COMPARING LLM PERFORMANCE ON CHESS VARIANTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, frontier LLMs have achieved rapid capability gains on reasoning-intensive benchmarks including math and coding. However, recent work has indicated that many of those capabilities may come from memorizing expanded training data rather than general gains in reasoning ability. To assess this possibility, we use chess as a testbed and test the performance of LLMs on a variety of non-standard *chess variants*, which are less represented in the training data than standard chess, against the chess variant engine Fairy Stockfish. We find that standard chess performance is strong in the opening moves and degrades in the midgame, while performance on chess variants remains weak throughout the game, providing support for the claim that memorization/overfitting to the training distribution accounts for a sizable portion of the high performance on reasoning benchmarks, which is further supported by our observation that LLMs perform better when given input chess moves in algebraic notation as compared to FEN. We also test a variety of other settings and find that enabling long chain-of-thought offers only marginal performance improvement in the midgame, and play against a "human-like" opponent or a non-human-like opponent of the same skill does not significantly affect performance. Overall, LLM performance is strongest in the settings most commonly represented in training data, and weaker on less represented settings.

1 INTRODUCTION

Recently, frontier LLMs have achieved rapid capability gains on reasoning-intensive benchmarks including math and coding. In September alone, major AI companies have created models that have achieved 97.6% on FrontierMath Tier 4 or 60.9% on Terminal-Bench-4.0 (OpenAI, 2026a; Anthropic, 2026). Furthermore, swarms of LLM agents in recent months have purportedly solved Millennium Prize Problems and caused large cybersecurity incidents (OpenAI, 2026b;c). As investment continues, major AI companies plan to continue expanding their models' capabilities.

However, recent academic work has indicated that many of those capabilities may come from memorizing expanded training data rather than general gains in reasoning ability. In several domains, making small modifications to tasks sharply decreases LLM performance on those tasks (Wu et al., 2024; Lewis & Mitchell, 2024). Furthermore, major AI companies are spending heavily to acquire high-quality data and reinforcement learning environments (AJ & Patel, 2026), suggesting that these are critical to continued growth in LLM capabilities.

To gain further insight into the degree of reasoning generalization versus data memorization in frontier models, we use chess as a testbed to compare the performance of LLMs on standard chess to their performance on non-standard chess variants, which are less represented in the training data. These tests are enabled by Fairy Stockfish (Fichter & Contributors, 2021), a chess variant engine that allows nearly arbitrary reconfiguration of movement rules, piece types, and goals on a chessboard while retaining the strong performance of Stockfish. As a result, we compare variants at a variety of memorization levels (Figure 1), ranging from standard chess with lots of representative training data, to Chess960 with a lesser but still notable amount of training data, to three novel variants with no training data to the best of our knowledge.

We find that standard chess performance is strong in the opening moves and degrades in the midgame, while performance on chess variants remains weak throughout the game. We also com-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

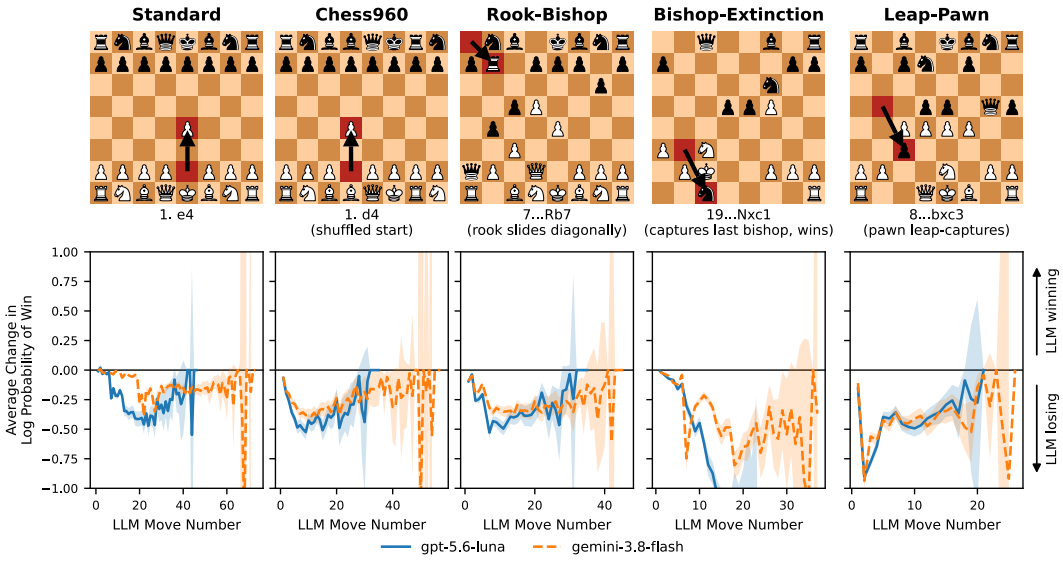


Figure 1: **(top)** Example of a move for each chess variant tested. Chess960 shuffles the first and last ranks; rook-bishop swaps the rook and bishop movement patterns; bishop-extinction sets the goal as capturing both bishops; leap-pawn only allows pawns to move and capture two spaces forwards. **(bottom)** Average change in the logarithm of the estimated LLM win probability for each move over the course of gameplay.

pare performance when prompted with the more common algebraic notation versus the less common Forsyth-Edwards Notation (FEN), and find that performance on FEN is slightly worse. Furthermore, we evaluate LLM performance against a chess engine trained to replicate human performance at a variety of skill levels (Monroe et al., 2026), as well as against a less "human-like" opponent whose moves are given by a mixture of Fairy Stockfish and a completely random baseline calibrated to perform at those same skill levels. We observe only slight differences in performance against opponents at the same skill level regardless of play style, and observe stronger play against weaker opponents. Finally, we evaluate the performance of LLMs at various reasoning levels. Overall, LLM performance is strongest in the settings most commonly represented in training data, and weaker on less represented settings.

2 Related Work

Counterfactual tasks for evaluating generalization: Recent work has explored a "counterfactual tasks" approach to evaluating generalization in LLMs: comparing success on a standard task to success on a variant of that task. Initial work in this line of research examined, among other tasks, standard chess versus Chess960 on opening legality, but did not examine full strategic play (Wu et al., 2024). Other work examines performance on counterfactual versions of string analogies (Lewis & Mitchell, 2024) and esoteric programming languages (Sharma & Chopra, 2026). Pleiss et al. (2026) use more common and less common positions in chess to compare generalization and memorization, finding that performance is worse in less common positions, but acknowledges that synthetically generated positions may have biased the results.

Chess-trained transformers: Other work assesses the performance of transformers specifically trained on chess. Toshniwal et al. (2022) find that even small transformers are capable of tracking board state and predicting legal moves. Ruoss et al. (2024) find that by training on data from Stockfish, transformers can learn highly accurate action-values in chess. The representation learned by transformers can also be intervened on to modify its internal board state or improve win rate (Karvonen, 2024). It is possible to improve chess transformer performance by using Monte Carlo tree search (MCTS), either by training the model to perform MCTS or using the model as input into a symbolic MCTS, and this approach appears to succeed in both chess and Chess960 (Schultz et al.,

2025). Finally, Mészáros et al. (2026) compare performance of a specially trained chess transformer to performance on Chess960 and Horde Chess, finding improved but still imperfect generalization.

LLMs on chess: Further work also examines the performance of LLMs on standard chess. LLMs allow both chess game datasets and natural language discussions of chess to be used to improve performance, and there exist large amounts of both in common training datasets (Feng et al., 2023). Interestingly, for the first two years after the release of ChatGPT to the public, it appeared that only `gpt-3.5-turbo-instruct` had reasonably strong chess performance, possibly due to curated data or a reduction in performance due to post-training (Acher, 2023; Dynomight, 2024a;b). Previous work has found that models as of late 2025 still struggle with questions regarding chess understanding (Wen et al., 2025), and that models often struggle against weak engines or even random play (Kolasani et al., 2025; Liu et al., 2026).

LLMs on novel games: Previous work has also examined the strength of LLMs on novel games. In a similar vein to the counterfactual-tasks line of work, Costarelli et al. (2024) evaluate LLMs on games unlikely to be in the training data and finds weak performance, though no comparison is made to similar games available in the training data. Paglieri et al. (2025) finds broad weakness in LLM performance in a number of video game settings. LLMs achieve low but nontrivial win rates on games with descriptions generated by LLMs; however, LLM presence in the generation loop may bias towards games with known mechanics (Verma et al., 2025). In non-adversarial games such as an arithmetic card game, supervised fine-tuning does not improve generalization while reinforcement learning does (Chu et al., 2025). Additionally, LLMs struggle to generalize rule-following and strong play to variants of games such as Tic-Tac-Toe and Solitaire, though fine-tuning on a broad dataset helps (Mishra et al., 2025; Bateni et al., 2025). LLMs also perform poorly on Baba Is You, a complex rule manipulation game that requires strong generalization of rule-following (van Wetten et al., 2025).

3 METHODS

3.1 FAIRY STOCKFISH

In order to create a controlled evaluation of LLMs playing chess variants, we need some automated opponent engine with controllable difficulty parameters. Fairy Stockfish v14.0.1 is a chess engine capable of handling highly arbitrary chess variants (Fichter & Contributors, 2021).[1] We also used a consistent search depth of 16 when running our experiments in order to maximize throughput while achieving an accurate evaluation of each position.

3.2 PROMPTS

The rule changes in any given variant needed to be specified to the model in order to clarify the available action space and strategies. We therefore gave the task specification to each model in the system prompt, including instructions regarding any rule changes between standard chess and the current variant. Aside from one setting with direct prompting, models were also asked to reason step by step while keeping reasoning brief and minimizing loops, in order to promote more effective reasoning while reducing costs. System prompts appear in Appendix A for each setting. Each model was then given the history of the game in algebraic notation (eg "1. e4 e5") as a user prompt. If the model made an illegal, invalid, or unparseable move, it was prompted with the failure reasoning and told to retry in a user message. User messages appear in Appendix B.

In order to prevent a model from continually making failed moves, we capped the number of potential retries at some R, then considered any model that exceeded this retry budget to have lost by *disqualification*. We refer to the percentage of games that are disqualified in a given setting as the *disqualification rate*. We calibrated R uniquely for each model as the value at which the disqualification rate stopped meaningfully decreasing. This was done so that the token cost of each experiment did not increase arbitrarily (if R is set too large) but that the model was not prematurely disqualified before it could produce a valid move (if R is set too small). The results of this proce-

[1] We chose not to utilize the Efficiently Updateable Neural Network (NNUE) component of Fairy Stockfish, since it must be retrained for every chess variant, and the non-neural search component of the engine was sufficient for strong gameplay performance.

dure can be found in Section 4.1. We subsequently used this value of R for all further settings with a given model.

After a successful move, all prior user and assistant messages were cleared to reduce input token costs. Each setting ran for 1000 full games, with each game from the starting position to checkmate, stalemate, or disqualification, and the seed of the model query was set to $42 + (i - 1)$ for each game, where i is the 1-indexed game number.

3.3 VARIANTS

In order to test several different aspects of chess for memorization, each chess variant was chosen to produce a different axis of variation. **Chess960** is a common chess variant that shuffles the pieces on the first and last ranks of the board at the start of each game, and there are game corpora for Chess960 online (Lichess.org, 2026), though in lesser quantities than standard chess. We consider three additional chess variants for which there are no existing databases to the best of our knowledge, and, for the latter two, for which there is no existing evidence to the best of our knowledge:

- **Bishop extinction:** changes the goal from checkmating the king to capturing both bishops, and our version of bishop extinction also removes the requirement to move the king out of check. While extinction chess has been described online, we found no databases of games or websites with extinction chess gameplay.
- **Rook-bishop:** swaps the movement patterns of rooks and bishops, while leaving rooks as the castling pieces.
- **Leap-pawn:** restricts pawns to always move two squares forward and to capture two squares forward and one square to the side, with no en passant allowed.[2]

All three of the variants we created retain the initial positions of pieces from standard chess. A demonstration of the rules of each variant can be found in Appendix C, and full configurations for each variant can be found in Appendix D.

3.4 MAIA AND MIXED ENGINES

Additionally, we hypothesized that whether or not play was "human-like", and therefore more or less represented in the training data, might affect model strength if memorization was present. We therefore created settings where the models played against `Maia-3-79M`, a chess engine trained to replicate human play at various Elo levels and styles (Monroe et al., 2026). We test the models against MAIA at Elo levels of 600, 1600, and 2600.

As our less "human-like" engine, we created MIXED engines, which would play randomly with probability p, and would otherwise play the move recommended by Fairy Stockfish. Since the MIXED engines randomly choose between playing randomly and playing near-optimally, we considered them unlikely to play in a human-like fashion. To calibrate p for each Elo value, we used the probabilistic bisection algorithm (PBA), which is capable of finding the root of a function with access to only a noisy oracle (Waeber, 2013; Kundert, 2022). We apply PBA to the function $f(r) = 2(r - 0.5)$, where $r = 1$ if the MIXED engine wins, $r = 0$ if the MAIA engine wins, and $r = 0.5$ if engines draw. We then run PBA with this function over 1000 iterations, with the MAIA engine set to Elo e, in order to determine the value of p such that the MIXED agent has the same Elo e. Details on our mixing probabilities and evaluation of the calibration can be found in Appendix E.

Additionally, we evaluated the model against a fully random baseline. Since the MAIA engine and the random choices are non-deterministic, we set their random seed to $42 + (i - 1)$ as well.

3.5 SCORING

Lastly, evaluating models on move quality throughout each game requires a scoring metric that is robust to chess variant, opponent strength, and move number. We use Fairy Stockfish to evaluate, at

[2]Due to the limitations of the variant description language provided by Fairy Stockfish, pawns may only promote to queens in this variant.

each position, how likely each player is to win, assuming that both players play perfectly for the rest of the game. Details of this calculation can be found in Appendix F. We then take the difference of the logarithms of win probability before and after each LLM move as our evaluation metric. This allows us to isolate the quality of LLM moves without simultaneously measuring opponent strength or increasing the weight placed on moves made while the game is more competitive. Finally, we also report the average of this metric per move as a measure of overall gameplay performance.

4 EXPERIMENTS

We begin in Section 4.1 by determining the maximum number of retry attempt R for each model, finding that `Gemini-3.8-Flash` requires 2 retries and `GPT-5.6-Luna` requires 6 retries (any additional retries do not significantly change performance). In Section 4.2, we evaluate whether models performed better with algebraic notation or FEN.

In Section 4.3 we test LLM performance on the chess variants. We evaluate whether disabling or increasing reasoning would reduce or improve model performance in Section 4.4, and found that neither had much effect. Next, in Section 4.5, we evaluate `Gemini-3.8-Flash` against MAIA and MIXED engines, as well as a random baseline. Finally, in Section 4.6, we evaluate whether or not providing the conversation history to the model had an effect on performance. In some sections, we also discuss game outcomes; more detailed charts of these outcomes are found in Appendix G

4.1 LLMS REQUIRE SEVERAL RETRIES

Disqualification rates as R increases are shown in Figure 2. While `Gemini-3.8-Flash` initially had a disqualification rate of 32.1% when given 0 retries, the disqualification rate quickly fell to 1.9% at 1 retry, then 0.0% at 2 retries. Since the disqualification rate had reached its minimum value at 2 retries, we consider it no longer capable of meaningfully improving, and therefore select $R = 2$ for `Gemini-3.8-Flash`.

Meanwhile, `GPT-5.6-Luna` had a disqualification rate of 93.2% when given 0 retries. The disqualification rate declined over the course of the next several retries, eventually reaching 8.2% at 6 retries and 8.0% at 7 retries. Since this small decline was well within the margin of error, we consider it unlikely to meaningfully improve further, and therefore select $R = 6$ for `GPT-5.6-Luna`.

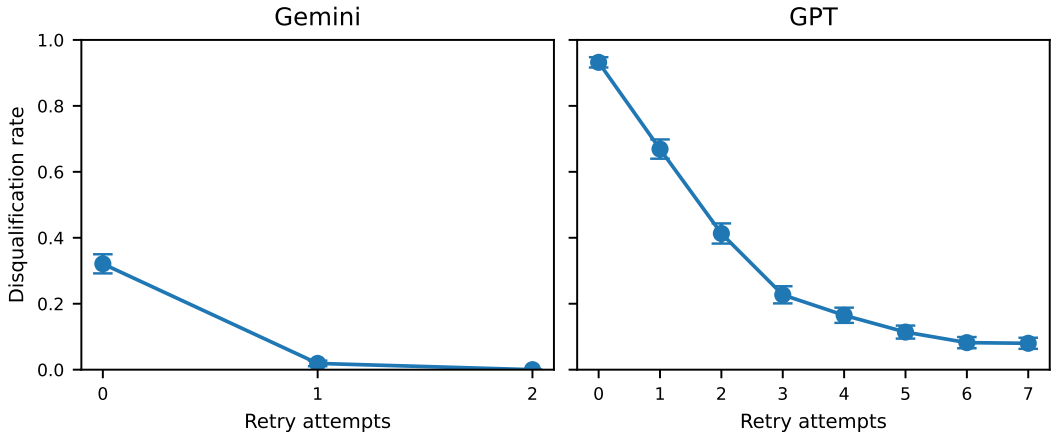


Figure 2: Disqualification rates due to repeated failed moves vs number of retry attempts the model was allowed (R).

4.2 LLMS PLAY SLIGHTLY BETTER WITH ALGEBRAIC NOTATION

Due to cost limitations, we only evaluate `Gemini-3.8-Flash` in this experiment. This experiment compares algebraic notation, a move-by-move presentation of a chess game (eg "1. e4 e5"), to FEN, a full representation of the current board state (without move history). Average move quality over

the course of play in both settings is shown in Figure 3. While performance in the opening remains strong in the FEN setting, the precipitous decline in performance occurs in move 17 rather than move 20. However, performance converges in both the algebraic and FEN settings once performance in the algebraic notation setting declines. Disqualification rates remain roughly equivalent between the two notations (0.5% for algebraic notation vs 0.0% for FEN). Since the models generally performed better with algebraic notation, we only use algebraic notation for all other experiments.

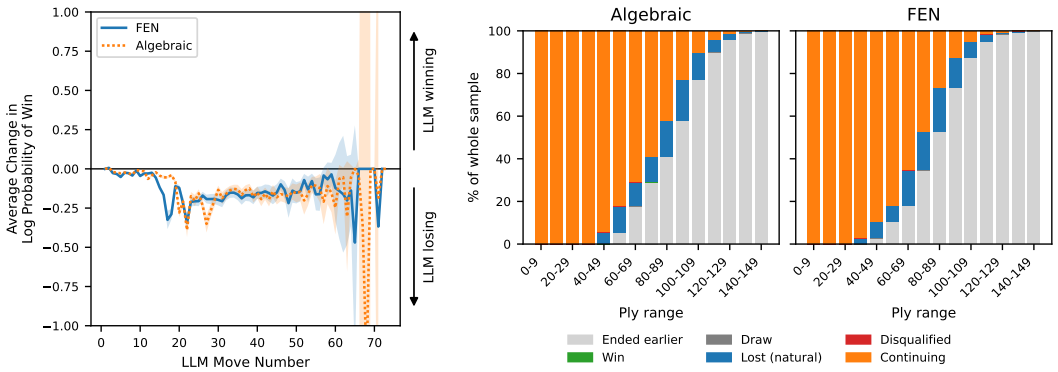


Figure 3: **(left)** The logarithm of the estimated LLM win probability for each move when using algebraic notation and FEN notation. **(right)** Outcomes in each setting binned by ply at which the outcome occurred.

4.3 LLMs Play Worse on Chess Variants

Average move quality over the course of play in each variant is shown in Figure 1.

Chess960 Both `Gemini-3.8-Flash` and `GPT-5.6-Luna` performed strongly in standard chess during the opening moves, but their performance decreased precipitously in move quality in the midgame (around move 20 for `Gemini-3.8-Flash` and move 7 for `GPT-5.6-Luna`). In Chess960, both models produced worse move quality in the opening than on standard chess, but had converged to standard chess performance by the midgame.

Bishop-extinction In bishop extinction, both models begin with a move quality similar to (but slightly less than) standard chess, but then have earlier and larger decreases in move quality as the midgame approaches. While `Gemini-3.8-Flash` remains at a roughly consistent but poor quality of play after the initial decrease, `GPT-5.6-Luna` continues to decline until all games in the bishop extinction setting are completed. This pattern may be explained by bishop extinction altering the goal but not the moves of play, making the optimal moves more similar to standard chess in the opening but more divergent by the midgame.

Rook-bishop In rook-bishop, the converse occurs: both models begin with a significantly worse move quality compared to standard chess, but converge to standard chess performance in the midgame. One possible explanation for this is the similarity of rook-bishop to Chess960: while the now-diagonally-moving rooks still castle, the variant otherwise resembles a rearrangement of the first and last ranks.

Leap-pawn In leap-pawn, both models begin and end the game with poor move quality. Some increase in performance on average is apparent towards the end of the sample available for leap-pawn, but the low sample size suggests that this increase could be due to noise or survivorship bias.

Disqualification Disqualification rates on most variants are inconsistent between `Gemini-3.8-Flash` and `GPT-5.6-Luna`. With `Gemini-3.8-Flash`, the disqualification rate rises significantly on Chess960 and rook-bishop, while remaining suppressed in all other variants. In contrast, `GPT-5.6-Luna`'s disqualification rate decreases on all variants, though this is

due to most games in variant settings not reaching the move where most disqualifications occurred in the standard setting.

4.4 Reasoning Effort Does Not Significantly Affect LLM Chess Skill

Similarly to our other experiments, we only ran the adjusted reasoning level settings on `Gemini-3.8-Flash`. This model had three options for reasoning: "low", "medium", and "high". Average move quality over the course of play is shown in Figure 4. We find that while the decline in performance from the opening to the midgame is slightly less sharp at higher reasoning levels, there is little difference between performance at any point in the game otherwise. Furthermore, while both medium and high reasoning eliminate the 0.5% disqualification rate remaining from low reasoning, this decline is unlikely to be meaningful given how small the disqualification rate already was.

`Gemini-3.8-Flash` does not have the option to disable reasoning, so we evaluate the no-reasoning settings using `Gemini-3.6-Flash`, the most recent Gemini model with the ability to disable reasoning. We evaluate three settings: "low" reasoning with chain-of-thought (CoT) prompting, reasoning disabled with CoT prompting, and reasoning disabled with direct prompting. We find that while the differences in performance are small between these settings, disabling reasoning increases disqualification rate, especially in the CoT prompting setting.

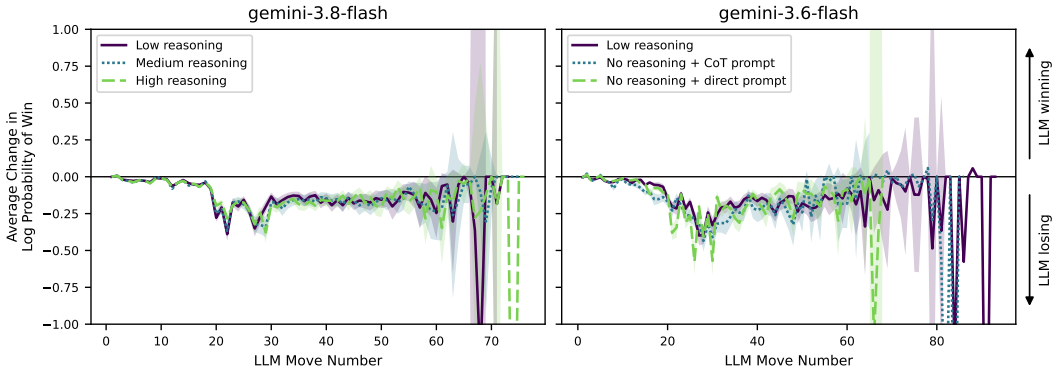


Figure 4: **(left)** The logarithm of the estimated LLM win probability for each move when using "low", "medium", and "high" reasoning on `Gemini-3.8-Flash`. **(right)** The logarithm of the estimated LLM win probability for each move when using "low" reasoning, no reasoning with CoT prompting, or no reasoning with direct prompting on `Gemini-3.6-Flash`.

4.5 Similarity of Play to Humans Does Not Affect LLM Performance

As previously, the only model we test in this experiment is `Gemini-3.8-Flash` due to cost considerations. Average move quality over the course of play is shown in Figure 5. Play against both Maia and Mixed opponents (described in Section 3.4) had higher move quality against weaker opponents, and in both cases, gameplay performance against the Elo 2600 declined earlier but remained stronger and more gradual than against the standard Fairy Stockfish opponent used in all other settings in this paper. There was no large difference between Maia and Mixed opponents in the Elo 600 or Elo 1600 settings, though play against the Maia opponent was slightly worse than play against the Mixed opponent in the Elo 2600 setting. Against a fully random opponent, `Gemini-3.8-Flash` retained strong move quality throughout the whole game. In all settings in this experiment, Gemini had a win rate of at least 60%, notable as the only settings where an LLM won any games at all. Disqualification rates also remained low for this experiment.

4.6 Conversation Persistence Does Not Affect LLM Performance

Once again, we evaluate this experiment on only `Gemini-3.8-Flash` due to cost considerations. In the persistence setting, we avoided clearing messages between moves, allowing the model to observe its previous messages and any strategy information it placed in its response. Average move

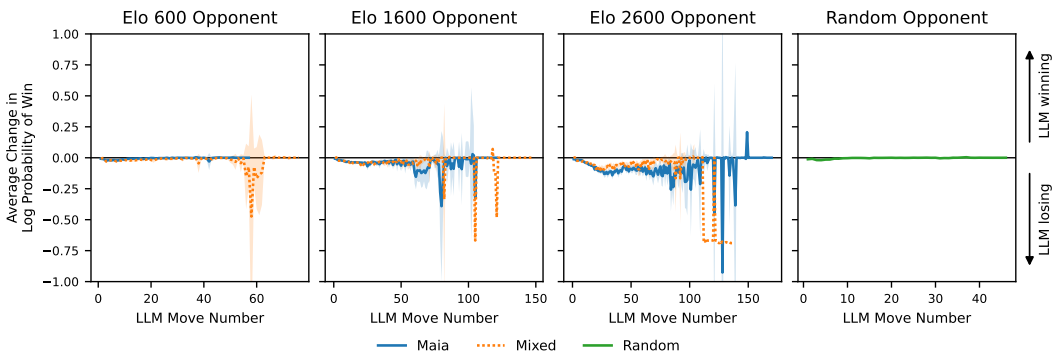


Figure 5: The logarithm of the estimated LLM win probability for each move with MAIA, MIXED, and RANDOM opponents at varying Elo.

quality over the course of play in each condition is shown in Figure 6. This had no effect during the opening and endgame. However, retaining information between moves has a slight negative effect on performance in the midgame. Disqualification rates remain roughly consistent between the settings (0.5% without move persistence vs 0.8% with move persistence).

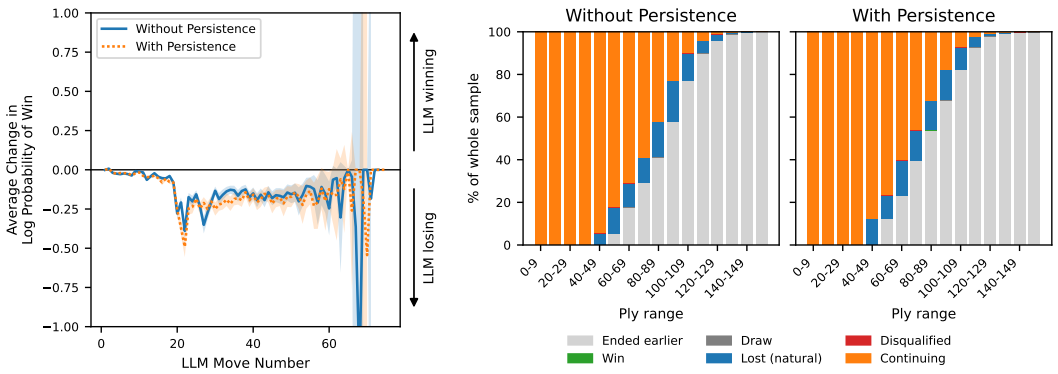


Figure 6: **(left)** The logarithm of the estimated LLM win probability for each move with and without allowing responses from previous moves to persist in context. **(right)** Outcomes in each setting binned by ply at which the outcome occurred.

5 CONCLUSION

We evaluate LLMs across several chess variants against Fairy Stockfish and measure move quality throughout each setting. We also compare LLM performance on algebraic notation versus FEN, as well as LLM performance against different strengths and styles of opponent engine. While current LLMs perform reasonably well at standard chess, they often flounder on chess variants involving small rule alterations. We find LLMs retain higher move quality on openings in standard chess and variants in which openings do not significantly differ, but become weaker in the midgame and variants where openings do significantly change. Furthermore, LLMs perform slightly worse in openings when positions are given in FEN rather than algebraic notation. Overall, this suggests that LLMs achieve their strongest performance on chess variants and positions that are easily memorizable (such as openings given in algebraic notation) while performing less well on chess variants and positions with worse representation in training data. This indicates that, at least in the domain of chess, a large share of gains in LLM reasoning are due to memorization of specific domains rather than generalization across broad sets of reasoning.

While we ran our experiments on recent, high-quality models, we did not run our experiments on the largest frontier models due to cost considerations. We are interested in seeing an extension of this work to the largest frontier models. Additionally, while our approach did allow us to focus on positions that occurred in actual gameplay, this prevented us from analyzing LLM responses to unusual positions, or in many cases, LLM capabilities during the endgame.

This paper focuses on a fixed number of static variants, which may reduce the effectiveness of future assessments using those variants if this paper is included in LLM training data. As a result, a key priority for future work may be to dynamically generate novel chess variants. Additionally, in the process of running these experiments, we captured a large number of reasoning chains produced by LLMs on each variant, and a qualitative analysis of these reasoning chains may produce further interesting insights into model behavior and common error modes. Further assessment of the origins of LLM capabilities may be bolstered by running this experiment on an open-data model such as `OLMo-3` and investigating its training data in more detail (Olmo et al., 2026). Finally, while our project focuses on evaluating existing LLMs, further insights may be found by finetuning or performing reinforcement learning on chess variants and examining performance, as well as assessing the post-trained model weights with interpretability techniques.

AI USE STATEMENT

In this work, we used generative AI tools to provide the LLM moves in our experiments, to implement code for the experiments (and in the process make design suggestions on the margin), to implement and assist in the design of the game log output format, and to implement code to process game logs for results. We have not used generative AI tools for proposing or refining hypotheses, developing theoretical models or conceptual frameworks, or assisting in translation, and mathematical tasks are not applicable to this work. Additionally, we used generative AI tools for designing figures, tables, and other complex LaTeX elements, writing software code, compiling a review of existing research, formatting references, and providing feedback on editing questions. We have reviewed all AI-assisted work. LLM-generated code was created with manual permission settings such that each step had to be approved by a human, papers brought up by the LLM literature review were checked to ensure they really existed, and writing was manually typed by a human. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Mathieu Acher. Debunking the chessboard: Confronting GPTs against chess engines to estimate Elo ratings and assess legal move abilities. `https://blog.mathieuacher.com/GPTsChessEloRatingLegalMoves/`, sep 2023. Blog post.

AJ and Dylan Patel. RL Environments and RL for Science: Data Foundries and Multi-Agent Architectures. `https://newsletter.semianalysis.com/p/rl-environments-and-rl-for-science`, January 2026. Blog post.

Anthropic. Introducing Claude Fable 5.1 and Claude Mythos 5.1. `https://www.anthropic.com/claude-fable-and-mythos-5-1`, September 2026. Blog post.

Bahar Bateni, Benjamin Pratt, and Jim Whitehead. LLM game rule understanding through out-of-distribution fine-tuning. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 21(1):2–11, 2025. doi: 10.1609/aiide.v21i1.36804.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=dYur3yabMj`.

Anthony Costarelli, Mat Allen, Roman Hauksson, Grace Sodunke, Suhas Hariharan, Carlson Cheng, Wenjie Li, Joshua Clymer, and Arjun Yadav. GameBench: Evaluating strategic reasoning abilities of LLM agents, 2024. URL `https://arxiv.org/abs/2406.06613`.

Dynomight. Something weird is happening with LLMs and chess. https://dynomight.net/chess/, November 2024a. Blog post.

Dynomight. OK, I can partly explain the LLM chess weirdness now. https://dynomight.net/more-chess/, November 2024b. Blog post.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. ChessGPT: bridging policy learning and language modeling. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Fabian Fichter and Contributors. Fairy-Stockfish: A chess variant engine, 2021. URL https://github.com/fairy-stockfish/Fairy-Stockfish. GPL-3.0 License.

Niklas Fiekas. python-chess: a chess library for Python, 2026. URL https://github.com/niklasf/python-chess. GPL-3.0+ License.

Adam Karvonen. Emergent world models and latent variable estimation in chess-playing language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=PPTrmvEnpW.

Sai Kolasani, Maxim Saplin, Nicholas Crispino, Kyle Montgomery, Jared Quincy Davis, Matei Zaharia, Chi Wang, and Chenguang Wang. LLM CHESS: Benchmarking reasoning and instruction-following in LLMs through chess, 2025. URL https://arxiv.org/abs/2512.01992.

Kale Kundert. pba_waeber2013, 2022. URL https://github.com/kalekundert/pba_waeber2013. MIT License.

Martha Lewis and Melanie Mitchell. Using counterfactual tasks to evaluate the generality of analogical reasoning in large language models. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 46, 2024. URL https://escholarship.org/uc/item/58d9s666.

Lichess.org. lila: Lichess.org open-source chess server, 2022. URL https://github.com/lichess-org/lila. AGPL-3.0 License.

Lichess.org. Lichess chess960 games dataset. https://huggingface.co/datasets/Lichess/chess960-chess-games, 2026.

Jincheng Liu, Sijun He, Jingjing Wu, Xiangsen Wang, Yang Chen, Zhaoqi Kuang, Siqi Bao, and Yuan Yao. ChessArena: A chess testbed for evaluating strategic reasoning capabilities of large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7901–7954, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.360. URL https://aclanthology.org/2026.acl-long.360/.

Anna Mészáros, Patrik Reizinger, and Ferenc Huszár. Out-of-distribution evaluation of rule-based and strategic reasoning in chess transformers. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=JFGUsQ68rG.

Prakamya Mishra, Jiang Liu, Jialian Wu, Xiaodong Yu, Zicheng Liu, and Emad Barsoum. TTT-bench: A benchmark for evaluating reasoning ability with simple and novel tic-tac-toe-style games. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2809–2831, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.140. URL https://aclanthology.org/2025.emnlp-main.140/.

Daniel Monroe, George Eilender, Philip Chalmers, Zhenwei Tang, and Ashton Anderson. Chessformer: A unified architecture for chess modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2ltBRzEHyd.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

OpenAI. GPT-6 Astra: A new generation of intelligence. https://openai.com/index/gpt-6-astra/, September 2026a. Blog post.

OpenAI. OpenAI and Hugging Face partner to address security incident during model evaluation. https://openai.com/index/hugging-face-model-evaluation-security-incident/, July 2026b. Blog post.

OpenAI. On the Navier–Stokes Millennium Prize problem. https://openai.com/index/navier-stokes-solution/, September 2026c. Blog post.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fp6t3F669F.

Leonard S. Pleiss, Maximilian Schiffer, and Robert K. von Weizsaecker. Disentangling generalization and memorization in large language models using chess, 2026. URL https://arxiv.org/abs/2601.16823.

Anian Ruoss, Gregoire Deletang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, Cannada A. Lewis, Joel Veness, and Tim Genewein. Amortized planning with large-scale transformers: A case study on chess. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=XlpipUGygX.

John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada A. Lewis, Anian Ruoss, Tom Zahavy, Petar Veličković, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomasev. Mastering board games by external and internal planning with language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KKwBo3u3IW.

Aman Sharma and Paras Chopra. EsoLang-Bench: Evaluating genuine reasoning in large language models via esoteric programming languages, 2026. URL https://arxiv.org/abs/2603.09678.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11385–11393, 2022. doi: 10.1609/aaai.v36i10.21390.

Fien van Wetten, Aske Plaat, and Max van Duijn. Baba is LLM: Reasoning in a game with dynamic rules, 2025. URL https://arxiv.org/abs/2506.19095.

Vivek Verma, David Huang, William Chen, Dan Klein, and Nicholas Tomlin. Measuring general intelligence with generated games, 2025. URL https://arxiv.org/abs/2505.07215.

Rolf Waeber. *Probabilistic Bisection Search for Stochastic Root-Finding*. PhD thesis, Cornell University, 2013.

Qianfeng Wen, Zhenwei Tang, and Ashton Anderson. ChessQA: Evaluating large language models for chess understanding, 2025. URL https://arxiv.org/abs/2510.23948.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1819–1862, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.102. URL https://aclanthology.org/2024.naacl-long.102/.

A System Prompts

Default system prompt:

```
You are an expert chess player, playing against another chess player.
When given the move history, you will respond with a move.
Reason about each move step by step.
Keep your reasoning as brief as possible, and avoid reasoning loops.
When you are finished reasoning, give your final answer in the form ‘‘Answer: ’’,
    followed by your move in SAN with no move numbering or additional punctuation.
Do not place any additional text after your final answer.
```

System prompt for FEN setting in Section 4.2:

```
You are an expert chess player, playing against another chess player.
When given an FEN string, you will respond with a move.
Reason about each move step by step.
Keep your reasoning as brief as possible, and avoid reasoning loops.
When you are finished reasoning, give your final answer in the form ‘‘Answer: ’’,
    followed by your move in SAN with no additional punctuation.
Do not place any additional text after your final answer.
```

System prompt for Chess960 setting in Section 4.3:

```
You are an expert Chess960 player, playing against another Chess960 player.
When given the move history, you will respond with a move.
Reason about each move step by step.
Keep your reasoning as brief as possible, and avoid reasoning loops.
When you are finished reasoning, give your final answer in the form ‘‘Answer: ’’,
    followed by your move in SAN with no move numbering or additional punctuation.
Do not place any additional text after your final answer.
```

System prompt for bishop-extinction setting in Section 4.3:

```
You are an expert chess player, playing against another chess player.
This is a chess variant with a different win condition than standard chess:
    instead of winning by checkmating the king, you win by capturing both of your
    opponent's bishops. The king is not a royal piece in this variant: it is not
    necessary to move out of check, moves that leave your own king in check are
    legal, and the king itself can be captured like any other piece - none of this
     ends the game by itself. Every other rule is standard, including castling and
     pawn promotion, except that a pawn may also promote to a king.
When given the move history, you will respond with a move.
Reason about each move step by step.
Keep your reasoning as brief as possible, and avoid reasoning loops.
When you are finished reasoning, give your final answer in the form ``Answer: '',
    followed by your move in SAN with no move numbering or additional punctuation.
Do not place any additional text after your final answer.
```

System prompt for rook-bishop setting in Section 4.3:

```
You are an expert chess player, playing against another chess player.
This is a chess variant with one rule change from standard chess: rooks and
    bishops fully swap movement patterns. Rooks move diagonally, the same way a
    bishop normally would. Bishops move orthogonally, the same way a rook normally
     would. Every other rule is standard, including that rooks still castle
    normally.
When given the move history, you will respond with a move.
Reason about each move step by step.
Keep your reasoning as brief as possible, and avoid reasoning loops.
When you are finished reasoning, give your final answer in the form ``Answer: '',
    followed by your move in SAN with no move numbering or additional punctuation.
Do not place any additional text after your final answer.
```

System prompt for leap-pawn setting in Section 4.3:

```
You are an expert chess player, playing against another chess player.
This is a chess variant with one rule change from standard chess: pawns move
    differently. A pawn always moves exactly two squares straight ahead, never one
    , and this is blocked if either the square immediately in front of it or the
    square two ahead is occupied. A pawn captures by leaping two squares straight
    ahead and one square to either side (the shape of a knight's move, restricted
    to the two forward ones), landing only on a square occupied by an enemy piece
    - pawns do not capture diagonally, this capture is a leap so it is not blocked
     by anything in between, and there is no en passant. When a pawn reaches the
    last rank it must promote, and always promotes to a queen - there is no choice
     of promotion piece in this variant. Every other rule is standard, including
    castling.
When given the move history, you will respond with a move.
Reason about each move step by step.
Keep your reasoning as brief as possible, and avoid reasoning loops.
When you are finished reasoning, give your final answer in the form ``Answer: '',
    followed by your move in SAN with no move numbering or additional punctuation.
Do not place any additional text after your final answer.
```

System prompt for no reasoning + direct prompting setting in Section 4.4:

```
You are an expert chess player, playing against another chess player.
When given the move history, you will respond with a move.
Do not reason or explain your choice.
Respond with only your move in SAN, with no additional punctuation (such as
    leading ellipsis, move number, or annotation symbols), and no other text
    before or after it.
```

B USER MESSAGES

AmbiguousMoveError. Occurs when the SAN move proposed matches more than one possible legal move.

```
Your response was an ambiguous move. Please disambiguate your move.
```

IllegalMoveError. Occurs when the move proposed is illegal.

```
Your response was an illegal move. Please make a legal move.
```

InvalidMoveError. Occurs when the text after "Answer:" is not parseable (usually due to additional punctuation).

```
Your response was not formatted correctly. Please format your move in SAN as
    instructed, with no additional punctuation (such as leading ellipsis, move
    number, or annotation symbols).
```

InvalidResponseError. Occurs when no instance of "Answer:" is found.

```
Your response was not formatted correctly. Please give your final answer in the
    form ‘‘Answer: ’’, followed by your move, with no additional text.
```

C RULES DEMONSTRATIONS

D VARIANT CONFIGURATIONS

Bishop-extinction.

```
[bishop-extinction:extinction]
extinctionValue = loss
extinctionPieceTypes = b
```

Rook-bishop. When boards of this configuration are shown to the LLM, "v" is replaced by "b", and "w" is replaced by "r".

```
[rook-bishop:chess]
bishop = -
rook = -
customPiece1 = v:R
customPiece2 = w:B
promotionPieceTypes = nvwq
castlingRookPiece = w
startFen = wnvqkvnw/pppppppp/8/8/8/8/PPPPPPPP/WNVQKVNW w KQkq - 0 1
```

Leap-pawn. When boards of this configuration are shown to the LLM, "j" is replaced by "p".

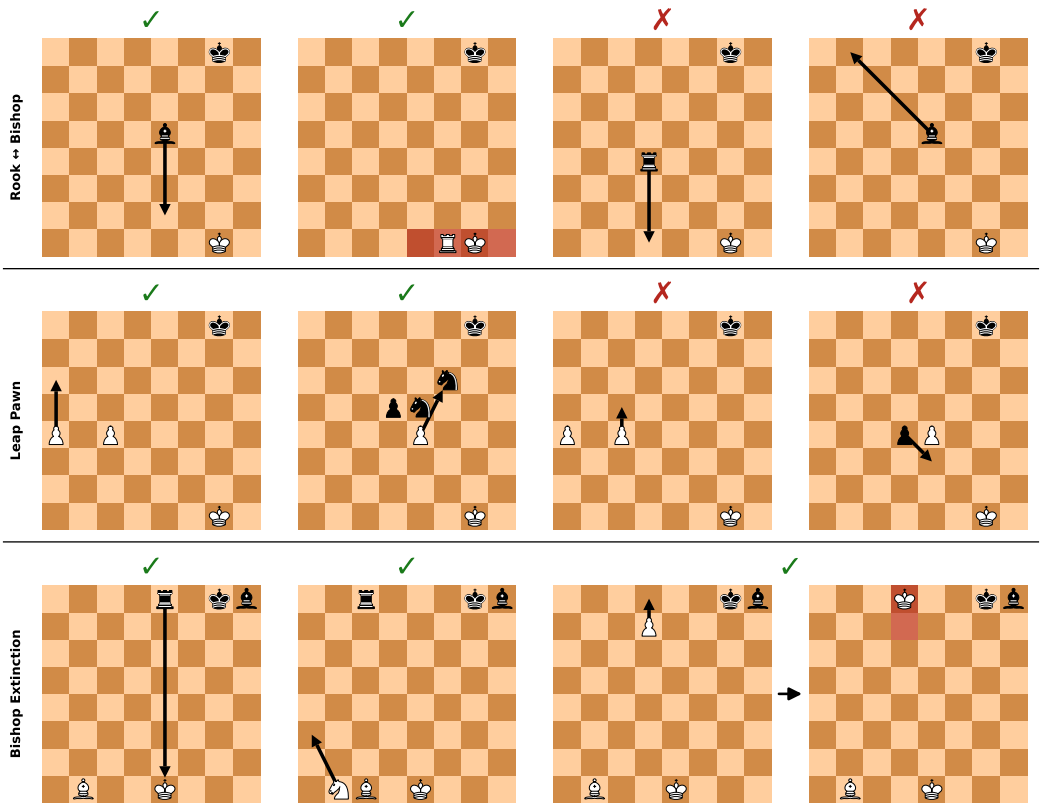


Figure 7: Examples of legal and illegal moves in each variant.

```
[leap-pawn:chess]
pawn = -
customPiece1 = j:fcNfmW2
promotedPieceType = j:q
mandatoryPiecePromotion = true
mobilityRegionWhiteCustomPiece1 = *2 *4 *6 *8
mobilityRegionBlackCustomPiece1 = *1 *3 *5 *7
startFen = rnbqkbnr/jjjjjjjj/8/8/8/8/JJJJJJJJ/RNBQKBNR w KQkq - 0 1
```

E MIXED STOCKFISH/RANDOM CALIBRATION

PBA tests multiple games at the same estimate $\hat{p}$ until it becomes confident enough that $\hat{p}$ falls below or above the true value of p. Then, PBA switches to another value of $\hat{p}$. Allowing PBA to completely converge would take a significant amount of time, so we set p for the MIXED engine to be the empirical value $\hat{p}$ at the end of each calibration run. Since each calibration run spent over 500 iterations on the $\hat{p}$ found at the end of the run, we consider this to likely be close to the true value of p. Below, for each MAIA engine Elo, we record the values of p, the number of games spent on p (which we call n), and the Win/Draw/Loss record in favor of the MIXED engine out of those n games.

F FAIRY STOCKFISH SCORING DETAILS

The Fairy Stockfish engine has a built-in evaluation metric that scores positions in terms of centipawn score c or "mate in N." These metrics are initially incommensurable, so we adapt python-chess's Lichess Win/Draw/Loss (WDL) function to convert to a win probability (Lichess.org, 2022;

Table 1: Win/draw/loss record of the MIXED engine against MAIA at value p, by Elo

Elo	p	n	W/D/L
600	0.383424	556	244/113/199
1600	0.165741	940	455/64/421
2600	0.103588	966	418/78/470

Fiekas, 2026). If the value given by Fairy Stockfish is a "mate in N" score, we convert it to a centipawn score via the function $21 - \min(10, |N|)$. If the value given by Fairy Stockfish is originally a centipawn score c, we clamp it to within $[-1000, 1000]$ in order to keep it from running into the "mate in N" range. Finally, we compute the win probability from the resulting centipawn score c using the sigmoid function calibrated from Lichess:

$$\frac{1}{1 + \exp(-0.00368208 \cdot c)}$$

.

While the standard python-chess library rounds this value, we use the unrounded value for greater accuracy.

G OUTCOME CHARTS

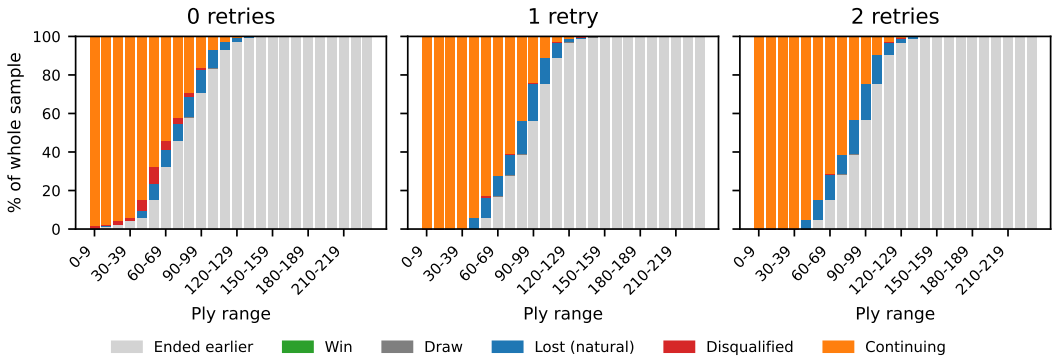


Figure 8: Outcomes for `Gemini-3.8-Flash` in Section 4.1.

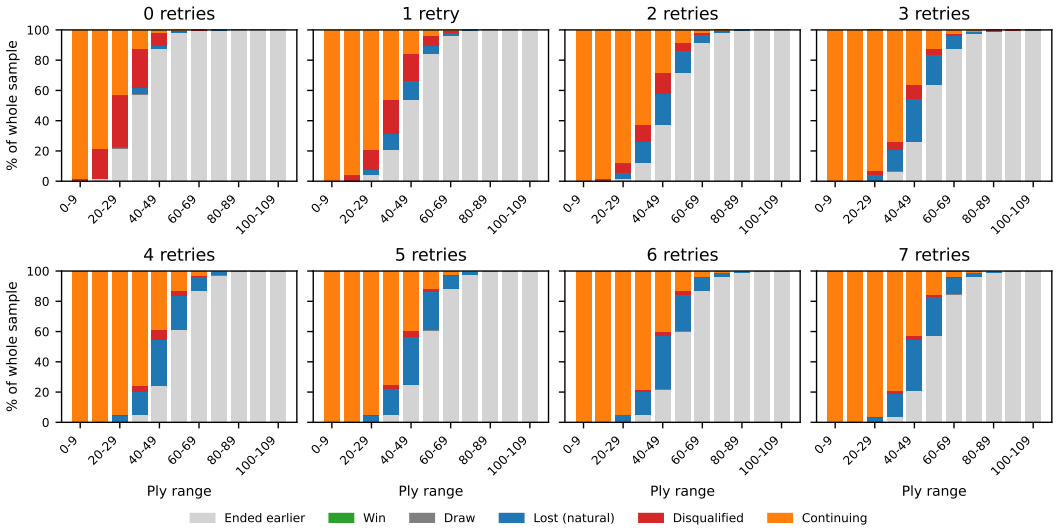


Figure 9: Outcomes for `GPT-5.6-Luna` in Section 4.1.

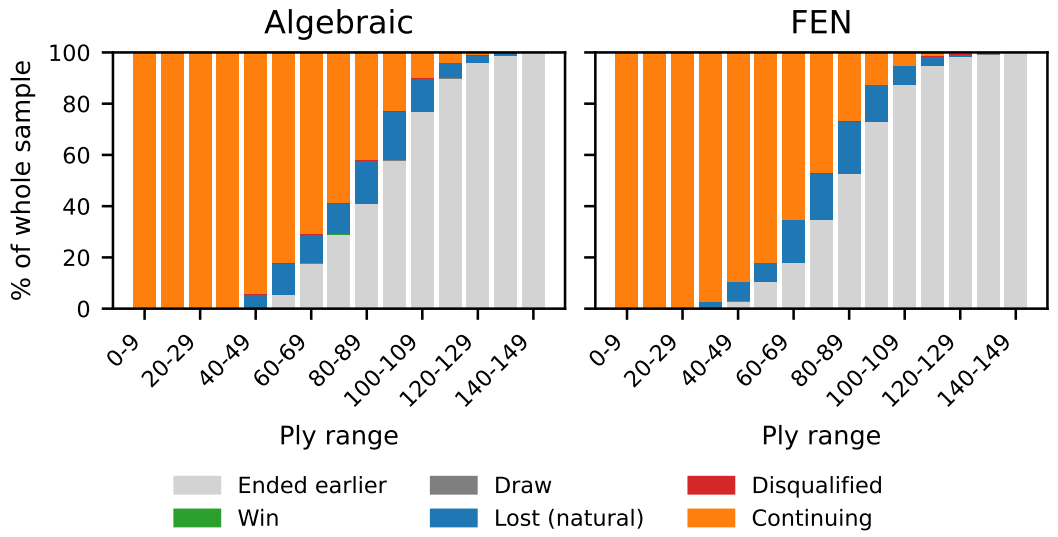


Figure 10: Outcomes for `Gemini-3.8-Flash` in Section 4.2.

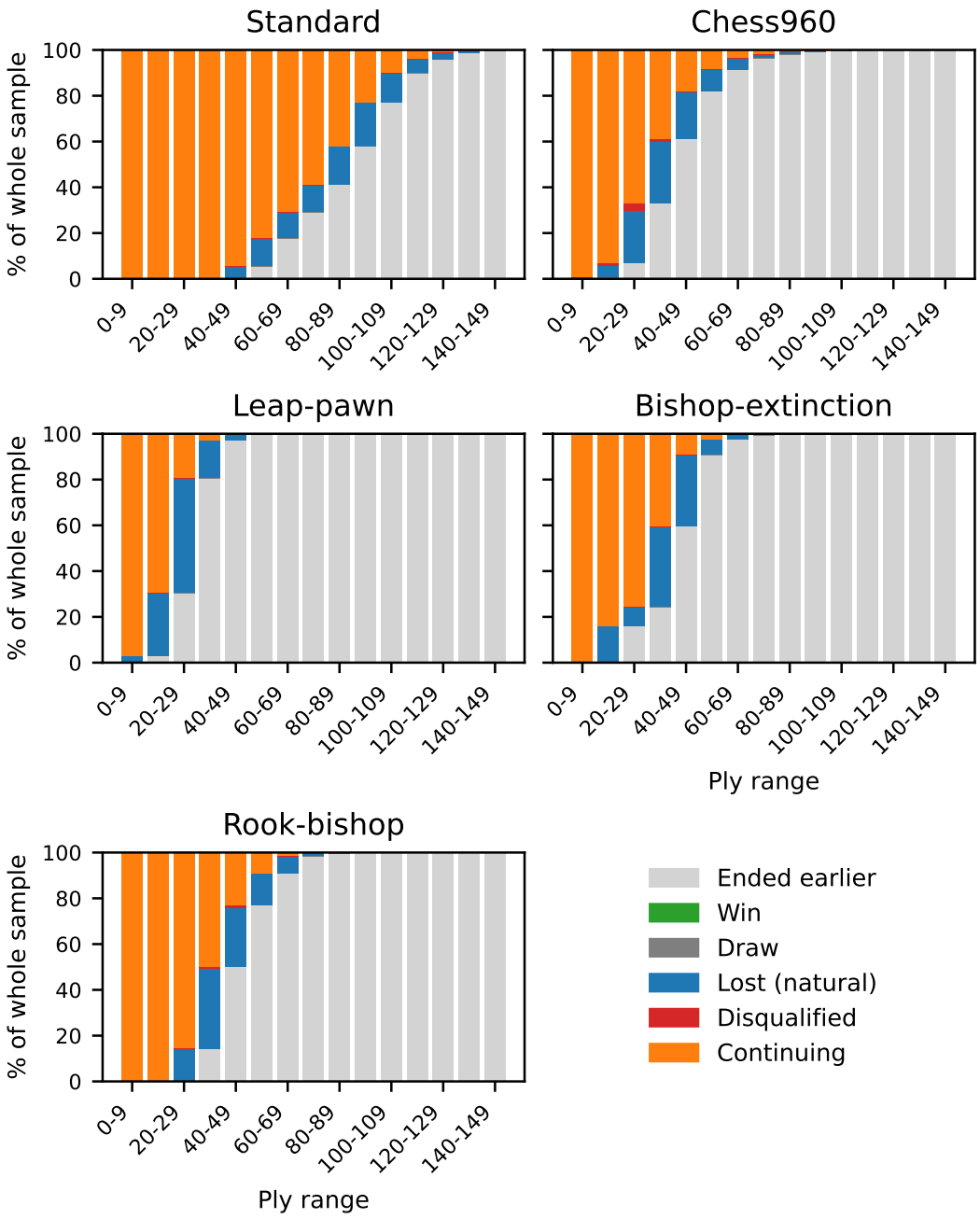


Figure 11: Outcomes for `Gemini-3.8-Flash` in Section 4.3.

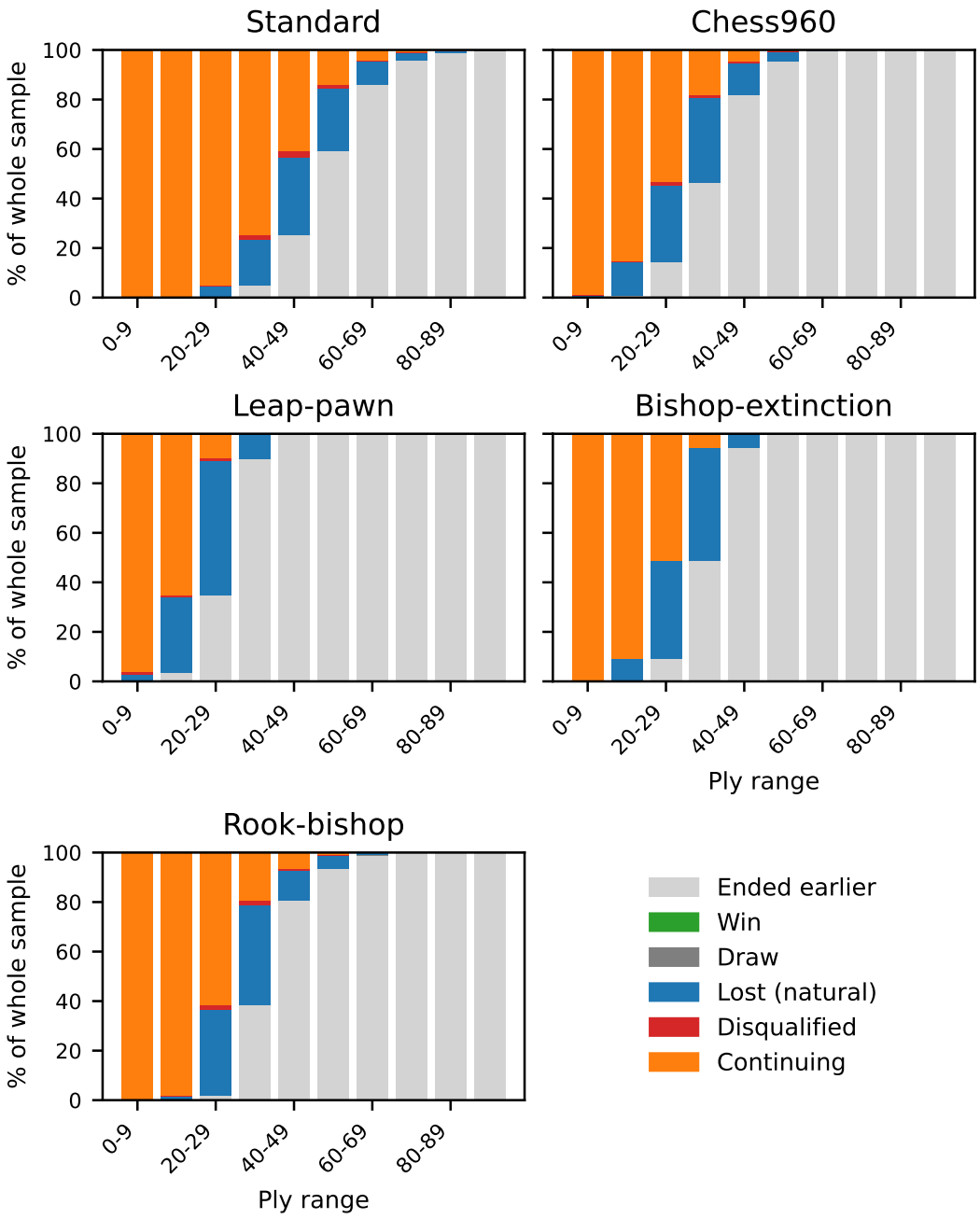


Figure 12: Outcomes for `GPT-5.6-Luna` in Section 4.3.

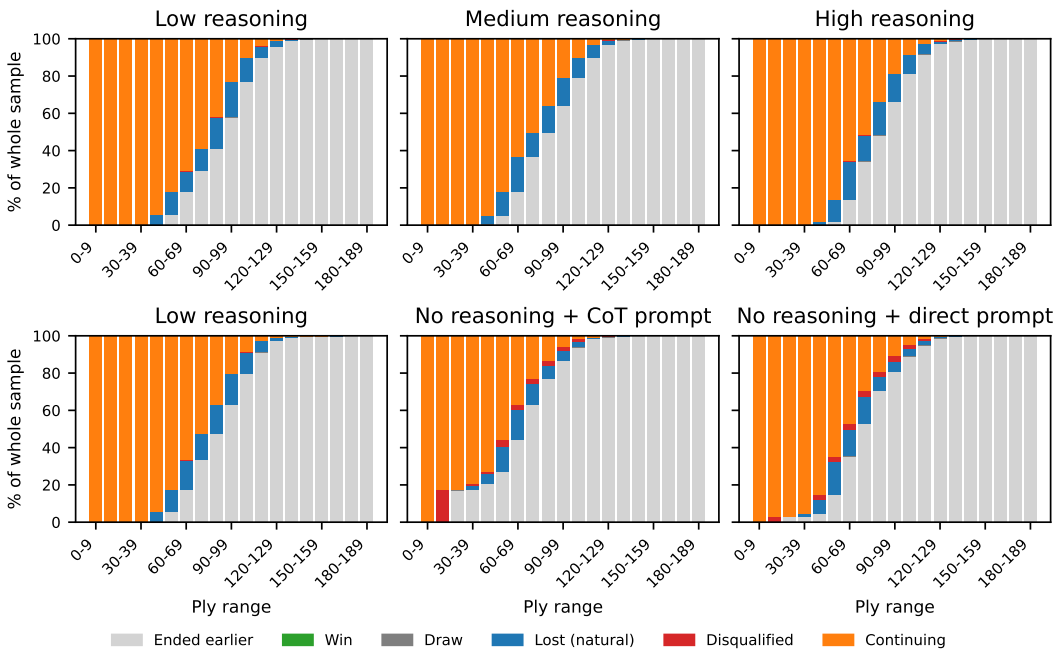


Figure 13: Outcomes for `Gemini-3.8-Flash` and `Gemini-3.6-Flash` in Section 4.4.

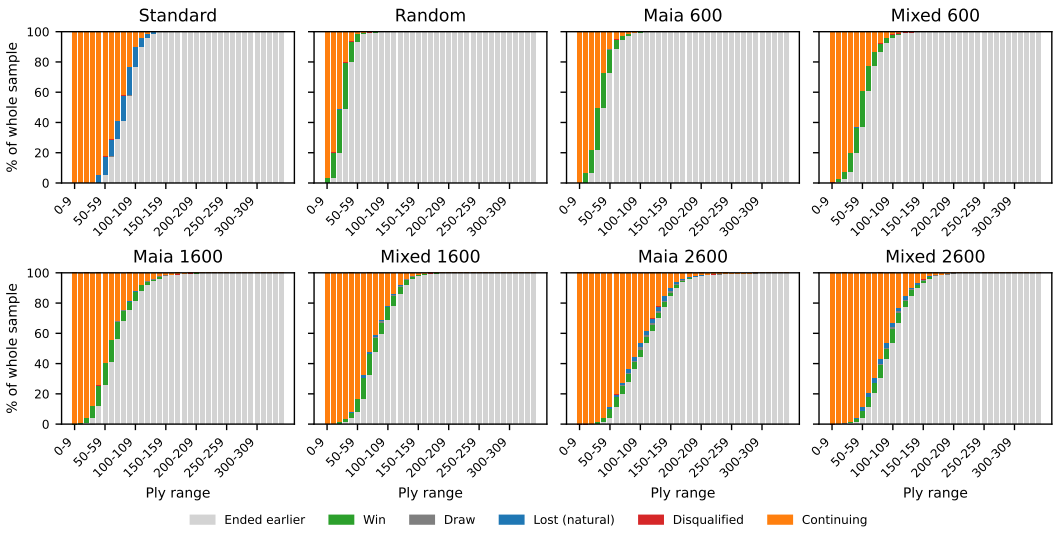


Figure 14: Outcomes for `Gemini-3.8-Flash` in Section 4.5.

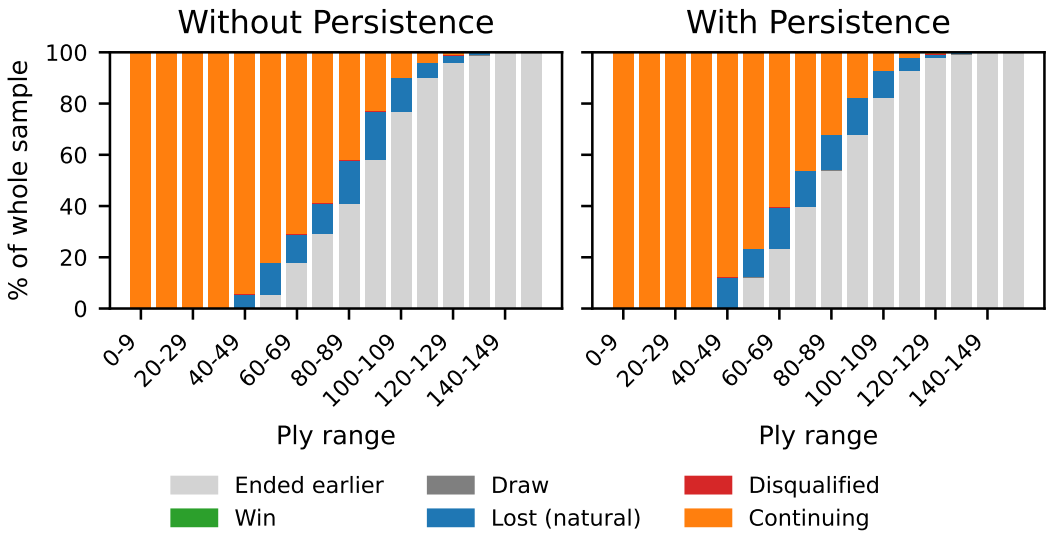


Figure 15: Outcomes for `Gemini-3.8-Flash` in Section 4.6.