beat a random player, even when we let them query for legal moves. When playing against engines, these models still fare poorly, with o3 (low) only achieving a 758 Elo in LLM CHESS. Through extensive ablations on specific parts of the game, we find that LLM performance varies widely based on the format of the conversations and prompt, suggesting a lack of robustness in their reasoning abilities.

Altogether, our comprehensive experiments show that chess is a worthy testbed for benchmarking the reasoning and instruction-following ability of LLMs and that current state-of-the-art models lack the ability to generalize their strong reasoning performance to be as impressive in chess as in other domains.

2 LLM CHESS

Here we introduce LLM CHESS (Figure 1), explaining our design choices and the metrics we use to score the models.

2.1 Design

In chess, an action taken by one side is referred to as a half-move or ply while two concurrent plys are referred to as a move, one by white, the other by black. [1] At each ply, we initiate a conversation with the end goal of outputting a valid chess move. We format all moves in Universal Chess Interface (UCI) format, a commonly used notation for chess engines [Huber and Meyer-Kahlen, 2000]. Each conversation consists of several turns, where each turn a LLM is prompted with instructions to output a valid action. We offer three actions to the LLM: 1) `get_current_board`, which fetches and presents the state of the current board using a unicode board, 2) `get_legal_moves`, which fetches a list of legal moves in UCI format, and 3) `make_move`, which takes a UCI-formatted string as input, adjusts the board state with that move, then ends the LLM's turn. Ablations on these choices are

[1] When it is clear that we are only discussing one side's actions, we occasionally overload move to refer to a ply, i.e., making a move in a ply refers to a single piece movement for that specific ply.

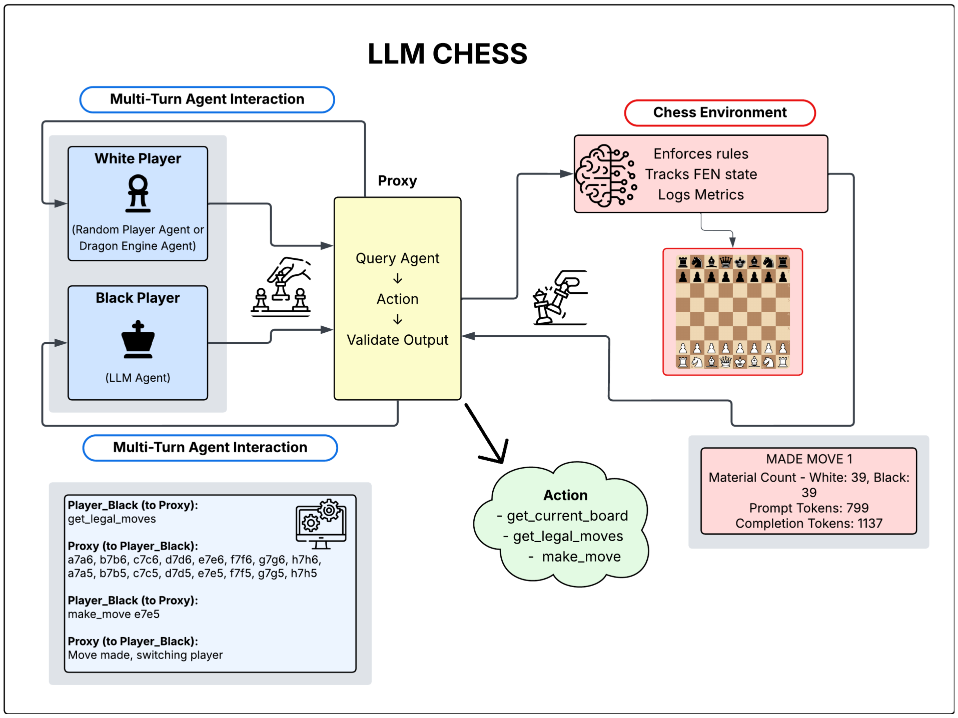


Figure 1: Overview of the LLM CHESS benchmark. White and Black player agents (random or engine for White, LLM for Black) interact with a central proxy that issues agent queries, validates outputs, and invokes one of three actions (`get_current_board`, `get_legal_moves`, `make_move`) The Chess Environment enforces the rules, updates and logs the FEN state, and records per-move metrics for downstream analysis.

presented in Section 3.4. We implement our LLM in an agentic setting using the AG2 framework [Wu et al., 2023a, Wang et al., 2025].

We cap each game at 100 moves (200 plys), have a max of 10 conversation turns per ply, and allow a max of 3 attempts per conversation turn for the LLM to provide a legal action or move. The LLMs view each ply as independent of all others, as we do not provide any game history. While this differs from humans who know their previous moves when playing chess, this aligns more with the machine setting where a model should be able to make the best move given the board state alone. Importantly, this setting does not eliminate the need for long-term planning: models must continue to be aware of how the moves they choose will impact future board states. Instructions provided to the LLM to initiate the conversation and resulting from various actions are presented in Appendix C. From preliminary testing, we somewhat surprisingly found many LLMs performed poorly against random players. So, we evaluate a wide set of models against random players to get a general sense of their abilities. Then, on particularly good models, we play them against a chess engine with variably configured skill.

Random Player We benchmark over 50 models by playing 30 games as black against a random player, who chooses a move at random from all legal moves. We choose a random player first because we want to focus on game-playing ability while removing skill as a main focus, i.e., to see if the model can play and finish a game of chess without having game-ending issues from instruction-following issues or choosing invalid moves.

Chess Engine From the initial models, we choose a subset of models to play against Komodo's Dragon 1 engine, which can be set at various skill levels from 1-25. As an estimate, Skill 1 is around Elo 250, then each subsequent skill level is a 125 boost in Elo based on chess.com games [Kaufman and Lefler, 2020]. Since chess.com is one of the most popular online chess platforms, having over 200 million members [Chess.com, 2025], this lets us ground our LLM performance in the real world. We run experiments against Dragon 1 at 30 games per skill level starting at Skill 1 and up to Skill

5, representing Elos of $\{250, 375, 500, 625, 750\}$ on chess.com. While currently we do not evaluate with too high of Skills, our framework permits easy extensibility: as LLMs become better and better, we can increase the difficulty of the opponents to prevent saturation.

2.2 Metrics

LLM CHESS evaluates LLMs by playing full chess games. However, we also evaluate the reasoning ability of the LLM with various per-ply metrics rating the quality of each move, as well as the instruction-following ability by examining how the model engages with our agentic structure.

Per-model The main way we quantify performance is to calculate a LLM's Win/Loss percentage against an opponent, which is the difference between wins and losses as a percentage of total games:

$$\text{Win/Loss} = \frac{1}{2}\left(\frac{\text{llm\_wins} - \text{opponent\_wins}}{\text{total\_games}}\right) + 0.5$$

Win/Loss admits easy interpretability: 50% means a model has equal wins and losses. To win a game, LLM must checkmate its opponent. LLMs can lose or draw in the following ways: 1) Chess-based. The LLM could lose through checkmate by the opponent or draw due to various rules (stalemate, insufficient material, seventy-five moves without a capture or pawn move, fivefold repetition, or the game reached 100 moves). 2) Instruction-based. The LLM loses if it reaches the maximum number of conversation turns without making a move (10) or if it reached the maximum number of attempts (3) at a conversation turn without selecting a valid action. We call failures here instruction-following errors. 3) Model errors. These are errors due to the model or how it's served like timeout for reasoning models. We exclude all games with these errors when playing against a random player so we could better analyze behavior, but include them when playing against Dragon 1 to simulate what would happen in a real-world scenario.

While Win/Loss is helpful for observing the quality of LLM performance against weaker opponents, it is less grounded in the world of chess. So, for LLMs that perform sufficiently well against random players and against the engine at various skill levels, we calculate Elo [Elo, 1978]. Normally Elo ratings update dynamically between players, but here we treat each engine opponent's rating R_i as fixed and encode the LLM's game outcomes as $S_i \in \{1, 0.5, 0\}$. Under Elo theory, the expected score against R_i is

$$E_i(R) = \frac{1}{1 + 10^{(R_i - R)/400}}.$$

Rather than updating R incrementally, we find the maximum-likelihood rating $\hat{R}$ by solving $\sum_i \big(S_i - E_i(\hat{R})\big) = 0$. Around $\hat{R}$, the observed Fisher information $\mathcal{I}(\hat{R}) = \sum_i E_i(\hat{R})\big(1 - E_i(\hat{R})\big)(\ln 10/400)^2$ yields a standard error $\mathrm{SE} = 1/\sqrt{\mathcal{I}}$ and thus a 95% confidence interval $\hat{R} \pm 1.96\,\mathrm{SE}$ [Glickman, 1999]. We detail the exact skill levels we evaluate against for each model in the experiments section.

Per-game For each game, we calculate the number of moves per game and the reason for each loss. We also record other metrics focused on instruction-following throughout the game that do not depend on the quality of the moves. For `get_current_board` and `get_legal_moves` we calculate the average number of times that action was called per ply. We also calculate the average number of times `make_move` was called but resulted in an invalid move, as well as the average number of invalid actions that were selected.

Per-ply Besides analyzing performance on a game level, we also calculate the performance per ply. After the LLM calls `make_move` in each ply, we calculate the Win% (Equation (1)), the chance of winning a game from the given position as defined by Lichess [Lichess, 2025]. This analysis is based on centipawns, which are calculated by Stockfish representing how much worse the player's move was than the engine's [Linville, 2023]. We present the Win% for the LLM averaged over each ply, which tells us whether the LLM held a more favorable position throughout the game.

$$\text{Win\%} = 50 + 50 * (2/(1 + \exp(-0.00368208 * \text{centipawns})) - 1) \tag{1}$$

Then, based on the difference in Win%, $\Delta = \text{Win\%}_{\text{before move}} - \text{Win\%}_{\text{after move}}$ (where a higher Δ means the player's Win% decreased), we can calculate Blunders, Mistakes, and Inaccuracies, common classifications of moves used by online chess platforms, following the Lichess cutoffs [Lichess, 2023]:

$$\text{Judgment} = \begin{cases} \text{Blunder} & \text{if } \Delta \geq 30 \\ \text{Mistake} & \text{if } \Delta \geq 20 \\ \text{Inaccuracy} & \text{if } \Delta \geq 10 \end{cases} \tag{2}$$

We present the average Blunder, Mistake, and Inaccuracy rate per ply, as well as Best, the rate in which the LLM selected the best move as identified by Stockfish. We note that since our Win% scores are based on centipawns, these metrics can depend on the hyperparameters of Stockfish. Full implementation details (including the `score_to_cp` function and `BLUNDER_THRESHOLD`, `MISTAKE_THRESHOLD`, `INACCURACY_THRESHOLD`) are available in Appendix A.

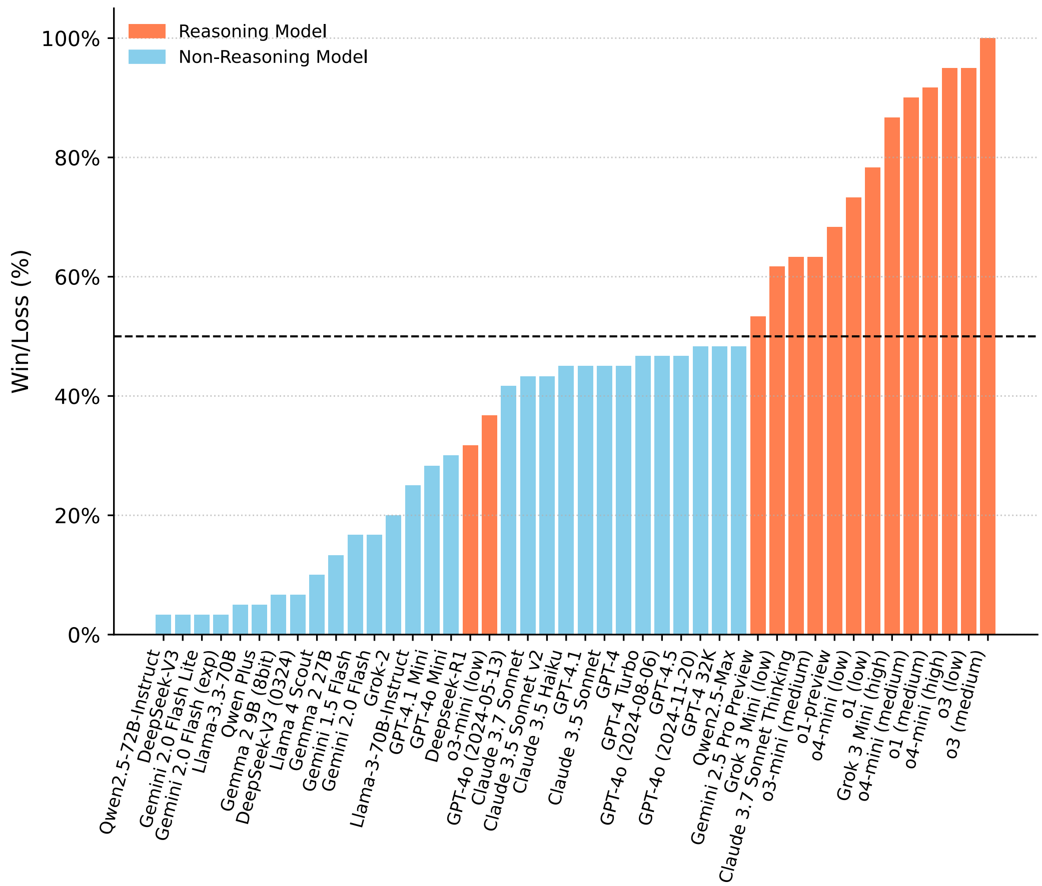


Figure 2: Win/Loss of LLM players versus random opponents. The dashed line marks a Win/Loss of 50%, which represents an equal amount of wins and losses.

3 Experiments

By default, all LLMs are run with a temperature of 0.3 and a Top P of 1.0. More details about the models we evaluate on and how they are run is detailed in Appendix A.

3.1 LLMs vs. Random

We present the Win/Loss of 44 LLMs versus a random player for 30 games in Figure 2. Most notably, we find that most models are not able to consistently beat a random player; in fact, only models with reasoning abilities are able to perform better than 50%. To analyze the reasons behind this poor performance, we present per-game metrics including how the LLMs won and lost in Table 1. Note that the only way black can win is through a checkmate. For each of these metrics, we present the average over all reasoning and non-reasoning models, as well as on the two top and bottom reasoning and non-reasoning models.

Table 1: Per-game metrics for Reasoning (shaded) vs Non-Reasoning models. We choose the top and bottom two models in each category (ranked among 15 reasoning, 29 non-reasoning models) based on Win/Loss from among all models with a Win/Loss over zero. We include the percent of losses due to errors in instruction-following (Instruction) or checkmates by white (MateW), as well as the amount of draws (Draw), checkmates by black (MateB), and average moves over all games.

Model	Instruction (%)	Draw (%)	MateW (%)	MateB (%)	Avg Moves
Reasoning Avg	24.4	30.2	0.0	45.4	93.7
Non-Reasoning Avg	71.9	24.6	2.8	0.7	73.9
o3 (medium)[1]	0.0	0.0	0.0	**100.0**	40.1
o3 (low)[2]	0.0	10.0	0.0	90.0	63.5
Qwen2.5-Max[1]	0.0	**96.7**	3.3	0.0	**197.4**
GPT-4o (2024-11-20)[2]	0.0	90.0	**6.7**	3.3	194.9
o3-mini (low)[14]	36.7	53.3	0.0	10.0	139.3
Deepseek-R1[15]	60.0	16.7	0.0	23.3	88.2
Gemini 2.0 Flash Lite[28]	**90.0**	0.0	**6.7**	3.3	90.3
Qwen2.5-72B-Instruct[29]	**90.0**	6.7	3.3	0.0	64.1

Our results indicate that reasoning-enchanced LLMs dramatically outperform non-reasoning models in our random-opponent setting. Reasoning models have an average win rate of 45.4% with the top performers achieving close to 100%, whereas non-reasoning models have an average win rate of 0.7% with the top performer achieving 3.3%. This performance gap is further supported by a three-fold reduction in instruction-following errors: 72% for non-reasoning models vs 24% for reasoning models. Lastly, non-reasoning models almost always reach the maximum moves allowed if they don't have instruction-following issues, whereas reasoning models converge around 94 moves per game. While these statistics demonstrate that enhanced reasoning capabilities substantially improve both instruction-following and overall game performance, even the best LLMs secure wins in only about 90% of games against a random opponent, indicating poor real world performance.

Table 2: Per Ply Classification Rates (%) for Reasoning (shaded) vs Non-Reasoning Models.

Model	Blunder (↓)	Mistake (↓)	Inaccuracy (↓)	Best (↑)
GPT-4.1-mini	31.3	8.7	13.4	4.1
o4-mini (low)	11.2	3.5	5.5	10.8
o4-mini (medium)	**4.2**	**1.1**	**4.0**	**19.5**
Grok 3 Mini	9.1	2.0	5.8	8.5
Grok 3 Mini (low)	13.6	2.8	6.7	9.8
Grok 3 Mini (high)	**4.9**	**1.5**	**3.4**	**18.2**

To see how models perform throughout the game, we present per-ply metrics on a handful of models performing at various levels in Table 2. Our results show that reasoning models make far fewer bad moves and substantially more "best" moves than non-reasoning models. For example, o4-mini (medium) blunders only 4.2% and mistakes 1.1% of the time per ply, compared to 31.3% blunders and 8.7% mistakes for GPT-4.1-mini. Furthermore, o4-mini (medium) selects the "Best" move 19.5% of the time versus just 4.1% for GPT-4.1-mini. The same trend continues with Grok 3 Mini and Grok 3 Mini (high). These results confirm that enhanced reasoning capacity reduces catastrophic errors while boosting tactical decision making.

Notably, we also ran experiments on over 10 models that have a 0% Win/Loss, often resulting from difficulties with instruction-following. We present these models in Table 4 in Appendix B. We also present additional results for some models on more games in Appendix B.

3.2 LLMs vs. Chess Engine

While random players are a good test of LLMs' abilities to complete games, they often make moves that are nonsensical and are not realistic as a chess opponent. As such, some LLMs are able to perform very well against random players: the best models o3 (medium/low) and o4-mini (high) have a Win/Loss of at least 90%. To increase the difficulty of the games and ground LLMs in real-world performance, we now focus on our most powerful models to play against Dragon 1: o3 (low), Grok 3 Mini (high), o4-mini, o3-mini .

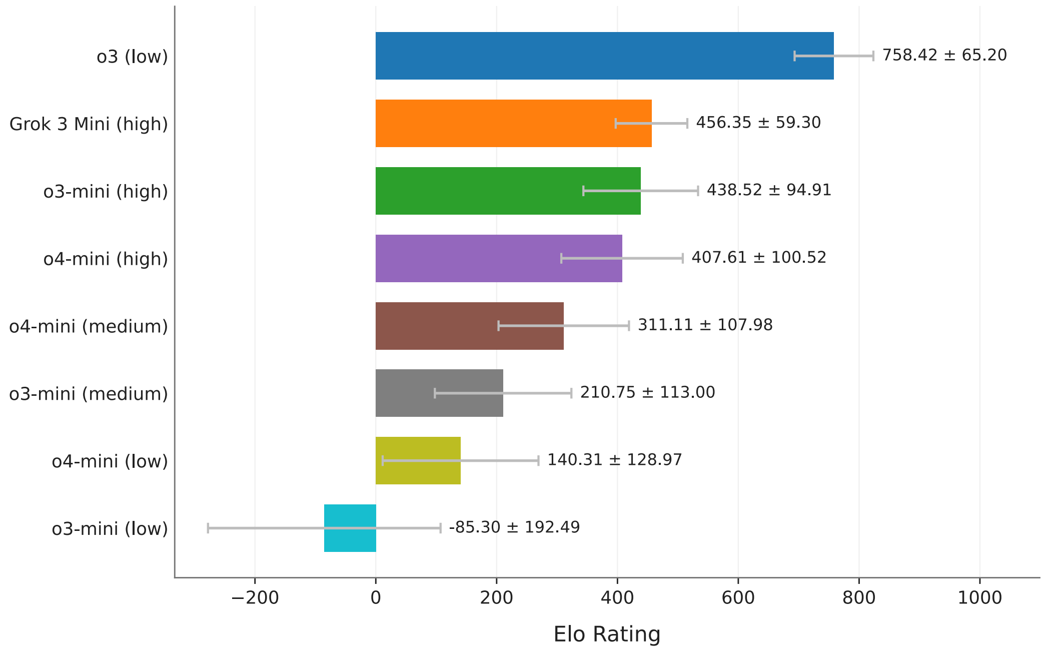


Figure 3: Elo of top reasoning models estimated using Dragon 1.

Figure 4 reports estimated Elo ratings (±95 % CI) for o3 (low), Grok 3 Mini (high), o4-mini, o3-mini when playing at least 30 games against Dragon 1 at skill 1. For o3 (low) and Grok 3 Mini (high) we play against all skills 1–5 (Elos 250–750). We include more about the models, skills they played against, and Elo calculation in Appendix A. These Elo estimates confirm several key insights. First, increased reasoning effort directly translates to higher real-world playing strength. For example, boosting o4-mini from "low" to "medium" reasoning settings raises its Elo by roughly 170 points. Second, even the strongest LLM in our study, o3 (low), peaks at an adjusted Elo of about 758, which remains far below human master level (approximately 2000), underscoring how far LLMs lag behind specialized chess engines and general human gameplay.

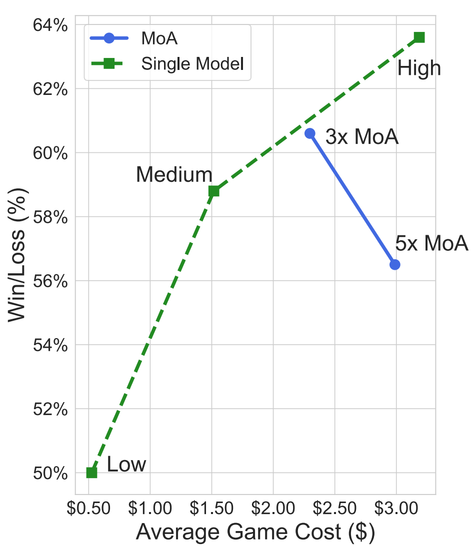
3.3 Exploring Test-time Scaling

Scaling Wide Besides increasing the number of tokens one model uses, we also run experiments using multiple instances of the same model in parallel. To do so, we apply a Mixture-of-Agents (MoA) approach where we have multiple proposer model calls fed into a separate aggregator model that provides the output [Wang et al., 2024]. This occurs at every step of the conversation . We run two settings on 30+ games with black against Dragon 1 Skill 1 using either 3x and 5x o4-mini (low) as the proposers and always use o4-mini (medium) as the aggregator. Results are in Figure 4a. [2]

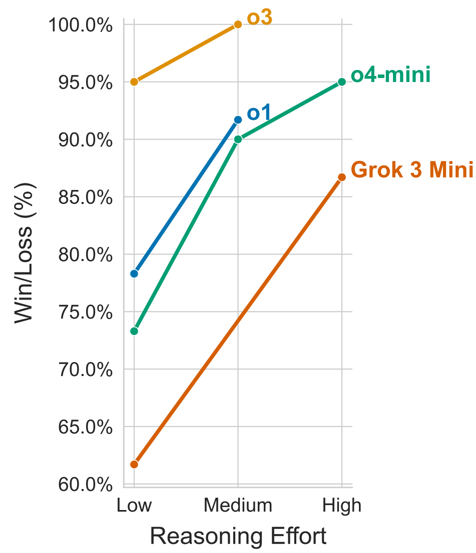
Scaling Deep We show o1, o3, o4-mini, and Grok 3 Mini at various reasoning levels vs a random player in Figure 4b. Similar to other reasoning domains, we find scaling with more tokens improves performance on LLM Chess, with increases of up to 15% from low to medium, and 20% from low to high. [3]

[2]Note that we tried to use o4-mini (low) as the aggregator but it failed, not providing a valid action but instead commenting on the quality of the proposers' responses.

[3]Empirically, we notice that as we try to run OpenAI models with higher reasoning effort, they are more likely to result in a timeout. See Appendix D for further discussion.



(a) Scaling wide with o4-mini and MoA.



(b) Scaling deep with increased reasoning effort.

Figure 4: Performance comparisons of reasoning models. (a) Cost-performance tradeoff for Win/Loss with o4-mini variants at each possible reasoning effort along with 3x and 5x MoA using o4-mini (low) as the proposer and o4-mini (medium) as the aggregator. (b) Win/Loss when scaling with variable reasoning effort.

3.4 Ablations

We design three types of ablations on o4-mini (low) and Grok 3 Mini (low) by varying the actions we present to the model during the conversation (Actions), the state of the board from the LLM's perspective (Board Representation), and adding or removing information the LLM has access to during the conversation (Changing Information). In each of the settings in each category we run 30 games per model against a random player with the LLM playing as black (unless stated otherwise). Results are in Table 6 in Appendix B. With these results, we see performance varies widely, showing the lack of robustness in reasoning in the chess setting.

Overall, we find that simplifying the agentic scenario by removing actions and instead supplying the removed information automatically shows an increase in performance on both Grok 3 Mini (low) and o4-mini (low). In both cases, offering only `make_move` offers substantial improvements in Win/Loss, with o4-mini (low)'s performance increasing by over 20%. This signifies the difficulty of reasoning models engaging in agentic interactions in LLM CHESS. Performance with both an ASCII board and FEN is similar to our default setting for Grok 3 Mini (low), while for o4-mini (low) we see performance improve by over 15% in both cases, reaching 95% for FEN. This suggests that some LLMs have similar performance across board representations, while some have trouble generalizing.

Though LLM CHESS's agentic setting can be challenging for some models, a major advantage given to the model is their ability to query for legal moves with `get_legal_moves`.

When removing this ability, we see a decline in model capabilities of almost 30% for Grok 3 Mini (low) and 10% for o4-mini (low), meaning that information is still difficult for LLMs to collect. We also experiment with including the previous moves, mimicking the information a human would have when playing chess. With this, we see performance is similar to the respective settings without previous moves, showing that at least against random players, the LLM's are not gaining much about knowing what has already occurred in the game.

4 Related Work

Chess and AI Transformers have been applied to chess in both foundation and domain-specific settings. While prior work has suggested that large language models (LLMs) display surprising competence in chess [Dynomight, 2024, Acher, 2023], these findings often rely on a small set of models, static PGN completions, or idealized prompting conditions. Studies such as the Chess Transformer [Noever et al., 2020], Chessformer [Monroe and Chalmers, 2024], and BERT-based rule learners [DeLeo and Guven, 2022] demonstrate improved move legality and opening play, but confine game play to offline or single-turn evaluations. Our findings show that when evaluated in interactive or compositional settings, language models fail to adhere to basic rules, lose track of the game state, or hallucinate with illegal moves. More recent work has involved fine-tuning transformer architectures directly on a large-scale chess corpus, such as ChessGPT [Feng et al., 2023] and Amortized Planning Transformers [Ruoss et al., 2024], with the latter treating chess as a planning problem. While these approaches show promise, they are typically assessed on win rate or move legality, focusing little on generalization, instruction-following, or reasoning. For LLMs, several open-source efforts have attempted to benchmark on chess tasks, such as generating legal moves or competing in scripted tournaments [Carlini, 2024, Ndzomga, 2024]. Other analyses examine how LLMs internalize chess rules from PGNs [Stöckl, 2021] and how LLMs can predict chess puzzle difficulty [Miłosz and Kapusta, 2024] or they include chess as part of a larger benchmark [Khan et al., 2025]. While these frameworks provide initial insights, they typically focus only on outcome-level metrics such as win/loss or Elo, often over a narrow set of models. In contrast, our benchmark systematically exposes these limitations across a diverse model pool, revealing fragility in real-time play and strategic reasoning.

Strategic Reasoning and Game Benchmarks Our work builds on a growing field of literature that poses games as testbed for strategic and multi-step reasoning. GTBench [Duan et al., 2024] and ZeroSumEval [Khan et al., 2025] leverage inter-model competition to assess strategy and robustness, while ChatArena [Wu et al., 2023c] and MastermindEval [Zhang et al., 2024] extend the space of game evaluation into multimodal and logic-heavy tasks. Additional studies in multi-game consistency [Toshniwal et al., 2022] highlight gaps in rule following and tactical depth when LLMs pivot between environments. While these efforts highlight the strengths and limitations of LLMs in planning, consistency, and rule/instruction following, they are typically spread across tasks or lack domain-specific human interpretability. Chess on the other hand, is a deeply studied environment with transparent rules, interpretable decision sequences, and established human baselines. Our benchmark combines all of these strengths in a reproducible testbed that evaluates both instruction-following and multi-step reasoning under game constraints.

5 Conclusion

Chess has long been an important factor in the development of AI systems. However, LLMs, today's most powerful generalist models, have yet to have been sufficiently tested on the domain, missing out on the insights that have often been made by doing so. To remedy this, we introduced LLM CHESS, a benchmarking framework for reasoning and instruction-following in LLMs in chess. Compared to standard reasoning benchmarks, our setting is more difficult: unlike math or coding where LLMs are beginning to reach the level of seasoned experts, LLMs in our chess framework are weak and many can barely consistently beat even a random player. LLM CHESS also allows for easy dynamic extensibility through modification of the skill of opponents, as well as resistance to memorization given the combinatorial spaces in chess.

6 Limitations

With LLM CHESS, we present a framework for evaluating LLMs on reasoning and instruction-following in chess by calculating Win/Loss, Elo, and per-move diagnostics. We impose a 100-move cap (200 plies) and restrict each ply to 10 conversation turns with a maximum of 3 action attempts, which can prematurely end deeper strategic sequences. Prompt length constraints and AG2 client timeouts (10 min per move) also prevented a lot of model runs from completing their reasoning, introducing variability in measured performance and limiting reproducibility under longer-horizon settings. We present additional information about the time out errors in Appendix D.

References

M. Acher. Debunking the chessboard: Confronting gpts against chess engines to estimate elo ratings and assess legal move abilities. https://blog.mathieuacher.com/GPTsChessEloRatingLegalMoves/, 2023.

G. M. Adel'son-Vel'skii, V. L. Arlazarov, A. Bitman, A. Zhivotovskii, and A. V. Uskov. Programming a computer to play chess. *Russian Mathematical Surveys*, 25(2):221, 1970.

A. Bernstein and M. de V. Roberts. Computer v. chess-player. *Scientific American*, 198(6):96–107, 1958.

A. Blanch. Chess Instruction Improves Cognitive Abilities and Academic Performance: Real Effects or Wishful Thinking? *Educational Psychology Review*, 34(3):1371–1398, Sept. 2022. ISSN 1040726X. doi: 10.1007/s10648-022-09670-9. URL https://www.proquest.com/docview/2700444564/abstract/4D353806656F41A3PQ/1. Num Pages: 1371-1398 Place: New York, Netherlands Publisher: Springer Nature B.V.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4, Apr. 2023. URL http://arxiv.org/abs/2303.12712. arXiv:2303.12712 [cs].

A. P. Burgoyne, G. Sala, F. Gobet, B. N. Macnamara, G. Campitelli, and D. Z. Hambrick. The relationship between cognitive ability and chess skill: A comprehensive meta-analysis. *Intelligence*, 59:72–83, Nov. 2016. ISSN 0160-2896. doi: 10.1016/j.intell.2016.08.002. URL https://www.sciencedirect.com/science/article/pii/S0160289616301593.

M. Campbell, A. J. Hoane Jr, and F.-h. Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

N. Carlini. chess-llm. https://github.com/carlini/chess-llm, 2024. Accessed: 2025-05-14.

L. Chen, J. Q. Davis, B. Hanin, P. Bailis, I. Stoica, M. Zaharia, and J. Zou. Are more llm calls all you need? towards scaling laws of compound inference systems, 2024. URL https://arxiv.org/abs/2403.02419.

Chess.com. Chess.com members. https://www.chess.com/members, 2025. Accessed: 2025-05-02.

J. H. Condon and K. Thompson. Belle. In *Chess skill in man and machine*, pages 201–210. Springer, 1983.

DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu,

Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, Jan. 2025. URL http://arxiv.org/abs/2501.12948. arXiv:2501.12948 [cs].

M. DeLeo and E. Guven. Learning chess with language models and transformers. In *Data Science and Machine Learning*, DSML 2022, page 179–190, 2022. doi: 10.5121/csit.2022.121515. URL http://dx.doi.org/10.5121/csit.2022.121515.

J. Duan, R. Zhang, J. Diffenderfer, B. Kailkhura, L. Sun, E. Stengel-Eskin, M. Bansal, T. Chen, and K. Xu. Gtbench: Uncovering the strategic reasoning capabilities of llms via game-theoretic evaluations. In *Advances in Neural Information Processing Systems*, volume 37, pages 28219–28253, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/3191170938b6102e5c203b036b7c16dd-Paper-Conference.pdf.

Dynomight. Something weird is happening with llms and chess. https://dynomight.net/chess/, 2024.

A. E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., New York, 1978. ISBN 0668047216 9780668047210.

N. Ensmenger. Is chess the drosophila of artificial intelligence? A social history of an algorithm. *Social Studies of Science*, 42(1):5–30, Feb. 2012. ISSN 0306-3127, 1460-3659. doi: 10.1177/0306312711424596. URL https://journals.sagepub.com/doi/10.1177/0306312711424596.

T. Feng, C. Jin, J. Liu, K. Zhu, H. Tu, Z. Cheng, G. Lin, and J. You. How Far Are We From AGI: Are LLMs All We Need?, Nov. 2024. URL http://arxiv.org/abs/2405.10313. arXiv:2405.10313 [cs].

X. Feng, Y. Luo, Z. Wang, H. Tang, M. Yang, K. Shao, D. Mguni, Y. Du, and J. Wang. Chessgpt: Bridging policy learning and language modeling. *arXiv preprint arXiv:2306.09200*, 2023. URL https://arxiv.org/abs/2306.09200.

M. E. Glickman. Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 48(3):377–394, 1999.

A. D. d. Groot. *Thought and Choice in Chess*. Walter de Gruyter, 1978. ISBN 978-90-279-7914-8. Google-Books-ID: EI4gr42NwDQC.

R. Huber and S. Meyer-Kahlen. Universal chess interface (uci) protocol specification, 2000. URL https://www.chessprogramming.org/UCI.

A. Jankovic and I. Novak. Chess as a Powerful Educational Tool for Successful People. 2019.

L. Kaufman and M. Lefler. Komodo Chess - README.txt. https://komodochess.com/store/pages.php?cmsid=14, 2020.

H. Khan, H. A. Alyahya, Y. Alnumay, M. S. Bari, and B. Yener. Zerosumeval: Scaling llm evaluation with inter-model competition. *arXiv preprint arXiv:2504.12562*, 2025. URL https://arxiv.org/abs/2504.12562.

Lichess. Advice.scala in lila repository. https://github.com/lichess-org/lila/blob/cf9e10df24b767b3bc5ee3d88c45437ac722025d/modules/analyse/src/main/Advice.scala, 2023. Accessed: 2025-05-07.

Lichess. Lichess Accuracy metric. https://lichess.org/page/accuracy, 2025. Accessed: May 4, 2025.

R. Linville. Understanding average centipawn loss in chess, 2023. URL https://www.chess.com/blog/raync910/average-centipawn-loss-chess-acpl.

H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual Instruction Tuning, Dec. 2023. URL http://arxiv.org/abs/2304.08485. arXiv:2304.08485 [cs].

J. McCarthy. Chess as the Drosophila of AI. In T. A. Marsland and J. Schaeffer, editors, *Computers, Chess, and Cognition*, pages 227–237. Springer New York, New York, NY, 1990. ISBN 978-1-4613-9082-4 978-1-4613-9080-0. doi: 10.1007/978-1-4613-9080-0_14. URL http://link.springer.com/10.1007/978-1-4613-9080-0_14.

S. Miłosz and P. Kapusta. Predicting chess puzzle difficulty with transformers, 2024. URL https://arxiv.org/abs/2410.11078.

D. Monroe and P. A. Chalmers. Mastering chess with a transformer model, 2024. URL https://arxiv.org/abs/2409.12272.

A. Mumuni and F. Mumuni. Large language models for artificial general intelligence (agi): A survey of foundational principles and approaches, 2025. URL https://arxiv.org/abs/2501.03151.

F. S. Ndzomga. What happens when llms play chess? https://github.com/fsndzomga/chess_tournament_nebius_dspy, 2024. Accessed: 2025-05-14.

M. Newborn. Chess 4.7 gives levy a run for his money. *The Mathematical Intelligencer*, 1:215–217, 1979.

M. Newborn. *Kasparov versus Deep Blue: Computer Chess Comes of Age*. Springer Science & Business Media, Dec. 2012. ISBN 978-1-4612-2260-6. Google-Books-ID: IiXjBwAAQBAJ.

D. Noever, M. Ciolino, and J. Kalin. The chess transformer: Mastering play using generative language models, 2020. URL https://arxiv.org/abs/2008.04057.

OpenAI. Learning to reason with LLMs, Sept. 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. d. A. B. Peres, M. Petrov, H. P. d. O. Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. J. Weinmann, A. Welihinda,

P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph. GPT-4 Technical Report, Mar. 2024. URL http://arxiv.org/abs/2303.08774. arXiv:2303.08774 [cs].

F. Prost. On the Impact of Information Technologies on Society: an Historical Perspective through the Game of Chess, Mar. 2012. URL http://arxiv.org/abs/1203.3434. arXiv:1203.3434 [cs].

M. Rosholm, M. B. Mikkelsen, and K. Gumede. Your move: The effect of chess on mathematics test scores. *PloS one*, 12(5):e0177257, 2017.

A. Ruoss, G. Delétang, S. Medapati, C. Zang, and I. Mordatch. Amortized planning with large-scale transformers: A case study on chess. *arXiv preprint arXiv:2402.04494*, 2024. URL https://arxiv.org/abs/2402.04494.

G. Sala and F. Gobet. Does chess instruction improve mathematical problem-solving ability? Two experimental studies with an active control group. *Learning & Behavior*, 45(4):414–421, Dec. 2017. ISSN 1543-4508. doi: 10.3758/s13420-017-0280-3. URL https://doi.org/10.3758/s13420-017-0280-3.

G. Sala, A. P. Burgoyne, B. N. Macnamara, D. Z. Hambrick, G. Campitelli, and F. Gobet. Checking the "Academic Selection" argument. Chess players outperform non-chess players in cognitive skills related to intelligence: A meta-analysis. *Intelligence*, 61:130–139, Mar. 2017. ISSN 0160-2896. doi: 10.1016/j.intell.2017.01.013. URL https://www.sciencedirect.com/science/article/pii/S0160289616301635.

C. E. Shannon. XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, Mar. 1950. ISSN 1941-5982. doi: 10.1080/14786445008521796. URL https://doi.org/10.1080/14786445008521796. Publisher: Taylor & Francis.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

H. Simon and W. Chase. Skill in chess. In *Computer chess compendium*, pages 175–188. Springer, 1988.

H. A. Simon and J. Schaeffer. Chapter 1 The game of chess. In *Handbook of Game Theory with Economic Applications*, volume 1, pages 1–17. Elsevier, 1992. ISBN 978-0-444-88098-7. doi: 10.1016/S1574-0005(05)80004-9. URL https://linkinghub.elsevier.com/retrieve/pii/S1574000505800049.

A. Stöckl. Watching a language model learning chess. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1369–1379, 2021. URL https://aclanthology.org/2021.ranlp-1.153/.

A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting. Large language models in medicine. *Nature Medicine*, 29(8):1930–1940, Aug. 2023. ISSN 1546-170X. doi: 10.1038/s41591-023-02448-8. URL https://www.nature.com/articles/s41591-023-02448-8. Publisher: Nature Publishing Group.

S. Toshniwal, S. Wiseman, K. Livescu, and K. Gimpel. Chess as a testbed for language model state tracking, 2022. URL https://arxiv.org/abs/2102.13249.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and Efficient Foundation Language Models, Feb. 2023. URL http://arxiv.org/abs/2302.13971. arXiv:2302.13971 [cs].

A. M. Turing. Chess. In D. Levy, editor, *Computer Chess Compendium*, pages 14–17. Springer New York, New York, NY, 1988. ISBN 978-1-4757-1968-0. doi: 10.1007/978-1-4757-1968-0_2. URL https://doi.org/10.1007/978-1-4757-1968-0_2.

C. Wang, Q. Wu, and A. Contributors. Ag2: Open-source framework for building ai agents. https://docs.ag2.ai/latest/docs/home/, 2025.

J. Wang, J. Wang, B. Athiwaratkun, C. Zhang, and J. Zou. Mixture-of-Agents Enhances Large Language Model Capabilities, June 2024. URL http://arxiv.org/abs/2406.04692. arXiv:2406.04692 [cs].

J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned Language Models Are Zero-Shot Learners, Feb. 2022. URL http://arxiv.org/abs/2109.01652. arXiv:2109.01652 [cs].

N. Wiener. *Cybernetics or control and communication in the animal and the machine*. The MIT Press, Oct. 2019. ISBN 978-0-262-35590-2. doi: 10.7551/mitpress/11810.001.0001. URL https://doi.org/10.7551/mitpress/11810.001.0001. tex.eprint: https://direct.mit.edu/book-pdf/2254528/book_9780262355902.pdf.

Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023a. URL https://arxiv.org/abs/2308.08155.

S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann. BloombergGPT: A Large Language Model for Finance, Dec. 2023b. URL http://arxiv.org/abs/2303.17564. arXiv:2303.17564 [cs].

Y. Wu, Z. Jiang, A. Khan, Y. Fu, L. Ruis, E. Grefenstette, and T. Rocktäschel. Chatarena: Multi-agent language game environments for large language models. https://github.com/chatarena/chatarena, 2023c.

Y. Zhang, S. Mao, T. Ge, X. Wang, A. de Wynter, Y. Xia, W. Wu, T. Song, M. Lan, and F. Wei. Llm as a mastermind: A survey of strategic reasoning with large language models. *arXiv preprint arXiv:2404.01230*, 2024. URL https://arxiv.org/abs/2404.01230.

A Experimental Settings

We ran all LLMs with a default temperature of 0.3 and Top P of 1.0 for the models that took them as parameters (some models like OpenAI's reasoning models don't take a temperature). If models like Deepseek-R1 have a recommended temperature (0.6), we try to use that instead.

A.1 Centipawn Calculation using Stockfish

We ran Stockfish v17 (path configurable via `stockfish_path`) in UCI mode with the following settings: fixed analysis depth of 20 plies, no time limit per move, a single thread, 128 MB hash size, MultiPV=1, and Skill Level=20. We convert the engine's `Cp` or `Mate` score to centipawns via a standardized function: centipawn values directly for `Cp` evaluations, and ± 1000 for any mate score: positive for winning mates, negative for losing mates. Blunder, Mistake, and Inaccuracy thresholds are based on Lichess's Win% cutoffs: 30%, 20%, and 10% respectively [Lichess, 2023]. These hyperparameters provide consistent, interpretable per-ply metrics while keeping analysis costs tractable.

A.2 Dragon 1 Settings

All Dragon 1 experiments were run on the following computer: Windows 11, WSL 2, Core i5 13600KF, 64GB DDR5 RAM, RTX4090.

A.3 Model Information

In Table 3 we map all the API model names and additional settings (e.g., quantization) to their cleaned name used in the paper. Note that all open source models not run through an API (e.g., groq) were run with quantization on a RTX 4090.

A.4 Elo Calculations

To calculate Elo, we played at least 33 total games against varying skill levels in Dragon 1 with the following models: o3 (low), Grok 3 Mini (high), o4-mini, o3-mini . We provide Win/Loss and number of games against each skill level in Table 5. Note that we played o3 (low) and Grok 3 Mini (high) against skills 1-5 (each $\geq$ 169 games), and o4-mini (high) against skills 1-2 (each $\geq$ 49 games), and the rest of the models against skill 1 (each $\geq$ 33 games). We also played o3 (low) against skill 10 because we found that it performed quite well against skill 5 (71.9% Win/Loss). However, we found that against skill 10, o3 (low) only achieved a 3.0% Win/Loss, meaning even the most powerful model we thoroughly tested still has a ways to go.

Pseudocode for the Elo calculations resulting in the values in Figure 3 is in Algorithm 1, which takes in a list of opponents with their Elo and corresponding win (1), draw (0.5), loss (0) and calculates an estimate for the LLM's Elo and a 95% confidence interval. Notably, when calculating Elo we add a correction of 35 points to correct for the fact that the LLMs always play as black. We base this on analysis finding that white empirically wins about 54% of games when facing an opponent of the same rating, which equates to 35 points[4].

B Additional Results

B.1 Ablations

We present full results on all our ablations for Grok 3 Mini (low) and o4-mini (low) in Table 6. We always play 30 games against a random player with the LLM as black except for the LLM as white setting, where the roles are reversed. We also use the default unicode board in all settings except the No Legal Moves setting. Because the default unicode board does not have all board information (e.g., castling rights), we provide a FEN for No Legal Moves instead, meaning we are comparing to the FEN setting as the No Legal Moves baseline. We also

[4] https://en.chessbase.com/post/the-sonas-rating-formula-better-than-elo

Algorithm 1 Estimate True Elo Rating

Require: Records $R = \{(E_i, S_i)\}_{i=1}^{n}$ $\triangleright$ E_i opponent Elo, $S_i \in \{0, 0.5, 1\}$
Require: White-advantage W $\triangleright$ 35 Elo
Ensure: True rating R_{true} and 95% CI half-width ME

function $\textsc{ExpectedScore}(r,\ (E_i)_{i=1}^{n})$
  for $i \leftarrow 1$ **to** n **do**
    $\hat{S}_i \leftarrow 1 \,/\, \left(1 + 10^{(E_i - r)/400}\right)$
  end for
  return $(\hat{S}_i)_{i=1}^{n}$
end function
function $\textsc{ScoreDiff}(r)$
  $\hat{S} \leftarrow \textsc{ExpectedScore}(r,\ (E_i)_{i=1}^{n})$
  return $\sum_{i=1}^{n} (\, S_i - \hat{S}_i \,)$
end function
// 1) Solve for the black rating of the LLM
$R_{\text{black}} \leftarrow \textsc{FindZero}(\text{ScoreDiff}, [\min_i E_i - 400,\ \max_i E_i + 400])$ $\triangleright$ find r such that $\text{ScoreDiff}(r) = 0$ and is within 400 Elo of the min and max opponent Elos
// 2) Compute Fisher information at R_{black}
$\hat{S} \leftarrow \textsc{ExpectedScore}(R_{\text{black}},\ (E_i)_{i=1}^{n})$
$\mathcal{I} \leftarrow \sum_{i=1}^{n} \hat{S}_i (1 - \hat{S}_i)\, (\ln 10/400)^2$
$\text{SE} \leftarrow 1/\sqrt{\mathcal{I}}$
// 3) Adjust for white-advantage and form 95% CI
$R_{\text{true}} \leftarrow R_{\text{black}} + W$
$\text{ME} \leftarrow 1.96 \times \text{SE}$
return $(R_{\text{true}},\ \text{ME})$

note that each time the LLM fails to select a valid move in `make_move`, it is provided a message with the board state in FEN like `Failed to make move: illegal uci: 'd5e4' in 1k3b2/1p2pp1r/p7/3p4/3r4/8/PKb5/8 b - - 3 35`. So note when we change the board state in our ablations, regardless of what we change it to we still always see this FEN when an illegal move is made.

Implementation Details For Always Board State we remove `get_current_board` from the list of actions and instead always provide the board state in the prompt. For Always Legal Moves we do the same but for `get_legal_moves`. For Only `make_move` we remove both `get_current_board` and `get_legal_moves` from the list of actions and instead include the board state and legal moves in the prompt, leaving `make_move` as the only action. This mimics a non-agentic scenario since there is only one action needed in every conversation, so each should only have one turn unless a mistake is made in making a move. We present examples of ASCII and FEN (Forsyth–Edwards Notation) boards below:

Example of ASCII board

```
rnbqkbnr
pppppppp
........
........
.P......
........
P.PPPPPP
RNBQKBNR
```

Example of FEN board

rnbqkbnr/pppppppp/8/8/6P1/8/PPPPPP1P/RNBQKBNR b KQkq - 0 1

For No Legal Moves, we simply remove `get_legal_moves` and replace the unicode board with a FEN board. For Previous Moves, we include all previous moves in an ordered list in UCI notation before the Game Loop Prompt. Here, it is black's turn and there have been 10 full moves and 21 plys:

> **Previous Moves Prompt**
>
> Previous moves (UCI): 1. e2e3 g8f6, 2. a2a4 e7e5, 3. e1e2 b8c6, 4. b1a3 f8e7, 5. a3b1 e5e4, 6. b2b3 e8g8, 7. c1a3 d7d5, 8. g2g4 f6g4, 9. a3d6 e7d6, 10. d1e1 g4e5, 11. b1a3

For Previous Moves + Only `make_move`, we use the Only `make_move` setting but prepend the Previous Moves Prompt in the same way as for Previous Moves.

Analysis Overall, we see for our Actions ablations, performance always increases for both models when we choose to remove actions and include their information in the prompt instead, suggesting that the models still struggle to choose the actions they need in the agentic system.

For Board Representation, we see Grok 3 Mini (low) performance is robust to changes from unicode to ASCII or FEN, while for o4-mini (low) ASCII is 15% better than unicode and FEN is 6.7% better than ASCII. We also see that when the LLM is the white player performance increases as expected, but still remains below 90% for both models.

When Changing Information, we see removing the ability to query for legal moves decreases performance by almost 30% for Grok 3 Mini (low) and almost 10% for o4-mini (low) compared to the FEN baseline. This shows that o4-mini (low) has a better grasp of the legal moves, but both models struggle, as expected. We see that while including previous moves improves the Win/Loss of both models, it also decreases the average Blunder rate (Table 7). In fact, while o4-mini (low) only improves by 3.4% in Win/Loss over the baseline, there is a large drop in blunders of 9.6%, meaning that including previous moves helps the model avoid larger mistakes during play. When including previous moves in the Only `make_move` setting, we see similar but slightly worse performance than in Only `make_move`, suggesting when the model is only focused on making the next move without needing to call other actions for information, the previous moves either don't help or slightly harm performance.

B.2 LLMs with 0% Win/Loss

In Table 4, we include all models we ran with 0% Win/Loss (35 models) versus a random opponent that attempted to complete 30 games. We excluded any games with timeout or API errors. For these models, all losses are due to instruction-following failures with models making too many invalid actions or conversation turns.

B.3 Full Results

For direct comparisons, in the main body we presented results for LLMs vs Random on 30 games. However, to increase the reliability of our evaluation, we ran an increased amount of games on a variety of models. We include results for all games we ran along with the number of games for each result in Table 8. We see that even with more games, the general ranking of models and pattern remains the same: reasoning models perform best, while non-reasoning models struggle to reach over 50% Win/Loss.

B.4 Comparison with Other Reasoning Benchmarks

Large language models excel on standard reasoning benchmarks: for instance, OpenAI's o1 model achieves 11.1 out of 15 (74%) on the AIME with a single sample per problem, 12.5 out of 15 (83%) using self-consistency over 64 samples, and 13.9 out of 15 (93%) after re-ranking 1000 samples via a learned scorer [OpenAI, 2024]. These scores exceed the performance of the majority of AIME participants; for comparison, scoring 10 or above typically places a student in the top 5% of test-takers nationally. On programming contests like Codeforces, o1 attains an Elo of 1258 (62nd percentile) in its preview release and 1673 (89th percentile) in its main version, surpassing most active competitors on the platform. In stark contrast, when evaluated on our interactive chess benchmark, LLMs peak at Elo 758 against an engine calibrated to chess.com, corresponding to a skill level far below that of an average online chess player. This contrast underscores a key insight: while LLMs can exceed the abilities of most humans in math and coding competitions, they exhibit a striking weakness in real-time, multi-step strategic environments like chess. Our benchmark surfaces these limitations by requiring not only domain knowledge but also agentic consistency, planning, and game state awareness.

C Implementation Details

Here we include all prompts supplied to the model, as well as a sample dialog for a single move. Below is the prompt that initiates the conversation with the LLM:

Game Loop Prompt

You are a professional chess player and you play as black. Now is your turn to make a move. Before making a move you can pick one of the following actions:
- ‘`get_current_board`’ to get the schema and current status of the board
- ‘`get_legal_moves`’ to get a UCI formatted list of available moves
- ‘`make_move` <UCI formatted move>’ when you are ready to complete your turn (e.g., ‘`make_move` e2e4’)

Respond with the action.

Reflection prompt for when the chess environment can’t parse an action in the LLM response:

Reflection Prompt for Unparseable Action

Invalid action. Pick one, reply exactly with the name and space delimitted argument: `get_current_board`, `get_legal_moves`, `make_move` <UCI formatted move>

Reflection prompt for illegal move request by LLM:

Reflection Prompt for Illegal Move

Failed to make move: illegal uci: ‘b6c5’ in
`r1bqk2r/pppp1ppp/1N2n1P1/4pKb1/8/P4Q1N/1PPPP1R1/R1B2B2 b kq - 12 25`

Sample dialog of a conversation the LLM has in the agentic environment to output a single move (the board is encoded in Unicode characters and provided to the LLM):

Sample Dialog for Single Move

```
Proxy (to Player_Black):
You are a professional chess player and you play as black. Now is your
turn to make a move. Before making a move you can pick one of the
following actions:
- ‘get_current_board’ to get the schema and current status of the board
- ‘get_legal_moves’ to get a UCI formatted list of available moves
- ‘make_move <UCI formatted move>’ when you are ready to complete your
turn (e.g., ’make_move e2e4’)
Respond with the action.

Player_Black (to Proxy):
get_current_board

Proxy (to Player_Black):
```

```
♖ ♘ ♗ ♕ ♔ ♗ ♘ ♖
♙ ♙ ♙ ♙ ♙ ♙ ♙ ♙
○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○
♟ ♟ ♟ ♟ ♟ ♟ ♟ ♟
♜ ♞ ♝ ♛ ♚ ♝ ♞ ♜

Player_Black (to Proxy):
get_legal_moves

Proxy (to Player_Black):
a7a6, b7b6, c7c6, d7d6, e7e6, f7f6, g7g6, h7h6, a7a5, b7b5, c7c5, d7d5,
e7e5, f7f5, g7g5, h7h5

Player_Black (to Proxy):
make_move e7e5

Proxy (to Player_Black):
Move made, switching player
```

D OpenAI Reasoning Model Timeouts

OpenAI reasoning models exhibited occasional timeout errors at higher levels of reasoning effort. They were the only models we tested that often failed to return a response within the default AG2 client timeout of 10 minutes, throwing the following error:

```
TimeoutError: OpenAI API call timed out. This could be due to congestion
or too small a timeout value. The timeout can be specified by setting
the 'timeout' value (in seconds) in the llm_config (if you are using agents)
or the OpenAIWrapper constructor (if you are using the OpenAIWrapper
directly).
```

In all cases, no retries were made. For random opponents these games were excluded, but against Dragon 1 they were treated as losses for the LLM. As we focus on real-world chess performance, it is reasonable to enforce consistent time limits and thus assigning a loss should a player fail to make a move. We note that these issues are likely due to OpenAI's server or the way it handles high reasoning efforts. Timeout issues are the reason for the lower ranking of some OpenAI reasoning models when tested with higher reasoning efforts.

Increasing the timeout did not solve the issue. We suspect that some of the game prompts triggered failure modes in models, just like some games states and corresponding prompts provoked hallucinated moves in non-reasoning models.

The the statistics on timeout errors observed while testing Dragon 1 vs o3-mini, o3, and o4-mini are in Table 9.

Table 3: API name and settings (e.g., quantization, reasoning effort) mapped to the clean model name used in the paper. If quantized, we ran locally.

API Name and Settings	Cleaned Model Name
gpt-4-0613	GPT-4
qwen2.5-7b-instruct-1m	Qwen2.5-7B-Instruct
internlm3-8b-instruct	InternLM3-8B-Instruct
qwen-max-2025-01-25	Qwen2.5-Max
qwen2.5-14b-instruct@q8_0	Qwen2.5-14B-Instruct (Q8)
qwq-32b	QWQ-32B
o3-2025-04-16-low	o3 (low)
gpt-4o-2024-08-06	GPT-4o (2024-08-06)
mistral-nemo-12b-instruct-2407	Mistral-Nemo-Instruct-2407
gpt-35-turbo-1106	GPT-3.5 Turbo (11/06)
o1-preview-2024-09-12	o1-preview
grok-3-mini-beta-high	Grok 3 Mini (high)
claude-v3-5-sonnet-v1	Claude 3.5 Sonnet
amazon.nova-lite-v1	Amazon Nova Lite
gemini-2.0-flash-exp	Gemini 2.0 Flash (exp)
o4-mini-2025-04-16-low	o4-mini (low)
llama-3-70b-instruct-awq	Llama-3-70B-Instruct
gpt-4.5-preview-2025-02-27	GPT-4.5
deepseek-chat-v3	DeepSeek-V3
gemma-2-27b-it@q6_k_l	Gemma 2 27B
llama3.1-8b	Llama-3.1-8B
claude-v3-5-haiku	Claude 3.5 Haiku
qwen2.5-72b-instruct	Qwen2.5-72B-Instruct
gpt-4.1-nano-2025-04-14	GPT-4.1 Nano
granite-3.1-8b-instruct	Granite-3.1-8B-Instruct
llama3-8b-8192	Llama-3-8B
gemma2-9b-it-groq	Gemma 2 9B
qwen-turbo-2024-11-01	Qwen Turbo
gpt-4o-2024-11-20	GPT-4o (2024-11-20)
amazon.nova-pro-v1	Amazon Nova Pro
o1-2024-12-17-low	o1 (low)
qwen-plus-2025-01-25	Qwen Plus
gpt-35-turbo-0301	GPT-3.5 Turbo (03/01)
mercury-coder-small	Mercury Coder Small
deephermes-3-llama-3-8b-preview@q8	DeepHermes-3-Llama-3-8B-Preview
o4-mini-2025-04-16-high	o4-mini (high)
gpt-4o-mini-2024-07-18	GPT-4o Mini
gpt-4-turbo-2024-04-09	GPT-4 Turbo
o4-mini-2025-04-16-medium	o4-mini (medium)
gemini-2.5-pro-preview-03-25	Gemini 2.5 Pro Preview
gpt-4-32k-0613	GPT-4 32K
phi-4	Phi-4
gemini-2.0-flash-thinking-exp-1219	Gemini 2.0 Flash Thinking
mistral-small-instruct-2409	Mistral-Small-Instruct-2409
mistral-small-24b-instruct-2501@q4_k_m	Mistral-Small-24B-Instruct-2501
llama-2-7b-chat	Llama-2-7B-Chat
gemma-3-12b-it@iq4_xs	Gemma 3 12B (iq4)
claude-v3-7-sonnet-thinking_10000	Claude 3.7 Sonnet Thinking
gemini-1.5-flash-001	Gemini 1.5 Flash
deepseek-chat-v3-0324	DeepSeek-V3 (0324)
deepseek-reasoner-r1	Deepseek-R1
llama-4-scout-cerebras	Llama 4 Scout
chat-bison-32k@002	Chat-Bison-32K
qwen2.5-14b-instruct-1m	Qwen2.5-14B-Instruct
o1-2024-12-17-medium	o1 (medium)
claude-v3-haiku	Claude 3 Haiku
grok-3-mini-beta-low	Grok-3 Mini (low)
o3-mini-2025-01-31-low	o3-mini (low)
llama-3.1-tulu-3-8b@q8_0	Llama-3.1-Tulu-3-8B
gpt-4o-2024-05-13	GPT-4o (2024-05-13)
gpt-35-turbo-0125	GPT-3.5 Turbo (01/25)
claude-v3-7-sonnet	Claude 3.7 Sonnet
gemma-2-9b-it-8bit	Gemma 2 9B (8bit)
gpt-35-turbo-0613	GPT-3.5 Turbo (06/13)
gemini-2.0-flash-lite-preview-02-05	Gemini 2.0 Flash Lite (preview)
o3-mini-2025-01-31-medium	o3-mini (medium)
gpt-4.1-2025-04-14	GPT-4.1
gemini-2.0-flash-lite-001	Gemini 2.0 Flash Lite
o3-2025-04-16-medium	o3 (medium)
gemini-2.0-flash-001	Gemini 2.0 Flash
deepseek-r1-distill-qwen-14b@q8_0	DeepSeek-R1-Distill-Qwen-14B
ministral-8b-instruct-2410	Mistral 8B Instruct
deepseek-r1-distill-qwen-32b@q4_k_m	DeepSeek-R1-Distill-Qwen-32B
llama-3.3-70b	Llama-3.3-70B
grok-2-1212	Grok-2
gemma-3-12b-it@q8_0	Gemma 3 12B (q8)
gemma-3-27b-it@iq4_xs	Gemma 3 27B
claude-v3-5-sonnet-v2	Claude 3.5 Sonnet v2
gpt-4.1-mini-2025-04-14	GPT-4.1 Mini

Table 4: LLMs with a 0% Win/Loss on 30 games along with the reasons for their losses. Note that none of these models were able to complete games but instead always lost due to instruction-following failures. Reasoning models are shaded.

Model	Too Many Wrong Actions	Max Turns
Amazon Nova Lite	76.7	23.3
Amazon Nova Pro	100.0	0.0
Claude 3 Haiku	10.0	90.0
Chat-Bison-32K	100.0	0.0
DeepHermes-3-Llama-3-8B-Preview	96.7	3.3
DeepSeek-R1-Distill-Qwen-14B	100.0	0.0
DeepSeek-R1-Distill-Qwen-32B	73.3	26.7
Gemini 2.0 Flash Lite (preview)	100.0	0.0
Gemini 2.0 Flash Thinking	100.0	0.0
Gemma 2 9B	100.0	0.0
Gemma 3 12B (iq4)	100.0	0.0
Gemma 3 12B (q8)	100.0	0.0
Gemma 3 27B	100.0	0.0
GPT-3.5 Turbo (01/25)	100.0	0.0
GPT-3.5 Turbo (03/01)	100.0	0.0
GPT-3.5 Turbo (06/13)	100.0	0.0
GPT-3.5 Turbo (11/06)	100.0	0.0
GPT-4.1 Nano	100.0	0.0
Granite-3.1-8B-Instruct	60.0	40.0
InternLM3-8B-Instruct	60.0	40.0
Llama-2-7B-Chat	100.0	0.0
Llama-3.1-Tulu-3-8B	23.3	76.7
Llama-3-8B	90.0	10.0
Llama-3.1-8B	80.0	20.0
Mercury Coder Small	100.0	0.0
Mistral 8B Instruct	100.0	0.0
Mistral-Nemo-Instruct-2407	100.0	0.0
Mistral-Small-24B-Instruct-2501	100.0	0.0
Mistral-Small-Instruct-2409	100.0	0.0
Phi-4	100.0	0.0
Qwen Turbo	100.0	0.0
Qwen2.5-14B-Instruct	70.0	30.0
Qwen2.5-14B-Instruct (Q8)	96.7	3.3
Qwen2.5-7B-Instruct	100.0	0.0
QWQ-32B	93.3	6.7

Table 5: Total number of games played against each skill along with Win/Loss for all games playing against that skill.

Model	Skill	Total Games	Win/Loss
o3 (low)	1	33	81.8
	2	33	72.7
	3	33	75.8
	4	33	68.2
	5	32	71.9
	10	33	3.0
Grok 3 Mini (high)	1	33	51.5
	2	34	48.5
	3	34	41.2
	4	34	38.2
	5	34	25.0
o4-mini (high)	1	27	61.1
	2	22	56.8
o3-mini (high)	1	31	67.7
	2	26	57.7
o4-mini (medium)	1	40	53.8
o3-mini (medium)	1	38	39.5
o4-mini (low)	1	33	30.3
o3-mini (low)	1	33	10.6

Table 6: Win/Loss on ablations. LLM CHESS is the baseline.

Setting	Grok 3 Mini (low)	o4-mini (low)
LLM CHESS	61.7	73.3
Actions		
Always Board State	66.7	83.3
Always Legal Moves	68.3	93.3
Only `make_move`	71.7	**96.7**
Board Representation		
ASCII	63.3	88.3
FEN	63.3	95.0
LLM as White	**78.3**	83.3
Changing Information		
No Legal Moves	36.7	86.7
Previous Moves	75.0	76.7
Previous Moves + Only `make_move`	66.7	95.0

Table 7: Average Blunder rate (%) per ply when including previous moves vs baseline. Lower is better.

Model	LLM CHESS	Previous Moves
Grok 3 Mini (low)	9.1	**3.5**
o4-mini (low)	11.2	**1.6**

Table 8: Full results for LLM vs. Random on variable number of ≥ 30 games. Reasoning models are shaded. The percentage of games ending due to checkmate from either side, instruction-following failures, and draws are also displayed.

Player	Total Games	Win/Loss	Checkmate	Instruction		Draws			
			Checkmate	Wrong Actions	Max Turns	Stalemate	Insuff. Material	5x Repetition	Max Moves
o3 (medium)	48	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0
o3 (low)	41	96.3	92.7	0.0	0.0	0.0	0.0	2.4	4.9
o4-mini (high)	38	96.1	92.1	0.0	0.0	5.3	2.6	0.0	0.0
o1 (medium)	40	91.2	82.5	0.0	0.0	10.0	2.5	0.0	5.0
Grok 3 Mini (high)	44	86.4	72.7	0.0	0.0	4.5	4.5	0.0	18.2
o4-mini (medium)	159	84.3	68.6	0.0	0.0	11.9	12.6	0.0	6.9
o1 (low)	47	78.7	57.4	0.0	0.0	6.4	19.1	0.0	17.0
o4-mini (low)	74	70.9	44.6	0.0	0.0	17.6	9.5	0.0	28.4
o1-preview	30	68.3	46.7	10.0	0.0	3.3	20.0	0.0	20.0
o3-mini (medium)	44	67.0	36.4	2.3	0.0	20.5	4.5	0.0	36.4
Claude 3.7 Sonnet Thinking	37	62.2	24.3	0.0	0.0	0.0	18.9	0.0	56.8
Grok 3 Mini (low)	52	58.7	21.2	0.0	0.0	13.5	1.9	0.0	63.5
Gemini 2.5 Pro Preview	33	53.0	36.4	27.3	3.0	15.2	9.1	0.0	9.1
GPT-4 32K	33	48.5	3.0	0.0	0.0	0.0	0.0	0.0	97.0
Qwen2.5-Max	60	48.3	3.3	0.0	0.0	0.0	0.0	0.0	96.7
GPT-4o (2024-11-20)	71	47.9	12.7	0.0	0.0	0.0	0.0	0.0	87.3
Claude 3.5 Sonnet v2	60	47.5	8.3	3.3	0.0	1.7	0.0	0.0	86.7
Claude 3.5 Sonnet	60	46.7	18.3	1.7	0.0	0.0	0.0	0.0	80.0
GPT-4 Turbo	30	46.7	6.7	0.0	0.0	0.0	0.0	0.0	93.3
GPT-4.5	44	46.6	6.8	0.0	0.0	0.0	0.0	2.3	90.9
GPT-4	33	45.5	9.1	0.0	0.0	0.0	0.0	0.0	90.9
GPT-4o (2024-08-06)	59	44.1	15.3	0.0	0.0	1.7	0.0	0.0	83.1
GPT-4.1	80	43.8	13.8	1.2	0.0	0.0	0.0	0.0	85.0
Claude 3.5 Haiku	42	42.9	7.1	2.4	4.8	2.4	0.0	0.0	83.3
Claude 3.7 Sonnet	42	40.5	16.7	11.9	0.0	2.4	0.0	0.0	69.0
GPT-4o (2024-05-13)	60	40.0	11.7	8.3	0.0	0.0	0.0	0.0	80.0
o3-mini (low)	56	37.5	7.1	19.6	8.9	3.6	0.0	0.0	60.7
Deepseek-R1	31	32.3	22.6	51.6	6.5	3.2	9.7	0.0	6.5
GPT-4.1 Mini	84	30.4	9.5	3.6	26.2	0.0	0.0	0.0	60.7
GPT-4o Mini	30	30.0	3.3	36.7	0.0	0.0	0.0	0.0	60.0
Llama-3-70B-Instruct	30	25.0	3.3	46.7	0.0	0.0	0.0	0.0	50.0
Gemini 2.0 Flash	67	21.6	10.4	55.2	0.0	0.0	0.0	0.0	34.3
Grok-2	49	19.4	6.1	63.3	0.0	0.0	0.0	0.0	30.6
Gemini 1.5 Flash	30	16.7	6.7	60.0	0.0	0.0	0.0	0.0	33.3
Gemma 2 27B	30	13.3	6.7	66.7	0.0	0.0	0.0	0.0	26.7
Llama 4 Scout	39	10.3	2.6	64.1	12.8	0.0	0.0	0.0	20.5
Gemma 2 9B (8bit)	30	6.7	3.3	83.3	0.0	0.0	0.0	0.0	13.3
DeepSeek-V3 (0324)	45	5.6	2.2	88.9	2.2	0.0	0.0	0.0	6.7
Llama-3.3-70B	42	4.8	9.5	73.8	7.1	0.0	0.0	0.0	9.5
Qwen Plus	33	4.5	0.0	90.9	0.0	0.0	0.0	0.0	9.1
Gemini 2.0 Flash (exp)	30	3.3	0.0	90.0	3.3	0.0	0.0	0.0	6.7
Qwen2.5-72B-Instruct	30	3.3	3.3	90.0	0.0	0.0	0.0	0.0	6.7
Gemini 2.0 Flash Lite	66	1.5	4.5	95.5	0.0	0.0	0.0	0.0	0.0
DeepSeek-V3	70	1.4	1.4	90.0	5.7	0.0	0.0	0.0	2.9

Table 9: Number of timeout errors in OpenAI reasoning models when facing Dragon 1 opponents with varying skill levels. The default timeout is 10 minutes.

Opponent Skill Level	LLM	Total logs	Errors
1	o3 (low)	33	0
1	o3-mini (low)	33	0
1	o3-mini (medium)	38	0
1	o3-mini (high)	33	2
1	o4-mini (low)	33	0
1	o4-mini (medium)	40	0
1	o4-mini (high)	33	6
2	o3 (low)	33	0
2	o3-mini (high)	30	4
2	o4-mini (high)	30	8
2	o4-mini (high) w/ 20m timeout	29	7
2	o4-mini (high) w/ 60m timeout	6	4
3	o3 (low)	33	0
4	o3 (low)	33	0
5	o3 (low)	35	0
10	o3 (low)	33	0
10	o3 (medium) w/ 60m timeout	11	2

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's key contributions. Introducing the LLM CHESS agentic benchmarking framework, defining a comprehensive suite of per-model, per-game, and per-ply metrics, and presenting evaluation results over 50+ LLMs.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We dedicate Section 6 to discuss limitations, such as the 100-move cap, prompt length constraints, and timeout issues. We reflect on how these factors influence the robustness and generalizability of our benchmark

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All formulas underlying our Elo estimation and win-probability models are fully stated with their assumptions in Section 2.2, numbered and cross-referenced, and will becomplemented by a complete derivation and Fisher information proof in Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide full details on model configurations (temperature, Top-P), data splits, evaluation scripts, and hyperparameters in Section 3 and Appendix A, and will release our code and logs to enable exact reproduction of all experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code scripts and evaluation logs will be made publicly available on GitHub, and the full datasets of game records and metrics will be hosted on Hugging Face, with detailed instructions for downloading and reproducing every experiment provided in the supplemental material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3.1 and Appendix A show all the evaluation settings, including temperature (0.3), Top-P (1.0), instruction-following timeouts, game caps (100 moves, 10 turns per ply), and engine skill levels, alongside script commands and hyperparameter values needed to reproduce every experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report 95 % confidence intervals for Elo estimates (via Fisher information) in Figure 3 and include standard deviations for centipawn and Win % metrics across games, with all error-bar definitions and calculation methods to be detailed in Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Our paper does not specify the compute infrastructure (e.g., CPU/GPU types, number of workers, memory requirements, or wall-clock runtimes) used for running LLM evaluation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work involves automated evaluations of pretrained models on synthetic chess games and does not involve human participants, personal data, or activities that raise ethical concerns under the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss positive impacts, providing a rigorous, extensible benchmark to improve LLM reasoning robustness and drive safer AI development, as well as potential negative uses, such as adversarial fine-tuning of LLMs for game-theoretic exploits or misuse in generating deceptive, rule-based content, in Section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work evaluates existing LLMs on synthetic chess games and does not release any new pretrained models or scraped datasets that pose misuse risks, so no additional safeguards are required.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third-party code, data, and models—including Komodo's Dragon engine, Lichess metrics code, and publicly available PGN data—are explicitly cited with URLs, and their respective licenses and terms of use are noted in our bibliography and supplemental material.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We publish the full LLM CHESS game dataset and the evaluation framework's code on GitHub (including README, usage examples, and license) and Hugging Face (with dataset schema, download instructions, and versioning), ensuring all new assets are thoroughly documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our study only evaluates pre-trained models on synthetic chess interactions and does not involve any human subjects or crowdsourced data collection.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our study only evaluates pre-trained models on synthetic chess interactions and does not involve any human subjects or crowdsourced data collection.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core methodology revolves around playing chess with LLMs, which is standard practice for model evaluations in this domain and does not involve any non-standard or opaque use of LLMs that would require a separate declaration.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.