Chess Engine: Testing Report

1 Overview

The testing process was designed to evaluate both the external performance and the internal stability of my chess engine through two complementary approaches. First, the engine was benchmarked against Stockfish, an open-source engine widely regarded as one of the strongest available. Then, the engine was evaluated in self-play mode, where two identical instances competed under identical search parameters.

2 Methodology

2.1 Matches against Stockfish

To create opponents of varying strength, Stockfish's Elo limiter (`UCI_LimitStrength`) was used. This feature constrains the engine's playing ability to approximate a target Elo rating (a numerical measure of player strength). The engine was tested across three separate batches, each consisting of 24 games against Stockfish configured to a different Elo cap. To eliminate color bias, colors were alternated automatically: in each batch, my engine played 12 games as White and 12 as Black.

Batch	**Stockfish Elo cap**	**Games**
Batch 1	1500	24 (12W, 12B)
Batch 2	1900	24 (12W, 12B)
Batch 3	2300	24 (12W, 12B)

For context, a player rated 1500 Elo is considered of competent club strength, 1900 corresponds to strong amateur or expert level, and ratings above 2300 represent master-level play. This provides a basic intuitive scale for interpreting the difficulty of each test batch, although it should be noted that engine Elo ratings are estimates which only loosely correspond to human ratings.

In all cases, my engine was restricted to a fixed search depth of three plies, while Stockfish operated without a depth limit but was constrained to a maximum thinking time of 0.5 seconds per move. This configuration ensures that both engines operate within roughly comparable computational budgets, with my engine exploring a fixed search horizon and Stockfish utilizing its full evaluation capabilities within its temporal constraint.

All games were conducted locally using the `python-chess` engine interface, which manages move generation, legality checking, and PGN recording. Both engines were launched via the UCI protocol using a dedicated script (`match_vs_stockfish.py`). Each game was saved in PGN format, including metadata such as date, round, Stockfish configuration, and result, and custom headers were added to record the test parameters.

For each batch, all PGNs were merged into single files, which are available inside `testing_data/`.

2.2 Self-play match

Because the engine is fully deterministic, two identical instances playing under the same parameters will always reproduce the exact same game. As a result, traditional self-play testing (which relies on aggregating outcomes across multiple games) would not provide meaningful statistical insight. Nevertheless, a single controlled self-play match was conducted to provide a general diagnostic assessment of evaluation consistency and move-selection accuracy.

This match was generated using the built-in self-play function at a fixed search depth of three plies. The resulting game provides a reproducible case for assessing the coherence of the evaluation function. The game was recorded in PGN format and later evaluated through post-game computer analysis. The PGN file is included in `testing_data/`.

3 Evaluation metrics

3.1 Matches against Stockfish

Each batch produces a set of quantitative indicators automatically computed by the testing script. These are summarized as win–draw–loss counts and converted into numerical scores according to standard chess conventions: 1 point per win, 0.5 points per draw, and 0 per loss.

In addition to overall scores, results are broken down by color. Comparing the win rates for each side helps identify whether the engine performs symmetrically or whether its evaluation and search heuristics favor attacking or defensive positions. This comparison must take into account the fact that White (making the first move) typically enjoys a small but measurable statistical advantage.

Draw frequency also carries diagnostic value: a high proportion of draws against stronger opponents may indicate that the engine excels at defensive play and error avoidance, while frequent decisive outcomes against weaker settings reveal tactical opportunism and conversion ability.

3.2 Self-play match

The primary quantitative indicator used in self-play analysis is the average centipawn loss (CPL), computed by running a post-game analysis with Stockfish. CPL measures the average difference in resulting evaluation between the move the engine played and the move Stockfish would have chosen in the same position.

Lower CPL values indicate higher accuracy and stronger play. For context, the strongest modern engines typically achieve average CPL values below 10, corresponding to superhuman performance. Engines limited to around 30–50 CPL play at a strong expert or candidate master level, while values near 70–100 CPL are consistent with competent club-level or advanced amateur play.

In addition to accuracy, the game's outcome provides insight into the engine's internal stability. A draw suggests that both instances of the engine evaluate positions in a balanced way and that the search process remains stable over long sequences. Conversely, if one side were to win, it would indicate that small numerical biases or asymmetries in the evaluation function can accumulate over time, leading to self-reinforcing positional errors.

4 Results

4.1 Matches against Stockfish

A total of 72 games were played across three difficulty settings, each consisting of 24 games (12 as White and 12 as Black). The results below summarize the performance of my engine relative to Stockfish configured with different Elo caps.

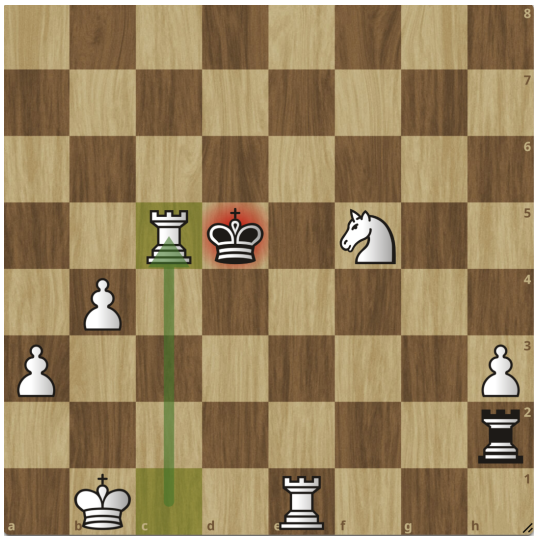
Batch	Opponent Elo setting	Games	W–D–L	Score	Score (%)
Batch 1	1500	24	19–3–2	20.5/24	85.42
Batch 2	1900	24	9–7–8	12.5/24	52.08
Batch 3	2300	24	4–6–14	7.0/24	29.17

The results demonstrate the expected decline in performance as the opponent's Elo rating increases. At 1900 Elo, performance was roughly even, suggesting that at this configuration the engines operate at a similar overall strength within the imposed time and depth constraints.

Color	Games	W–D–L	Score (%)
White	36	16–10–10	58.33
Black	36	16–6–14	52.78

As expected, the color-specific breakdown shows a slight advantage when playing as White. However, the near parity of results across colors indicates that the engine's evaluation and move selection remain balanced and do not exhibit a strong bias toward attacking or defending.

The images below illustrate final checkmate positions from games in which my engine defeated Stockfish at the 2300 Elo setting (games 14 and 20 from the third batch).



Games were visualized using Lichess, a free, browser-based chess platform that allows interactive playback and board visualization directly from PGN files.

4.2 Self-play match

The self-play match was analyzed to assess the engine's evaluation accuracy and internal stability. The game lasted 27 moves per side (54 plies in total) and ended in a draw, suggesting balanced evaluations and the absence of systematic bias between the two sides. Because the game outcome and move sequence are fully deterministic, the analysis focuses on positional quality rather than result variance.

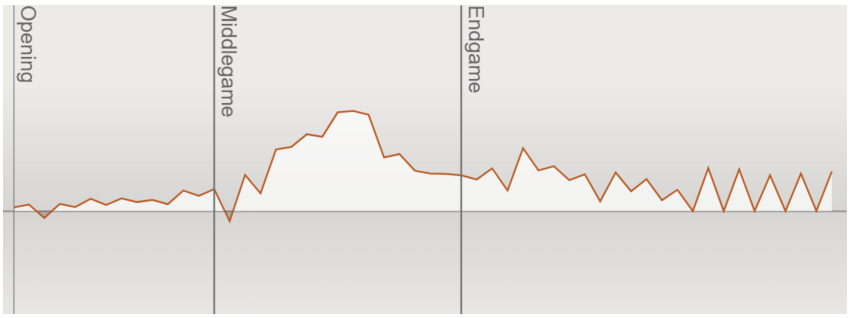
The table below summarizes the average centipawn loss (CPL) computed separately for each color using a Stockfish-based computer analysis on Lichess.

Average CPL (White)	48 centipawns
Average CPL (Black)	52 centipawns

The nearly identical CPL values for both sides seem to confirm that the engine's evaluation function behaves symmetrically, with no measurable difference in move-selection accuracy between colors.

The observed CPL values correspond to generally competent tactical play, although it should be noted that, since the sample size is one, these metrics should be interpreted qualitatively and not as a statistically significant measure of performance. In future versions, a small degree of randomization, such as stochastic move ordering or evaluation noise, could be introduced in order to allow multiple distinct self-play games and to conduct a deeper and more thorough analysis.

To visualize the game's progression, the evaluation trajectory derived from Stockfish analysis is shown below. The graph depicts the evolution of position evaluation from White's perspective throughout the game.



Although White achieved a modest middlegame advantage (approximately +290cp), both instances of the engine tend to converge toward somewhat balanced endgames, suggesting a generally stable evaluation behavior.