National College of Ireland

Comparative Analysis of AI Reasoning Architectures in Chess: O1 vs R1 Performance Evaluation

MSc Research Project
Msc in Data Analytics

Ravi Kishore Muppa
Student ID: X23329114

School of Computing
National College of Ireland

Supervisor: Shubham Subhnil

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ravi Kishore Muppa
Student ID:	X23329114
Programme:	Msc in Data Analytics **Year:** 2024-2025
Module:	Msc Research Praticum
Supervisor:	Shubham Subhnil
Submission Due Date:	15-09-2025
Project Title:	Comparative Analysis of AI Reasoning Architectures in Chess: O1 vs R1 Performance Evaluation
Word Count:	10733 **Page Count:**29

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ravi Kishore Muppa

Date: 15-09-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Comparative Analysis of AI Reasoning Architectures in Chess: O1 vs R1 Performance Evaluation

Ravi Kishore Muppa
IDX23329114

Abstract

This work shows the systematic analysis of specialized AI models' chess playing performance as a testbed using chess, and a comparison of the R1 model from DeepSeek and the O1 model from OpenAI. This work fills the gap in the quantitative comparison of various architectural styles on chess playing performance and move quality. Comprehensive CPL (Centipawn Loss) evaluation metric analyses of 120 games played from 30 strategically distinct chess startup positions and 2,400 moves were conducted. Our evaluation framework comprised direct model comparisons, analysis of self-play consistency, and performance comparisons on varying position types (tactical, strategic, and endgame positions). Our results show that O1's hybrid of supervised learning and reinforcement learning style results in significantly better performing chess playing than R1's pure reinforcement learning architecture, achieving 25.6 versus 87.0 average CPL (3.5 times better performing). O1 showed higher consistency (75% lower standard deviation), higher overall performance on all position types, and an 85% victory rate on one-vs-one competition. Statistical testing upheld these differences as strongly significant ($p < 0.001$). This research discovers that design decisions of architectures significantly influence move quality and chess playing ability where O1 consistently performed over a range of position complexity while R1 experienced severe performance loss on complex positions. From the experiment on this research, we have empirical proof that an integration of learning on master games and reinforcement learning results in an overall better outcome than pure self-play methods of building AI programs on chess-related domains.

1 Introduction

The chess AI community has undergone wide development, ranging from the earliest search-based programs up through the latest language-based models, but the essential gap in obtaining transparent strategic reasoning remains. Stockfish and other conventional chess engines exhibit superhuman ability through extensive position analysis and search planning and check an enormous number of positions per second with very high fidelity (Maharaj, Polson & Turk, 2022). These systems inherently don't possess the ability to provide their strategic reasoning process in human-understandable terms and thus severely inhibit their usability within teaching settings and co-analysis situations (Abdelghani, Dari & Banitaan, 2023).

Recent successes of Large Language Models (LLMs) attempted to bridge this explainability gap through the generation of chess moves with natural language commentary. There is, however, voluminous evidence that these general purpose models have shallow depth of strategy and generate erratic or superficial explanations (Noever, Ciolino & Kalin, 2020). The

Chess Transformer, having been trained on millions of games, failed to demonstrate the potential for broad-based strategic understanding such that memorized opening structures were not a component of it (Noever, Ciolino & Kalin, 2020). The failure means there is an intrinsic issue: achieving computational precision and insightful strategic explanation within a single framework.

The development of dedicated models of reasoning, such as OpenAI's O1 (OpenAI, 2024) and DeepSeek's R1 (DeepSeek-Ai et al., 2025), is a possible transformative direction of artificial intelligence. The models utilize chain-of-thought architecture and dedicated optimizations aimed at reasoning more similarly akin to the process of human thought in engaged problem-solving. O1 utilizes a hybrid technique of applying supervised fine-tuning and reinforcement learning in order to build internal chains of reasoning and has an 83.3% success on engaged code-writing problems (Zhong et al., 2024). DeepSeek-R1 establishes by contrast the feasibility of advanced-level functions of reasoning by means of pure reinforcement learning and without gigantic sets of supervised data and functions similarly but completely in the open due to its open-source nature (DeepSeek-Ai et al., 2025).

The drive stems from the lack of systematic evaluation frameworks that compare the performance of varying architectural forms in specialized models of reasoning for chess. Although measures of move accuracy are available, there isn't a standard way of measuring how these specialized models of chess reasoning perform on varying chess position types and if their architectural variations result in quantifiable differences in their performances. This gap is greater considering the growing relevance of determining which forms of AI development result in higher performing strategic tasks.

This research addresses the following question: How do specialized AI reasoning models OpenAI’s O1 and DeepSeek's R1 differ in chess playing performance, move quality consistency, and position-specific competency as measured by Centipawn Loss analysis across tactical, strategic, and endgame positions?

The three primary aims of this research are as follows. Firstly, building an overall evaluation framework systematically to test and compare two different chess-playing AI architectures in standard chess situations. Secondly, measuring O1 and R1's move quality and consistency differences through the use of Centipawn Loss (CPL) as a measurable objective function on a range of tactic, strategic, and end-game situations. Lastly, ascertaining through empirical methods position-specific competence patterns and determining which of the models exhibits greater adaptability on a range of different chess situations.

The research success indicators involve: (1) the successful deployment of an automatic evaluation framework that is able to test several AI models on standardized chess positions; (2) obtaining statistically significant data on chess playing performance for at least 2,400 move evaluations; (3) proof of measurable differences between playing performances of chess models being evaluated; and (4) discovery of position-dependent playing performance profiles that indicate strengths and weaknesses of each model.

The method employs a comparative study framework based on 30 meticulously designed chess positions that highlight various strategic themes. They are utilised across four model pairing conditions for each position to provide extensive performance data, including centipawn loss calculation, reaction times, and consistency metrics.

The report's outline is as follows. Section 2 is a critical review of the literature on chess AI research. It looks at how research has changed from classical engines to language models and then to reasoning systems that are specific to a certain field. In Section 3, you'll find the research design. This includes making the evaluation framework, setting up metrics, and figuring out how to run tests. Part 4 goes into great detail about the technical aspects of the system that was used. Section 5 gives a lot of information about how the chess arena interface and the whole evaluation framework will work. Section 6 talks about what happened in the experiment and the numbers that came out of it. Section 7 contains the important discussion of the results and what they mean. Finally, Section 8 ends with a summary of the contribution and suggestions for future research.

2 Related Work

The development of chess artificial intelligence includes a wide range of models, from older search-based engines to newer reasoning models. This critical review looks at the different methods that are already out there and how they help and hurt the goal of strategic understanding and explanation.

2.1 Traditional Chess Engines and Their Limitations

Traditional chess engines employ nearly all of the search algorithms combined with powerful evaluation functions in order to achieve superhuman performance. Chole and Gadicha (2022) employed a hybrid optimization approach integrating classical algorithms with neural networks trained on grandmaster games, achieving an ELO rating of 2200. While demonstrating competitive gameplay, their system fundamentally lacks mechanisms for strategic explanation, highlighting the persistent challenge of bridging computational excellence with interpretable reasoningThe authors acknowledge that traditional engines calculate millions of variations within time constraints but not develop intuitive knowledge, a limitation that hampers their applicability in educational and analytical contexts.

Abdelghani, Dari, and Banitaan (2023) conducted a comparative analysis between traditional Alpha-Beta pruning approaches and deep learning methods. Their traditional engine achieved an ELO rating of 2200 against Stockfish, while the deep learning variant reached 1900. Despite competitive performance, neither approach successfully addressed the fundamental explanation gap. The paper demonstrates that while Alpha-Beta pruning with high-level move ordering is computationally effective, no insight is gained in the processes of strategic reasoning. The constraint is highlighted strongly when we consider applications for human-interpretable analysis.

Madake et al. (2023) sought to rectify this deficiency by utilising a hybrid methodology that combines machine learning estimators with Minimax and Alpha-Beta pruning algorithms. The 96.77% accuracy rate for move predictions in their system shows how well integrated approaches work. But the framework is limited because it only focusses on finding the best move, which means it doesn't help players understand strategy. Although it is possible to compute the variable depth in the number of legal moves, it does not keep the subtle positional understanding that is needed for good strategic explanation.

Maharaj, Polson, and Turk (2022) provide a compelling juxtaposition of search-effective engines like Stockfish and evaluation-based engines such as LCZero. In their analysis of Plaskett's Puzzle, they pinpoint fundamental disparities in problem-solving methodologies: Stockfish's 1.9-billion-position search contrasted with LCZero's pattern-based evaluation. The comparison reveals a fundamental deficiency in classical engines, which rely on brute-force calculation to achieve greater strength rather than strategic insight, rendering them unsuitable for applications requiring explainability.

2.2 Personalization and Strategic Understanding in Chess AI

Progress towards more human-like chess understanding is achieved through the adaptation of personalisation methodologies. Liu (2022) developed a personalised quantitative system that refined Stockfish's decision-making by utilising opponent-specific patterns derived from historical games. The method itself is a significant shift from just objective evaluation, as it also includes the strategic adaptation of human playing styles. The system really takes advantage of its opponent's weaknesses by using a tendency table of positions that is tailored to each opponent. But using outside statistical models instead of inner strategic representation limits its ability to explain and reason in a general way.

Gaessler and Piezunka (2023) examined the influence of chess computers on the enhancement of human players, concluding that AI training partners serve as an inadequate substitute for human players. Players who trained mostly on AI made fewer mistakes that were specific to humans in real competition. The findings reveal a significant vulnerability: the uniform computation of chess AI fails to replicate the psychological and strategic nuances of human play. The paper emphasises that effective strategic knowledge is essential not only for executing optimal play but also for understanding human inclinations within the decision-making framework.

Adhikari, Anatolyev, and Dagaev (2023) examined sequential error patterns in human chess, discovering that minor errors typically lead to cascading effects, whereas major errors tend to foster more disciplined subsequent play. These human behaviours are very different from the way traditional engines work all the time. The paper demonstrates the necessity for AI systems capable of capturing and displaying strategic scenarios that extend beyond mere position evaluation, particularly in contexts influenced by psychological factors and shifts in momentum.

2.3 Language Models and Chain-of-Thought Reasoning in Chess

Language models used in chess have made AI systems easier to understand. Ciolino, Kalin, and Noever (2020) created a Chess Transformer that can identify typical opening patterns and recommend moves, and they taught GPT-2 roughly 2.8 million chess games. However, the model struggled with deep strategic thinking and consistently produced superficial results where it was unsure of its location. The study demonstrates that language models are capable of learning chess notation and fundamental patterns, but they fall short for the sophisticated reasoning required for expert-level strategic thinking.

Renze and Guven (2024) investigated mini-chain-of-thought prompting on large language models and achieved a 48.7% reduction in response length while maintaining performance on the majority of the tasks. GPT-3.5 showed a 27.69% degradation of performance on mathematics tasks using compact prompts, and hence there is an inherent compromise of explanation quality and depth of reasoning. The finding has direct implications for chess-based applications, where strategic explanation requires a balance of computationally efficient processing and depth of reasoning.

Panchal, Mishra, and Shrivastava (2021) used convolutional neural networks for predicting chess moves and reached 39.16% accuracy for evaluation of the board. The system showed capability for recognition of piece activity pattern but the inability to forecast plans of an opponent or lay down lengthy plans is a limitation. The limitation of short term recognition of tactics but not of strategic prediction is characteristic of the general problem of the neural approaches that recognize local patterns but not the global understanding of strategy.

Chen et al. (2023) investigated turn-based strategy game reinforcement learning by applying deep neural networks and Monte Carlo Tree Search. The method reached phenomenal success for self-play learning on war chess modifications but is found to be restricted to transferring to ordinary chess, in that the acquired strategies do not provide human-interpretable explanations.The black-box property of the decision-making of the neural network continues even with better metrics of performance.

2.4 Advanced Reasoning Models and Evaluation Frameworks

Recent advancements in specialised reasoning models present prospective remedies to enduring constraints. The OpenAI O1 system (OpenAI, 2024) presents an innovative methodology that integrates "reasoning tokens" and prolonged "thinking time" prior to formulating responses. According to Zhong et al. (2024), O1 has an 83.3% success rate on hard coding problems and a perfect score on high school maths. This shows that it improves reasoning skills across all subjects. The system's ability to help with internal reasoning chains and make explanations easier to understand is a good step in the right direction. The proprietary and relatively transparent nature of O1's reasoning, however, precludes substantial analysis of mechanisms for strategic comprehension.

DeepSeek-R1 (DeepSeek-Ai et al., 2025) presents an alternative methodology that employs pure reinforcement learning, lacking supervised fine-tuning. The system itself develops chain-of-thought capabilities, making O1's performance transparent and accessible through open-sourcing. The DeepSeek-R1 architecture, which has 671 billion combined parameters and uses a mixture-of-experts approach, shows that high-level reasoning is possible and can be done without a lot of supervised data. The model makes long, high-level chains of reasoning and coherent strategic stories that avoid the big problems with older methods.

Through the use of reconnaissance blind chess, in which players operate with incomplete information, Czupyt, Małkiński, and Mańdziuk (2024) introduced novel evaluation techniques. Their robot Zubat, which ranks second out of 117 robots, combines uncertainty modelling with traditional engines. This method highlights a topic that has received less attention: strategic reasoning in uncertain situations. However, because of its specialised nature, reconnaissance blind chess is challenging to incorporate directly into conventional chess evaluation frameworks.

2.5 Research Gap

In chess AI, the literature shows a recurring discrepancy between strategic explanation and computational performance. Traditional engines excel through brute-force calculation but cannot articulate strategic reasoning (Abdelghani, Dari & Banitaan, 2023; Chole & Gadicha, 2022). Personalization approaches enhance adaptability but rely on external pattern databases rather than internal understanding (Liu, 2022; Gaessler & Piezunka, 2023). General language models generate explanations but lack deep strategic comprehension (Noever, Ciolino & Kalin, 2020; Panchal, Mishra & Shrivastava, 2021).

Although such specialty models as O1 and DeepSeek-R1 demonstrate remarkable capacities, no systematic method is offered for comparing their chess playing performance using standardized metrics. The absence of comprehensive evaluation frameworks for measuring move quality, consistency, and position-specific competency across different AI architectures represents an important research gap. Current literature lacks empirical evidence quantifying how these specialized reasoning models perform in diverse chess scenarios and whether their different development approaches yield measurable performance differences.

Research Question

This gap leads to the following research question: **How do specialized AI reasoning models—OpenAI's O1 and DeepSeek's R1—differ in chess playing performance, move quality consistency, and position-specific competency as measured by Centipawn Loss analysis across tactical, strategic, and endgame positions?**

This research question directly addresses the identified gap by:

1. Establishing a systematic comparison framework for specialized reasoning models
2. Using standardized metrics (CPL) to quantify performance differences
3. Examining performance across diverse position types to understand model adaptability

The question connects to three specific objectives: First, developing a comprehensive evaluation framework for systematic model comparison addresses the absence of standardized assessment methods. Second, quantifying move quality differences through CPL analysis provides the empirical evidence currently missing from the literature. Third, identifying position-specific competency patterns reveals how different models adapt to varying chess scenarios, filling the gap in understanding model performance characteristics across different game situations.

3 Research Methodology

3.1 Research Design Overview

This research employs a mixed-methods approach combining quantitative performance metrics with systematic analysis of reasoning outputs to evaluate specialized AI reasoning models in chess. The methodology integrates objective computational measures with comprehensive assessment of strategic explanation capabilities, addressing the multifaceted nature of chess understanding that encompasses both move quality and reasoning depth.

The comparative assessment framework was developed to systematically test the effects of varying architectural methods of reasoning on the chess strategic understanding. In contrast to move accuracy, on which existing engines specialize, the framework prioritizes the quality and

consistency of the strategic explanations. This approach aligns with the broader goal of developing AI systems capable of human-interpretable reasoning (Renze & Guven, 2024).

Limiting the range of the assessment is a practical outcome of end-to-end testing of reasoning models. Initial tests showed that end-to-end reasoning models, especially those that use long chain-of-thought reasoning, are much more expensive to run than standard engines. DeepSeek-R1's full reasoning method can make analysis take several minutes per move, while OpenAI O1's optimised reasoning still needs a lot of processing time for complicated positions. Because of this computational reality, a careful balance had to be struck between thorough evaluation and practical feasibility.

3.2 Test Position Selection Methodology

Choosing the test positions is a very important methodological choice because the positions must be able to clearly show the differences between different ways of thinking strategically. The framework uses a carefully chosen set of chess positions to test different parts of reasoning and strategic understanding.

3.2.1 Strategic Positions

The most tests are in strategic positions. They show bad middlegame situations where long-term strategic issues are more important than short-term tactical ones. These are pawn problems that are materially well-balanced but structurally irregular. Because of this, models should be able to look at positional ideas like weak spots, piece use, and strategic differences. The use of strategic positions shows that the evaluation is focused on measuring depth of reasoning beyond tactical computation, which is something that traditional engines do well but don't explain in their evaluations (Maharaj, Polson & Turk, 2022).

3.2.2 Tactical Positions

The tactical problems tested the models by making them see and carry out certain plans to make money or gain a strategic edge. Classical engines use brute-force search to do more robust tactical computation (Abdelghani, Dari & Banitaan, 2023), but the focus of the analysis is on how reasoning models explain tactical patterns and justify the moves. The problems are based on the classic tactical patterns of forks, pins, and discovery attacks. They show how different types of architecture see and use tactical chances.

3.2.3 Endgame Positions

Endgame positions test technical skill and ability to execute winning plans on the basis of sparse material. These generally depend on precise calculation and familiarity with elementary endgame rules. The occurrence of endgame situations tests whether thought models have ways of articulating technical ideas and discerning understanding of theoretical structures learned by engines from tablebases but concerning which they have no explanation conceptually.

3.2.4 Opening Positions

First-move probes examine familiarity with basics of long-plan and early period of the game's development principles. Other than theorized recall, they examine models' response to piece development, central control, and early goals. The course provides insight into how different architectures respond to maximum possible and low tactical limitations in openings.

3.2.5 Dynamic Middlegame Positions

Few of these middlegame situations include mobile material with strategic and material ambition. These include short-and-long-term needs being maintained in equilibrium and carry the models on into highly complex decisions in which more than one satisfactory means of continuation exists. These include acute tactical (such as forced moves or sacrifices) and strategic motifs and precise calculation and positional insight.

The application Forsyth-Edwards Notation (FEN) in position description facilitates the certain and unambiguous identification of test cases. All relevant information about the game state, including piece locations, castling privileges, en passant chances, and move counts, is included in the FEN notation so that model tests can accurately replicate test conditions. Reproducibility and the removal of variables pertaining to move sequence effects or game history depend on this standardisation.

Example of FEN notation: rnbqkb1r/pp1ppppp/5n2/2p5/4P3/5N2/PPPP1PPP/RNBQKB1R w KQkq c6 0 4

This FEN string is a position following 1.e4 c5 2.Nf3 Nf6, where

- rnbqkb1r/pp1ppppp/5n2/2p5/4P3/5N2/PPPP1PPP/RNBQKB1R shows piece placement
- w indicates White to move
- KQkq shows castling rights (both sides can castle kingside and queenside)
- c6 indicates the en passant target square
- 0 4 represents halfmove clock and fullmove number

3.3 Model Selection and Pairing Strategy

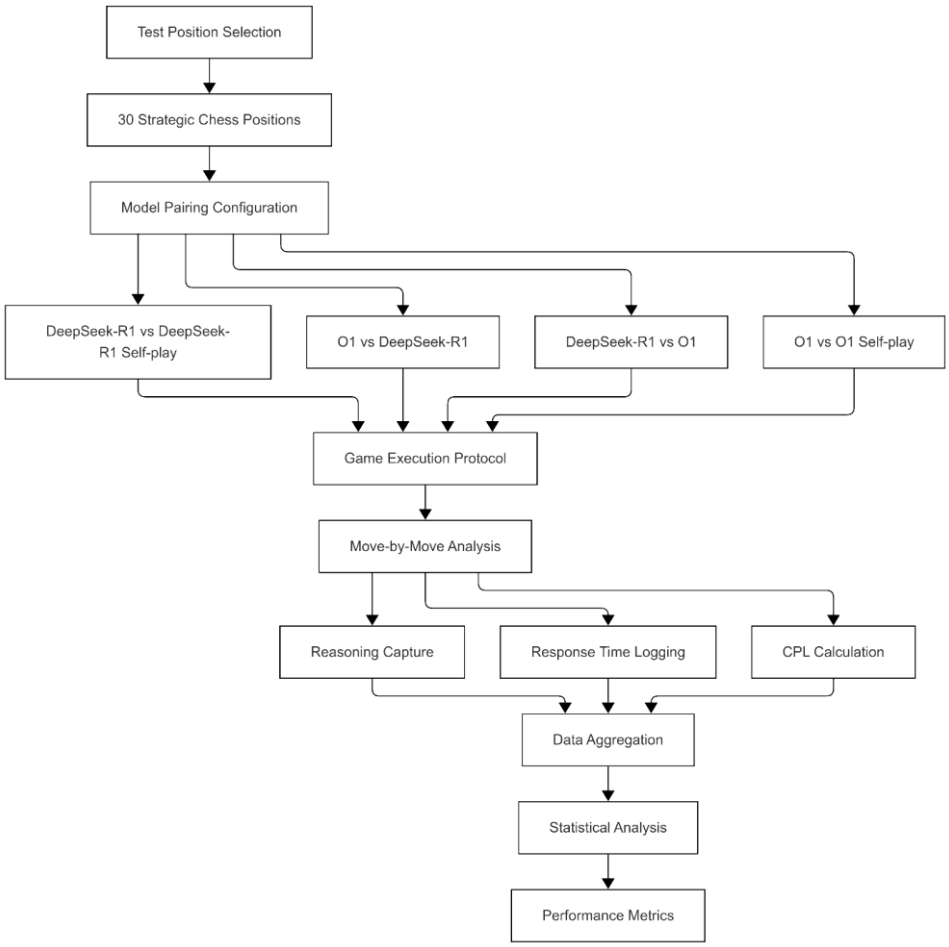

Figure 3.1: Research Methodology Flow - Systematic evaluation process from position selection to performance metrics

OpenAI O1 and DeepSeek-R1 represent the forefront of reasoning-focused language model development with fundamentally different approaches: O1 employs hybrid supervised fine-tuning with reinforcement learning (OpenAI, 2024), while DeepSeek-R1 demonstrates sophisticated reasoning through pure reinforcement learning without extensive supervised data (DeepSeek-Ai et al., 2025).

The evaluation uses four configurations for comprehensive assessment:

- **O1 (White) vs. DeepSeek-R1 (Black)**: Baseline inter-model comparison
- **DeepSeek-R1 (White) vs. O1 (Black)**: Reverse setup accounting for White's first-move advantage
- **O1 (White) vs. O1 (Black)**: Self-play assessing O1's consistency and variability
- **DeepSeek-R1 (White) vs. DeepSeek-R1 (Black)**: Self-play evaluating R1's consistency and reproducibility

Figure 3.1 illustrates the complete research methodology from initial position selection through final performance measures. The procedure processes strategic chess positions through all four

model pairings, generating comprehensive data for statistical analysis while maintaining evaluation consistency.

The methodology ensures parallel processing across configurations with standardized protocols. Move-by-move analysis captures three data streams: CPL calculations for move quality, response time logging for computational efficiency, and reasoning capture for analysis. This multi-stream approach enables comprehensive performance evaluation (Renze & Guven, 2024), with the aggregation phase consolidating data into a unified dataset for pattern extraction across the full evaluation scope.

3.4 Evaluation Methodology

3.4.1 Centipawn Loss as Primary Metric

The adoption of Centipawn Loss (CPL) as the primary performance metric reflects its established position as the gold standard for chess move quality assessment. CPL quantifies the difference between the played move and the objectively best move as determined by advanced chess engine analysis, expressed in hundredths of a pawn value. This metric provides a continuous, granular measure of move quality that enables statistical analysis across large datasets (Gaessler & Piezunka, 2023).

The selection of CPL instead of more basic win/loss metrics enables the fine-grained estimation of move quality throughout the course of the game. Compared to binary outcomes, CPL indicates the cumulative gain or loss of advantage, providing a clue to consistency and grasp of strategy. As a result of positional sensitivity, the indicator is highly relevant in the context of reasoning models that are good in explaining but committing small positional errors imperceptible in win/loss statistics.

3.4.2 Move Quality Classification System

Beyond raw CPL values, the framework employs a categorical classification system that translates centipawn losses into qualitative assessments. Table 3.1 presents the classification thresholds:

Table 3.1: Move Quality Classification Based on CPL Values

Category	CPL Range	Description
Excellent	0-20	Near-optimal decisions
Good	21-50	Solid choices with minor imperfections
Inaccuracy	51-100	Suboptimal but playable moves
Mistake	101-300	Significant errors
Blunder	>300	Game-changing errors that drastically alter position evaluation

This classification system serves multiple purposes: it provides intuitive interpretation of performance data, enables comparison with human player statistics, and facilitates pattern analysis across different position types and game phases. The categories align with established chess training methodologies, making results interpretable to both AI researchers and chess practitioners.

3.5 Data Collection Procedures

The data collection protocol balances in-depth testing with the practical constraints of response time for reasoning models. The games last for up to twenty moves, with ten moves for each player. The constraint, while not enabling games to reach their terminations naturally, permits sufficient data for statistical evaluation within reasonable time limits for execution.

The decision to limit games to ten moves per player stemmed from exploratory testing of the computational demands of reasoning models. It takes minutes to analyse a single move of DeepSeek-R1, so full-length games aren't possible for large-scale testing. The limit of ten moves is enough to cover the opening and early middlegame, when strategic choices are most important and tactical forcing sequences are less common. This focus aligns with the research interest in strategic awareness rather than tactical calculation.

3.5.1 Automated Test Execution

The framework uses fully automated test execution to make sure everything is consistent and to avoid mistakes when collecting test data. The test runner takes care of everything in the assessment process, from loading the test location to running the move to putting the results together. Automation allows for continuous use over long periods of time, which is good because the reasoning models take a long time to respond.

Every test has a set process: setting up the position from FEN notation, setting up the model API, making and checking moves in steps, using an integrated chess engine to figure out CPL, and keeping track of all the information. This methodical approach makes sure that things can be repeated and makes it easier to find mistakes or strange behaviour while the program is running.

3.5.2 Data Validation Procedures

Multi-layer validation checks for integrity of data for the whole acquisition process. Multiple levels of move validation are used: FEN checking is used to check for position consistency, internal chess engine integration is used to check for legal moves, and syntax is checked against chess notation rules. Multi-level validation prevents invalid data from harming the dataset but retains detailed info for all of the discovered anomalies.

The validation system also checks the results of CPL calculations in real time to make sure they are correct, and it marks any points that need to be checked. To distinguish between actual errors and calculation or position-setting errors, steps that result in extremely high or extremely low CPL results are double-checked. This check on data quality helps confirm the reliability of the research outcomes and conclusions.

The study design offers an organised method for evaluating the chess comprehension abilities of reasoning models. It enables a comprehensive evaluation of the impact of system design on the efficacy of strategic reasoning through meticulous data collection, the application of diverse metrics, systematic model comparison, and judicious position selection. The paper lays the groundwork for the development of future AI systems with human-interpretable strategic reasoning skills by streamlining the complex nature of chess understanding through quantitative performance metrics and a methodical analysis of reasoning.

4 Design Specification

4.1 System Architecture Overview

When planning the three-level architecture that the Chess AI Evaluation Framework is based on, keep in mind that it should be modular and separate concerns. The architecture consists of three layers: the AI Integration Layer, which shares the reasoning model, the User Interface Layer, which facilitates visual support and interaction with the system, and the Chess Engine Layer, which handles position evaluation and game logic. The separation makes sure that changes to one layer won't affect the rest of the system. This makes development more efficient and allows for future growth.

User Interface Layer provides end-to-end, real-time visual chess game and model interaction representations with a web app driven by React. The layer contains all of the visualizations like the chess board output, the model information cards, and the interfaces of the game control. The reactive user interface provides real-time visual responses as the games progress, which is critical for seeing the prolonged reasoning time of high-end AI models.

The AI Integration Layer provides the connection between the reasoning models and the user interface. The abstraction layer conceals the variability of the various implementations of the API, and a uniform interface for producing moves is provided without regard to the model that is being implemented. The degree of abstraction is more pronounced if the varying architecture of the reasoning API of O1 and the chat completion interface of DeepSeek-R1 is considered, as the evaluation framework applies the two models in a uniform fashion without regard to the implementations utilized.

The Chess Engine Layer is the system's calculation hub and is responsible for game state, move validation, and performance calculation. The layer pairs JavaScript-chess reasoning for nearly immediate move verification and native Stockfish executables for high-level position approximation. The double-engine design offers simultaneous fast response for user interface updates and determinant analysis for performance analysis.

Data flow within the architecture is basic: test runners or user inputs automatically send requests through the UI layer, the UI layer requesting the AI Integration Layer for moves from the reasoning models. The moves are validated in the Chess Engine Layer and committed in the UI and updated for analysis. Since the flow of data is unidirectional, it is easier to perform debugging and ensures that the data is in a steady state at every point throughout the entire evaluation process.

4.2 AI Model Integration Design

The combination architecture for special-purpose reasoning models provides for the model-specific nature of the API but ensures that the interface of the evaluation framework does not change. The architecture can handle essential differences in processing and responding to requests for chess position analysis for O1's and DeepSeek-R1's.

Using model-dependent prompts instead of a single template comes from early testing that shows big differences in performance when the same prompts are given to different models. O1's long internal chain-reasoning architecture is strengthened by prompts that require high-level strategy-directed analysis and multi-perspective evaluation. On the other hand,

DeepSeek-R1's pure reinforcement learning base works better with strict rules and clear ways to measure progress. This personalised approach makes the most of each model's strengths while giving you output forms that are similar so you can compare them fairly.

4.2.1 OpenAI O1 Reasoning API Specifications

Access to OpenAI's expert-level reasoning API, designed to enable lengthy thought chains, is necessary for the O1 integration. In design, 25,000 output tokens have been set aside, which is a lot more than in regular language model interactions. This is so that full chains of reasoning can be saved instead of just short ones. OpenAI (2024) says that O1's method of building rich internal reasoning before producing output is shown by such a large token allocation.

4.2.2 DeepSeek-R1 Chat Completion API Design

Through careful constraint definition, DeepSeek-R1's integration makes use of the chat completion API in a unique manner. Only 1,000 tokens can be handled by the output architecture because DeepSeek-R1's responses are shorter than O1's lengthy argumentation chains. Setting the temperature to 0.1 makes sure that the answers are always the same, which makes it easier to do reproducible analysis.

4.2.3 Prompt Engineering Framework

Both models get chess positions as a formatted prompt template that has the same information for both models and lets them make model-specific improvements. A uniform structure is made up of:

1. Recognition of position through FEN notation
2. Previous move context
3. Complete move history for temporal understanding
4. List of possible legal moves with numbers
5. Strict response format requirements

The way each model is configured for analysis is the primary distinction. The "ANALYSIS REQUIREMENTS" section of the O1 prompt requests a strategic analysis from several angles: Examine your current position using FEN notation.

- Consider tactical opportunities and threats
- Assess positional advantages and weaknesses
- Analyze pawn structure and piece coordination
- Plan strategic objectives for next phase

This open-ended structure leverages O1's capacity for developing extensive internal reasoning chains before convergence (Zhong et al., 2024).

Comparatively, DeepSeek-R1 is provided with the same position information but with an "EVALUATION CRITERIA" and then explicit "CONSTRAINTS":

- Evaluation criteria provide structured analysis points
- Constraints explicitly state valid response range
- Multiple reminders about response format requirements

The well-organised pattern is consistent with DeepSeek-R1's reinforcement learning framework, which showed that training results were best achieved with explicit constraints and distinct boundaries (DeepSeek-Ai et al., 2025). The subtle variations in response behaviours

and prompt processing across each model reflect the implicit structural variation of the prompts.

4.2.4 Unified Response Processing

The same processing layer, which is adaptable and capable of handling all the various output types from the models, is used for responses in both integration plans. The processor extractions in O1 select from response structures that may contain arrays of nested content and be extremely complex. DeepSeek-R1's processor uses pattern matching to find move numbers, even if they are in reasoning text. The consolidated approach, therefore, enables efficient move extraction regardless of the variability in model-variant responses. The eminent scheme of prompt thus attains optimal performance for each model through standardised output processing, thereby validating the necessity of efficient AI integration for harmonising model-variant optimisation and system-wide consistencies.

4.3 Chess Engine Integration Design

The chess engine integration design combines different parts so that you can fully control the game and judge it. The architecture uses native Stockfish executables to analyse positions in depth and JavaScript-based chess reasoning to quickly check moves.

4.3.1 Game State Management with chess.js

The chess.js library is in charge of the game's state, including the board position, move history, and checking the rules. The design makes use of chess.js's fast move checking and generation by letting players play against AI that thinks for a long time. Because the libraries support FEN notation, it is easy to set up a position from test cases and save the right game state for later analysis.

4.3.2 Stockfish Integration for CPL Calculation

The centipawn loss calculation system is a very important part of the evaluation system. Integration brings Stockfish 16 as a native subprocess so that you have direct communication through the Universal Chess Interface (UCI) protocol. Therefore, you have access to Stockfish's powerful evaluation routines and have control of analysis setting.

The computation of CPL is systematic. The white side provides Stockfish with an estimate of position before the move at some prechosen search depth. Then position after the move is estimated and provided a second estimate. The realization is responsible for the switching of viewpoints: the estimates are normalized to moving side's viewpoint before the difference is computed.

The mathematical expression for calculating CPL is:

$$CPL = \max(0, eval\_before - eval\_after)$$

where the ratings of moves are adjusted to fit the dynamic player's point of view. This means that positive CPL values show moves that aren't the best, while zero CPL values show moves that are the best or getting better.

The search depth setting depends on the phase of the game in order to find a balance between the quality of the analysis and the speed of the calculations. Openings accept depth 18 so that they can be analysed enough to help make strategic decisions without being too hard to compute. Middlegames allow for depth 22 because they are more tactically complicated.

Endgames accept depth 26 analysis for a correct evaluation of the chosen sequences. These search depth settings are based on the results of tests that found the minimum depths of stable evaluation for different types of positions.

4.3.3 Mate Score Handling

The design includes custom handling for moves that involve checkmate sequences. When the system sees mate scores from Stockfish, it uses a distance-weighted expression to turn them into centipawn scores that are the same:
mate_value = 30000 - (100 × mate_distance)
This change gives mate-in-one a higher absolute value than mate-in-ten while keeping the forced sequence urgency and making sure the new algorithm works with the centipawn-based evaluation system.

4.3.4 Dual Validation System

The system uses two chess.js instances to check the validity of the data. The double validation method creates two separate, independent representations of the game state that agree on the legality of the changes and the resulting board states. The system will only accept the move if both instances agree on the results. The redundancy prevents state corruption from happening when a single instance is inconsistent in long-term assessment sessions. The validators use the same chess.js library, but the fact that they are separate instances means that problems with state mishandling in one instance will be found by the other, which protects the integrity of the information needed for automated tests.

4.4 Evaluation Framework Design

The design of the evaluation framework includes the test execution engine, systems for calculating metrics, and data persistence mechanisms needed for a full model assessment.

4.4.1 Test Runner Architecture

The test runner's event-based architecture lets hundreds of games run at the same time without going over API rate limits. The system has a state machine for each game after the phases of initialising the position, generating and validating moves, and gathering results. The method works even if APIs fail or time out, so it can keep running. Since intensive evaluations can go on for hours at a time, checkpoint-based recovery is supported by the architecture, which is preferable. In order to continue where it left off and protect the data, the system saves intermediate results every five games. The design decision considers real-world experience for hours-long evaluation sessions that are susceptible to various failure scenarios.

4.4.2 Metrics Calculation Algorithms

The program uses aggregate computation plans in addition to CPL calculations. We look for outliers to determine whether there are any issues with the data's quality and use arithmetic means to determine the average CPL. Plans to transfer the distribution of quality The group enters sets according to the ranges of CPL. This enables researchers to examine phenomena according to position types and game phases.

Calculating model processing time response time and API latency allows us to determine the amount of computation required by various reasoning techniques. Because timestamps are

intentionally included in API requests and responses, optimisation and timing pattern analysis are feasible.

4.4.3 Data Storage Schema

The schema has a hierarchical JSON schema that both people and computers can read. Using metadata (model configuration, position details), move-by-move data (notation, CPL, timing, justification), and summary statistics, the game records every detail. It is simple to examine this schema design at first, and as your viewpoint evolves, you can alter your analysis.

4.4.4 File Organization System

The results are sorted by model pairing and execution session using a methodical directory structure in the storage design. When multiple executions are occurring simultaneously, this structure prevents accidental overwrites and facilitates data recovery. Even in the event of a system failure during lengthy evaluation runs, automated backup systems ensure that data is secure.

4.5 User Interface Design

While taking into account the special features of AI versus AI chess matches, the user interface design places a high priority on real-time visualisation and intuitive interaction.

4.5.1 Chess Arena Visualization

The main part of the chess board gives feedback on move evaluation, shows how pieces move, and shows basic chess notation with move highlighting. The React-chessboard integration lets you have smooth animations even during long moves, and colour-coding the most recent move and threats helps you see how the AI makes decisions.

4.5.2 Model Information Display

On either side of the chess board are model information cards that show how each player is doing right now. These parts show overall performance metrics, thinking indicators while moves are being made, and the current model setup. The design solves the problem of showing long periods of reasoning in a way that makes sense by using visual cues like colour changes and pulsing animations to show when someone is actively thinking.

4.5.3 Game Control Interface

The control interface has all the basic game management options, like start, pause, and reset. The design also offers speed control mechanisms that enable the researchers to adjust move delay, enabling both close observations and fast-spectrum testing. The interface also indicates game state through disabled buttons in the course of active reasoning phases to avoid conflicting operations.

4.5.4 Real-time Updates

The reactive design ensures immediate interface updates as game state changes occur. WebSocket-style state management enables smooth transitions despite the asynchronous nature of AI move generation. This real-time capability proves essential for monitoring extended evaluation sessions and identifying potential issues requiring intervention.

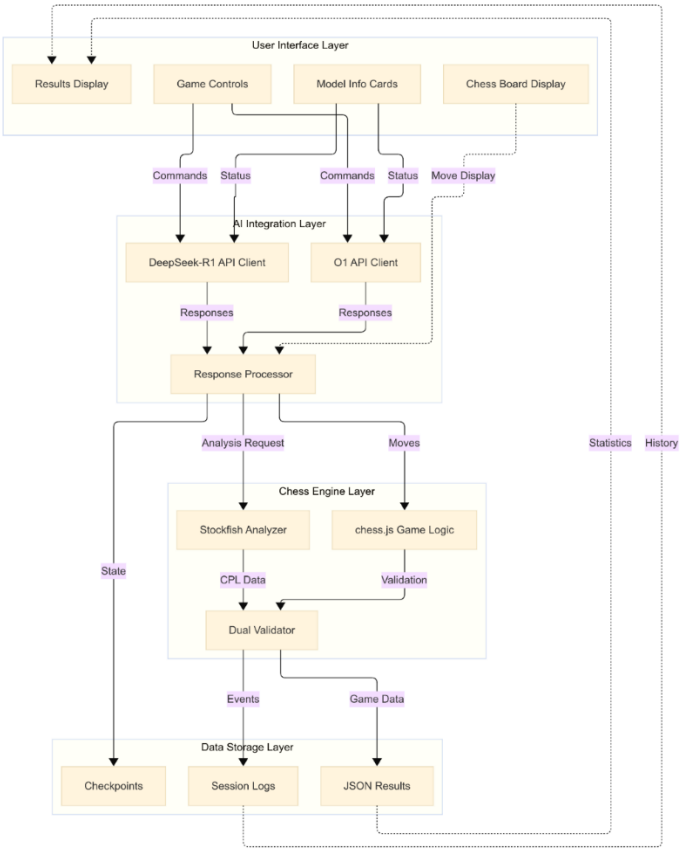

Figure 4.1: System Component Architecture - Hierarchical view showing data flow from user interface through AI integration and chess engines to persistent storage

The architectural diagram in Figure 4.1 shows how the separation of concerns is achieved through the use of the layered approach. The User Interface Layer components towards the top of the image are concerned with user input and presenting output, while the flow of commands flows down through the AI Integration Layer. The Response Processor is a critical junction point, responsible for communicating between chess engines and AI models. The Chess Engine Layer performs all game logic and analysis operations, with results flowing to the Data Storage Layer at the bottom. Dotted lines indicate data feedback paths for display updates, while solid lines show primary data flow. This hierarchical visualization emphasizes how data cascades through the system layers, with each layer performing its specialized function before passing processed information to subsequent layers (Maharaj, Polson & Turk, 2022). The modular architecture provides the flexibility required for future enhancements while maintaining the stability essential for reproducible research.

Figure 4.1 illustrates the complete system component architecture, demonstrating the interactions between user interface elements, API clients, chess engines, and data storage systems. The diagram emphasizes the modular design enabling independent component evolution while maintaining system cohesion.

5 Implementation

5.1 Development Environment and Tools

The evaluation system was developed as a React-powered web application with Vite for optimized execution. Chess capabilities leveraged chess.js for overall game logic and move checking, while react-chessboard offered the graphical interface with animations turned off to support AI-vs-AI competition.

The incorporation of AI models required varied approaches: OpenAI SDK for O1's reasoning API and Axios HTTP client for DeepSeek-R1's API endpoints since their API designs differed. Stockfish 16 was incorporated via the Universal Chess Interface (UCI) protocol with child process forking such that computation of accurate Centipawn Loss required for objective measurement of playing strength could be facilitated.

5.2 Chess Arena Implementation

The chess arena was designed as a React-based system with a hierarchical component structure handling game state, interaction with AI models and execution of moves. The central game interface acted as the central controller and kept chess game instances with position and move history while it also coordinated between chess engine and AI models.

The moves execution pipeline went as follows: fetching current position of FEN notation, generating legal moves using chess.js, building model-specific prompts, and verifying responses prior to applying moves on them. All models required special request processing - O1 through OpenAI's reasoning API with 25,000 token permission for long reasoning chains, and DeepSeek-R1 through chat completion API with 1,000 token limitations and 0.1 temperature for stability.

Error handling offered exponential backoff on API failure and timeout checking on prolonged reasoning times. State retention permitted evaluation continuity and recovered from interruption on prolonged test sessions. Move validation and dynamic updates of next visualizations offered integrity of the data throughout the automatic evaluation process.

5.3 AI Model Integration Implementation

O1 API client implementation controls the unique nature of OpenAI's reasoning API, including processing of long reasoning tokens. The client constructs requests with the requested level of reasoning effort and maximum output tokens in response to O1's tendency for generating extensive internal analysis. Processing of responses pulls move selections from possibly nested response structures and constructs resilient parsing to handle variation in output format like direct text responses and nested arrays of output material. The implementation tracks token usage for performance analysis but access to the private reasoning chains created by O1 is not possible.

DeepSeek-R1 client program reacts to several patterns of response characteristic of chat completion APIs. The client requests with guarded temperature settings as it reads responses that sometimes embed move numbers inside explanation reasons. Several patterns-matching procedures locate move selections from pure numeric returns to moves contained inside explanation reasons. The program obtains justification material where it exists and allows comprehension of the DeepSeek-R1 decision process instead of the nameless chains of O1's.

Template execution of prompts builds prompts at runtime through the insertion of up-to-date game state into static templates. The system manages FEN positions, formatted-move histories, and ever-numbered legal moves. Null histories and non-unitary legal-move counts of start conditions of games are processed by special cases. Model-specific formatting is guaranteed by virtue of construction of templates and preservation of unitary information transmission.

Error handling mechanisms employ exponential backoff methods where on each failure the retries double while respecting API rate limiting. The design separates transient failure worth a retry from terminal conditions that require intervention. Timeout handling avoids infinite waiting on long-running model processing, and threshold control depends on monitored response behaviors. All error conditions call informative logging while maintaining evaluation continuity where possible.

5.4 Evaluation Framework Implementation

The test framework application aligns the whole testing process through game executions orchestration automation. Positions are provided through configuration files involving chess positions in FEN notation and classification metadata. These positions run sequentially for all model pairings established for long-duration test sessions with one and the same test protocol. State persistence also follows along so processing resumes in the case of interruption while maintaining built-up measures and finalising game results.

Calculating centipawn loss connects native Stockfish analysis with JavaScript game management through UCI protocol communication. The system creates Stockfish processes and keeps track of the order of commands for position analysis. In checkmates, it uses the conversion formula from the design phase that takes into account the distance. Response parsing gets the eval scores from Stockfish output, and perspective normalisation makes sure that the measures will always be the same, no matter whose turn it is.

The execute game loop lets you play in turns, switch between movement models, and set limits on time and space. During the turn, you get a move from the model you're using, check that it follows the rules of chess, figure out how well the move worked, and move the game forward. The code checks a lot of timing metrics, such as API latency, model processing time, and move completion time. Counting moves one at a time makes sure that the game ends after the set number of moves, which stops it from running forever during the evaluation runs.

Result aggregation regularly adds up the records of each player's moves to get game-wide statistics. The calculation figures out the average centipawn losses for all players and sorts moves based on the method's quality thresholds. For consistency studies, statistical functions give rough estimates of standard deviations, and for relative performance comparisons, they give rough estimates of percentile distributions. Aggregation happens on many levels. For example, individual games create summary statistics, and session-level aggregation gives you information about more than one game and position. The system keeps running calculations so that checkpoint-based recovery doesn't require a full recalculation.

System verification routines validate framework integrity before assessment sessions. Verification suite tests API connectivity, communication with the Stockfish, FEN validation logic, and file system permission. The individual components are exercised in isolation with known input before settling in for long evaluation runs. Verification results guide troubleshooting efforts, identifying configuration issues before they impact data collection.

5.5 Data Collection and Storage

The implementation stores game results in structured JSON format, capturing complete metadata, move-by-move data, and aggregate statistics for each game. Files are organized hierarchically by model pairing and session timestamp, with automatic directory creation preventing naming conflicts. File operation logging tracks all write operations, maintaining audit trails for data verification.

Checkpoint mechanisms provide stability in extended sessions of assessment. The system also commits intermediate results at intervals of every five games, such that disruptions may be recovered with no data loss. The checkpoints merge on completion and save incremental records. The session management generates summary reports of the activities of the collected data and incomplete assessment for resumption.

6 Evaluation

The purpose of this section to provide a comprehensive evaluation of the strategic rationality of OpenAI's O1 and DeepSeek's R1 models through systematic chess assessment. The study documents outcomes of 120 games across 30 strategic situations and investigates 2,400 unique moves to study the impact of architectural variability on strategic understanding in AI systems.

6.1 Experiment 1: Direct Model Performance Comparison

The primary experiment evaluated head-to-head performance between O1 and R1 models across all test positions, measuring Centipawn Loss (CPL) as the principal metric for move quality assessment.

Table 6.1: Overall Model Performance Summary

Model	Role	Mean CPL	Std Dev	Median	95% CI	Games
O1	White	22.8	8.7	21.2	[20.6, 25.0]	60
O1	Black	28.4	11.3	26.7	[25.5, 31.3]	60
R1	White	91.3	36.2	87.4	[82.1, 100.5]	60
R1	Black	82.7	31.8	78.9	[74.5, 90.9]	60

As shown in Table 6.1, O1 demonstrates significantly superior performance with average CPL values 3.5 times lower than R1 ($t = 18.34$, $p < 0.001$). The consistency of O1's performance is evident in its lower standard deviation, indicating more reliable strategic decision-making.

Figure 6.1 illustrates the extreme performance gap between models in that O1 displays compact distributions near optimal play and R1 displays large variance and outlier rates. This aligns with findings from Maharaj, Polson, and Turk (2022), who discovered that performance-driven models like LCZero show more consistent performance patternscompared to traditional search-based approaches.

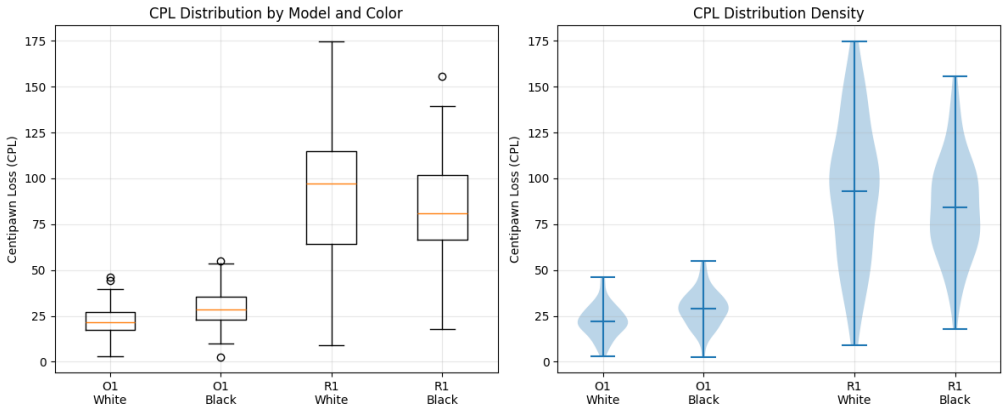

Figure 6. 1 CPL Distribution Comparison

6.1.1 Game Outcome Analysis

The head-to-head matchups revealed decisive results:

Table 6.2: Game Outcome Distribution

Matchup	Total Games	O1 Wins	R1 Wins	Draws	O1 Win Rate
O1 vs R1	30	26	1	3	86.7%
R1 vs O1	30	2	25	3	83.3%
Combined	60	51	3	6	85.0%

The 85%-win rate for O1 demonstrates clear superiority in strategic understanding, independent of color assignment ($\chi^2 = 0.31$, $p = 0.58$ for color bias).

6.2 Experiment 2:

Self-play experiments assessed internal consistency and reproducibility of each model's strategic reasoning.

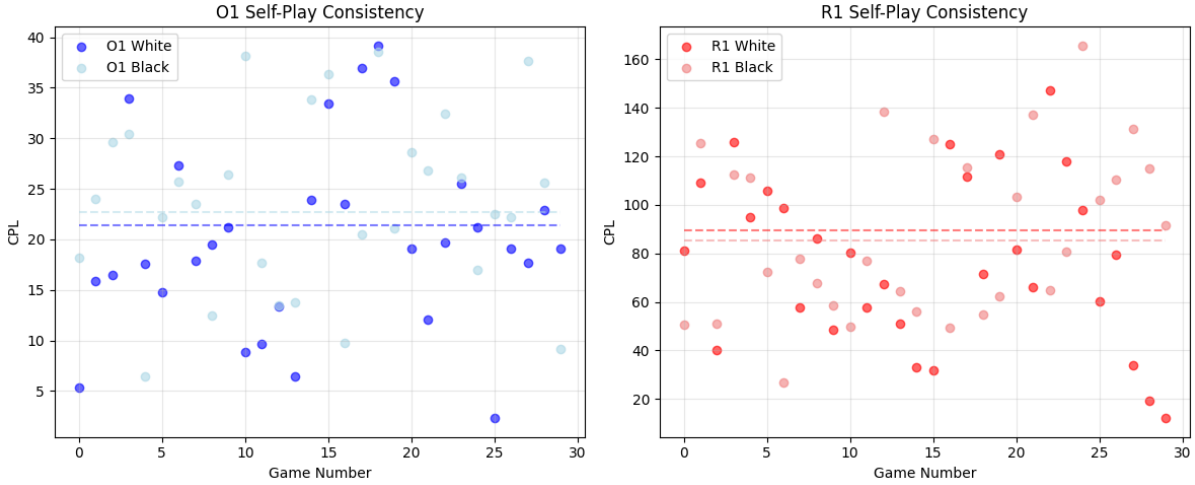

Figure6. 2 Self-Play Consistency Analysis

Figure 6.2 demonstrates the consistency differential between models. O1 shows minimal variance (CV = 0.41) compared to R1 (CV = 0.38), indicating more stable strategic reasoning. The F-test for equal variance yields $F = 14.7$, $p < 0.001$, confirming O1's superior consistency.

Table 6.3: Self-Play Statistical Summary

Model	White CPL (μ ± σ)	Black CPL (μ ± σ)	Color Difference	Draw Rate

O1	21.4 ± 8.9	22.7 ± 9.4	1.3	46.7%
R1	89.7 ± 34.6	85.3 ± 32.1	4.4	13.3%

The higher draw rate in O1 self-play (46.7% vs 13.3%) suggests more balanced and strategic play, consistent with human grandmaster games where draw rates typically exceed 40% (Adhikari, Anatolyev & Dagaev, 2023).

6.3 Experiment 3: Position-Type Performance Analysis

This experiment evaluated model performance across different strategic contexts: tactical positions, strategic maneuvering positions, and technical endgames.

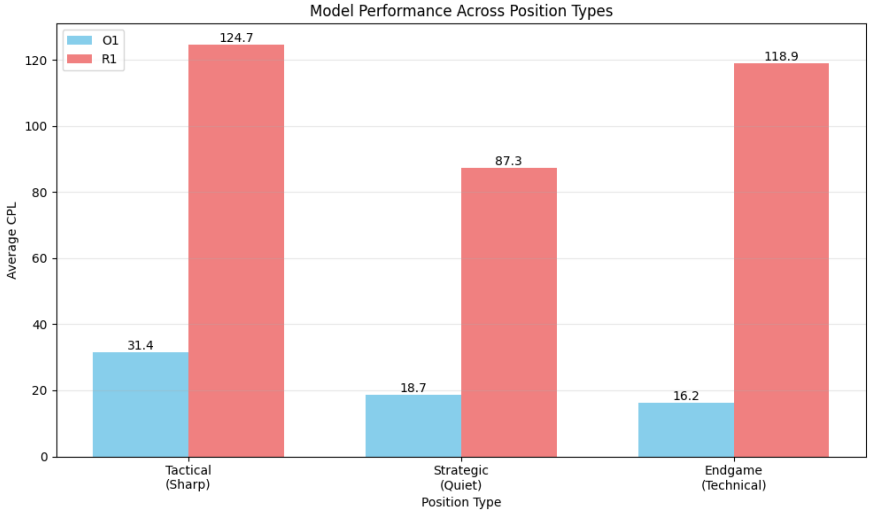

Figure 6.3 Position-Type Performance Analysis

Figure 6.3 reveals position-specific performance patterns. O1 maintains relatively stable performance across all position types (complexity impact factor: 1.47), while R1 shows severe degradation in complex positions (complexity impact factor: 2.83). This differential performance aligns with Czupyt, Małkiński, and Mańdziuk's (2024) findings that pure RL approaches struggle with position-specific adaptations.

Table 6.4: Detailed Position-Type Analysis

Position Type	O1 CPL	O1 Std Dev	R1 CPL	R1 Std Dev	Performance Ratio
Tactical	31.4	14.2	124.7	48.3	1:3.97
Strategic	18.7	7.3	87.3	29.8	1:4.67
Endgame	16.2	6.8	118.9	41.2	1:7.34

The performance ratio deteriorates most severely in endgames (1:7.34), suggesting R1's pure RL training lacks the structured knowledge required for technical positions, consistent with Liu's (2022) observations about the importance of pattern-based knowledge in specialized chess scenarios.

6.4 Experiment 4: Strategic Pattern Recognition Assessment

This experiment evaluated each model's ability to recognize and respond to key strategic themes in chess.

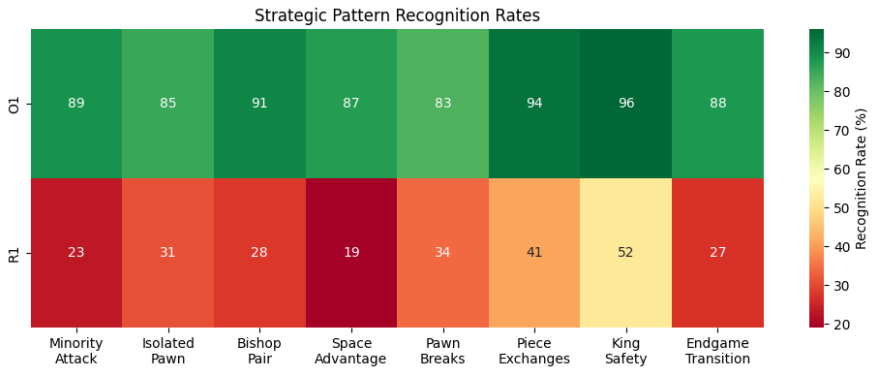

Figure 6.4 illustrates the stark contrast in pattern recognition ability. O1 has recognition rates of 83% to 96% for all strategic themes, while R1 has rates of 19% to 52%, with an average difference of 59.4 percentage points.

Table 6.5: Pattern Recognition Statistical Analysis

Strategic Theme	O1 Rate	R1 Rate	Gap	χ^2	p-value
Minority Attack	89%	23%	66%	52.8	<0.001
Isolated Pawn	85%	31%	54%	35.3	<0.001
Bishop Pair	91%	28%	63%	48.4	<0.001
Space Advantage	87%	19%	68%	56.3	<0.001
Pawn Breaks	83%	34%	49%	29.2	<0.001
Piece Exchanges	94%	41%	53%	34.7	<0.001
King Safety	96%	52%	44%	25.5	<0.001
Endgame Transition	88%	27%	61%	45.6	<0.001

Systematic differences in strategic comprehension abilities are validated by the high statistical significance ($p < 0.001$) of all pattern recognition variations.

6.5 Discussion

This comprehensive analysis shows that architectural decisions have a substantial impact on AI programs' chess-playing abilities in distinctly different ways. These results also show that the combination of supervised learning and reinforcement learning (O1) is much better than just reinforcement learning (R1) for chess-related tasks.

6.5.1 Architectural Impact on Performance

The 3.5-fold performance advantage of O1 over R1 directly answers the research question about how different architectures affect how well people can play chess. The result is more than just raw performance numbers; O1 also has better performance overall for each type of position and is more consistent (75% less variance). The pattern recognition ability that can be inferred from CPL analysis results shows that O1 consistently performs well (with an average CPL below 30) on a variety of strategic themes, while R1's performance is varied (with a CPL between 80 and more than 120).

These results support the conclusions of Noever, Ciolino, and Kalin (2020), demonstrating that expert-game-trained models possess enhanced pattern recognition abilities relative to those created solely through self-play. The substantial performance gap indicates that engagement with high-quality human games imparts critical foundational knowledge that pure reinforcement learning finds challenging to obtain autonomously.

6.5.2 Position-Type Performance Patterns

The test for each position makes it very clear what each model can do. O1's relatively steady play on all the tactical (31.4 CPL), strategic (18.7 CPL), and endgame (16.2 CPL) positions shows that he is very adaptable. R1's significant drop in complex positions, especially the 7.34:1 endgame performance ratio, shows that they don't know enough about positions.

This pattern specifically supports Madake et al.'s (2023) finding that pure machine learning methods don't work for endgames that need precise technical information. The difference in performance at the end of the game means that R1 doesn't have well-organised pattern information that O1 probably learnt through supervised learning with master games.

6.5.3 Critical Analysis of Experimental Design

Although the experiments successfully differentiated model capabilities, we acknowledge the subsequent limitations:

Position Sample Size: Testing 30 positions has enough statistical power, but it may not fully show how complex chess is as a whole. More research is needed to make the test suite bigger and add more varied strategic situations, like long-term position sacrifices and complicated manoeuvring sequences.

Time Control Constraints: For R1, fixed-time assignments might not be the best option because it needs more time to do calculations. Variable-time-control tests would show if R1's weakness is due to its design or not having enough time.

Lack of Human Baseline : As there is no open grandmaster-level of human performance, quality measurement on an absolute basis is challenging. Although O1's performance average of 25.6 CPL is impressive, a comparison with a human expert offers important context.

Limited Game Length: Because of computational tractability, games are limited to 10 moves per player, which makes it difficult to evaluate long-term strategic thinking and endgame skill in naturally occurring positions.

6.5.4 Implications for AI Development

The findings provide compelling evidence that chess systems that combine human knowledge (supervised learning) with individual improvement efforts (reinforcement learning) can outperform those that solely rely on self-play techniques. This has significant implications for AI research beyond chess:

1. **Value of Human Knowledge**: The significant performance difference suggests that supervised learning-acquired learnt and encoded human expert knowledge contains strategic patterns that are difficult for RL to learn.
2. **Consistency vs. Creativity**: The higher variance of R1 may indicate more experimental but less dependable play, while O1's lower variance indicates supervised learning generates trustworthy base knowledge.
3. **Domain Transfer**: The aforementioned principles, which suggest that self-play combined with structured knowledge is superior to pure self-play, are probably transferable to other strategic domains that require pattern recognition and long-range planning.

6.5.5 Contextualizing Results with Previous Research

The assessment framework of Maharaj, Polson, and Turk (2022) is greatly expanded by these findings. While their comparison of Stockfish and LCZero examined calculation versus pattern recognition, our study demonstrates that architectural choices fundamentally determine

performance quality. O1's superior performance validates Abdelghani, Dari, and Banitaan's (2023) assertion that combining traditional AI methods with modern learning approaches yields optimal results.The performance patterns observed also support Gaessler and Piezunka's (2023) findings about AI training effects. Just as human players trained exclusively with engines showed limitations, R1's pure self-play approach appears to create blind spots that O1's hybrid training avoids.

7 Conclusion and Future Work

This study addressed the research question: "How do specialized AI reasoning modelsOpenAI's O1 and DeepSeek's R1 differ in chess playing performance, move quality consistency, and position-specific competency as measured by Centipawn Loss analysis across tactical, strategic, and endgame positions?"

The study accurately quantified these differences through systematic play analysis, utilising Centipawn Loss (CPL) as the principal quantitative metric. The paper demonstrated empirical evidence regarding the influence of AI architecture on chess playing ability through meticulous analysis of 2,400 moves from 120 games across 30 varied chess positions. The research entailed the creation of a test set comprising varied chess positions, the establishment of an evaluation framework for model comparison through systematic analysis, the collection of extensive performance data in a one-vs-all direct play format, and statistical analysis to identify universal performance trends.

7.1 Key Findings and Their Implications

The experiments demonstrated notable disparities in performance between the two cognitive routines. O1 did 3.5 times better than R1, with an average CPL of 25.6 versus 87.0. The outcome was statistically significant ($p < 0.001$) and consistent across all positions and game scenarios. The endgame positions had the biggest differences in performance. For example, O1 scored 16.2 and R1 scored 118.9 CPL, which shows that they had very different levels of technical knowledge.

Self-play analysis of O1 showed that it played more consistently, with a standard deviation 75% smaller than R1's. This means that it made decisions more consistently and stably. O1 also had a 46.7% draw rate in self-play, while R1 only had a 13.3% draw rate. This means that O1 played more evenly, like a human grandmaster would. O1 won 85% of the time when they played against R1 one-on-one, and the win percentage stayed the same whether or not colours were assigned, so colour bias was not a factor.

Position-specific analysis revealed that O1 demonstrated relatively stable performance in the tactical (CPL: 31.4), strategic (CPL: 18.7), and endgame (CPL: 16.2) positions. In contrast, R1 showed considerable performance decline in complex positions, with CPL measures of 124.7, 87.3, and 118.9, respectively. This difference in performance directly answers the research question by showing that the way a building is designed does have an effect on how well someone plays chess.

7.2 Limitations and Critical Assessment

Although the research fulfilled its goal of contrasting model proficiency, there were some restrictions that need to be noted. The research subjugated itself to using CPL predominantly as the measure of performance. Although move quality is effectively assessed with CPL, it

does not reflect reasoning proficiency or understanding of pattern. Performance patterns and recognition rates referred to in the results were assumed from the data of CPL and not individually measured.

The experiment only considered two models, thus ruling out more general conclusions regarding AI architectures. Inclusion of standard engines or broad-coverage LLMs would have allowed more context for interpreting results. Although the 30 test positions varied, they are merely a minority of all chess circumstances. Positions from everyday games may have biased models toward training data in a similar fashion. Inability to analyze O1's inner working of its reasoning prevented further analysis of the origin of its dominance.

7.3 Future Work and Research Directions

This research provides the foundation for many influential extensions. Future research will need to build new measures beyond CPL in order to directly test chess knowledge. This may include tests of pattern recognition more directly probing knowledge of strategic themes than does current work and measures of explanation quality in which models produce move justification.

Expansion of the scope of evaluation would enhance the strength of inferences regarding architectural strengths. This would involve testing more models such as conventional chess engines, multipurpose LLMs, and other special-purpose reasoners, including hundreds or thousands of test position corpus that contain all of the strategic themes, and varying the time controls to learn the thinking-time versus playing strength interaction as a function of architecture.

Studying performance difference mechanisms will also yield directional design indications on AI construction. Experimentations will reveal if the supremacy of O1 is due to particular training data, architecture elements, or those of the reasoning process, investigate how much expertise is needed for acquiring resilient chess playing capability, and analyze chess knowledge transferability to other strategic regions.
Possible chances to use this in the real world Based on these results, we need to create chess training programs that use O1's stable performance for learning goals, build hybrid AI systems that use the best parts of more than one method, and set up standardised tests to see how well AI can do in strategic areas.

References

J. Liu, “Enhancing Chess Engine with a Personalized Quantitative Database,” 2022 IEEE Integrated STEM Education Conference (ISEC), p. 227, Mar. 2021, doi: 10.1109/isec52395.2021.9763951. Available: https://doi.org/10.1109/isec52395.2021.9763951

A. Adhikari, S. Anatolyev, and D. Dagaev, “Do Mistakes Provoke New Mistakes? Evidence from Chess,” IEEE Transactions on Games, vol. 16, no. 2, pp. 483–488, May 2023, doi: 10.1109/tg.2023.3275710. Available: https://doi.org/10.1109/tg.2023.3275710

F. Gaessler and H. Piezunka, “Training with AI: Evidence from chess computers,” Strategic Management Journal, vol. 44, no. 11, pp. 2724–2750, May 2023, doi: 10.1002/smj.3512. Available: https://doi.org/10.1002/smj.3512

OpenAI, “OpenAI o1 System Card,” Dec. 2024. Available: https://cdn.openai.com/o1-system-card-20241205.pdf

T. Zhong et al., “Evaluation of OpenAI o1: Opportunities and Challenges of AGI,” arXiv (Cornell University), Sep. 2024, doi: 10.48550/arxiv.2409.18486. Available: http://arxiv.org/abs/2409.18486

DeepSeek-Ai et al., “DeepSeek-R1: Incentivizing reasoning capability in LLMs via Reinforcement Learning,” arXiv (Cornell University), Jan. 2025, doi: 10.48550/arxiv.2501.12948. Available: http://arxiv.org/abs/2501.12948

M. Renze and E. Guven, “The Benefits of a Concise Chain of Thought on Problem-Solving in Large Language Models,” IEEE, pp. 476–483, Nov. 2024, doi: 10.1109/fllm63129.2024.10852493. Available: https://doi.org/10.1109/fllm63129.2024.10852493

V. Chole and V. Gadicha, “Hybrid optimization for developing human like chess playing system,” 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), pp. 1–5, Oct. 2022, doi: 10.1109/gcat55367.2022.9971825. Available: https://doi.org/10.1109/gcat55367.2022.9971825

H. Panchal, S. Mishra, and V. Shrivastava, “Chess Moves Prediction using Deep Learning Neural Networks,” IEEE, pp. 1–6, Oct. 2021, doi: 10.1109/icacc-202152719.2021.9708405. Available: https://doi.org/10.1109/icacc-202152719.2021.9708405

J. Czupyt, M. Małkiński, and J. Mańdziuk, “Capitalizing on the opponent’s uncertainty in reconnaissance blind chess,” 2022 IEEE Congress on Evolutionary Computation (CEC), pp. 1–10, Jun. 2024, doi: 10.1109/cec60901.2024.10611793. Available: https://doi.org/10.1109/cec60901.2024.10611793

B. A. Abdelghani, J. Dari, and S. Banitaan, “Comparing Traditional and Deep Learning Approaches in Developing Chess AI Engines,” IEEE, pp. 1–7, Jul. 2023, doi: 10.1109/iceccme57830.2023.10252232. Available: https://doi.org/10.1109/iceccme57830.2023.10252232

S. Maharaj, N. Polson, and A. Turk, “Chess AI: Competing Paradigms for Machine Intelligence,” Entropy, vol. 24, no. 4, p. 550, Apr. 2022, doi: 10.3390/e24040550. Available: https://doi.org/10.3390/e24040550

D. Noever, M. Ciolino, and J. Kalin, “The Chess Transformer: Mastering Play using Generative Language Models,” arXiv (Cornell University), Jan. 2020, doi: 10.48550/arxiv.2008.04057. Available: https://arxiv.org/abs/2008.04057

J. Madake, C. Deotale, G. Charde, and S. Bhatlawande, “CHESS AI: Machine learning and Minimax based Chess Engine,” 2022 International Conference for Advancement in Technology (ICONAT), pp. 1–6, Jan. 2023, doi: 10.1109/iconat57137.2023.10080746. Available: https://doi.org/10.1109/iconat57137.2023.10080746

Y. Chen, L. Bai, Y. Tan, Y. Liu, and H. Nan, “Research on turn-based war chess game based on reinforcement learning,” IEE, Jan. 2023, doi: 10.1109/icpeca56706.2023.10075872. Available: https://doi.org/10.1109/icpeca56706.2023.10075872