ARE LARGE VISION LANGUAGE MODELS GOOD GAME PLAYERS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Vision Language Models (LVLMs) have demonstrated remarkable abilities in understanding and reasoning about both visual and textual information. However, existing evaluation methods for LVLMs, primarily based on benchmarks like Visual Question Answering and image captioning, often fail to capture the full scope of LVLMs' capabilities. These benchmarks are limited by issues such as inadequate assessment of detailed visual perception, data contamination, and a lack of focus on multi-turn reasoning. To address these challenges, we propose LVLM-Playground, a game-based evaluation framework designed to provide a comprehensive assessment of LVLMs' cognitive and reasoning skills in structured environments. LVLM-Playground uses a set of games to evaluate LVLMs on four core tasks: Perceiving, Question Answering, Rule Following, and End-to-End Playing, with each target task designed to assess specific abilities, including visual perception, reasoning, decision-making, etc. Based on this framework, we conduct extensive experiments that explore the limitations of current LVLMs, such as handling long structured outputs and perceiving detailed and dense elements. **Code and data will be made publicly available.**

1 INTRODUCTION

Large Vision Language Models (LVLMs) (Liu et al., 2024a; Bai et al., 2023; Zhu et al., 2024) have recently demonstrated remarkable capabilities in processing and generating visual and linguistic information. These models extend the strengths of Large Language Models (LLMs) by incorporating the ability to understand and reason about visual data alongside textual inputs, exhibiting human-like reasoning capabilities across a variety of complex tasks that demand strong reasoning skills.

However, the current evaluation methods for LVLMs vary significantly across different studies, lacking a unified criterion. Most existing LVLM works are assessed using multimodal data established before the LVLM era, such as Visual Question Answering (VQA) (Antol et al., 2015), image captioning (Chen et al., 2015), Optical Character Recognition (OCR) (Singh et al., 2021), etc. While these tasks have been instrumental in assessing model performance, they present several limitations:

- **Inadequate Assessment of Detail Perception.** Existing evaluation tasks, such as generic VQA and image captioning, often fail to effectively assess models' ability to perceive and understand fine-grained visual details. For example, in some VQA datasets, questions can be correctly answered without referencing the accompanying images, relying solely on text-based world knowledge (Chen et al., 2024a). This is because some of these questions are not even genuine visual dependent, and thus they might not fully evaluate the models' visual perception abilities.
- **Risk of Data Contamination.** The extensive training of LVLMs on vast amounts of data from diverse sources increases the risk of data contamination, where training data includes some of the question-answer pairs used for testing (Dong et al., 2024; Zhang et al., 2024; Wei et al., 2023). Such overlap can lead to overestimated performance on tasks like VQA, as models may recall answers from their training data rather than genuinely understanding and reasoning about inputs.
- **Limitations of Metrics.** Existing metrics for evaluating LVLMs on generative tasks such as CIDEr and CLIP Score, heavily rely on short, predefined captions. These metrics are not designed to evaluate the rich and varied outputs that LVLMs can produce, especially for open-ended questions without unique correct answers. This limitation forces many LVLM methods to prompt

the models to generate short answers to fit these metrics (Li et al., 2023), which can restrict the models' ability to showcase their full capabilities.

- **Inconsistent Prompts.** LVLMs often employ specific prompts during training and inference, including zero-shot, few-shot, and Chain-of-Thought (CoT) prompting. These variations can significantly impact performance (Wei et al., 2022; Li et al., 2024b), making it difficult to attribute differences to the models themselves rather than the prompts used.
- **Neglect of Multi-turn/Long-context Reasoning.** Current evaluation benchmarks focus predominantly on single-turn interactions, overlooking the importance of multi-turn reasoning and long-context understanding in LVLMs (Liu et al., 2024b). While real-world applications often require engaging in extended dialogues and maintaining context over multiple exchanges, these capabilities are not adequately assessed by existing benchmarks.

Building upon the limitations of current benchmarks, there is a need for more comprehensive evaluation frameworks that assess the full scope of LVLMs' capabilities. Games, with their structured environments, offer a promising solution. They inherently require not only a detailed perception of dynamic game states but also the ability to formulate strategies, anticipate an opponent's moves, and adapt to new scenarios over multiple turns. This combination of visual understanding, long-term planning, and decision-making under constraints makes games particularly well-suited for evaluating LVLMs across a wide range of cognitive and reasoning tasks. Given that even simple AI algorithms, such as Minimax and Monte Carlo Tree Search, can master games through optimization (Campbell et al., 2002), it raises the question: *Can LVLMs generalize their advanced reasoning and perception skills to perform competitively in these structured, logic-based environments?*

To address this question, we propose a game-based evaluation framework **LVLM-Playground** to assess LVLMs' abilities in structured environments thoroughly. Unlike traditional benchmarks, games provide a controlled yet dynamic setting that naturally tests perception, reasoning, decision-making and competing abilities. This framework directly addresses the key limitations of existing evaluations, as outlined below:

- **Detailed Perception.** Board games like Chess and Go require a precise perception of game states. Models must accurately interpret the positions and identities of pieces, processing fine-grained visual details essential for strategic decision-making.
- **Unique Data.** Game data is largely absent from current LVLM training sets, reducing the risk of contamination. Additionally, diverse environments and varying UI designs in games offer a wide range of novel, procedurally generated scenarios.
- **Transparent Metrics.** Games have clear, rule-based outcomes, such as winning or losing, providing objective metrics for performance evaluation and eliminating the ambiguity often found in current benchmarks that rely on human-annotated answers.
- **Consistent Prompts.** In games, the rules can serve as a uniform prompt, clearly defining what the model should do. This ensures models are evaluated on a level playing field, reducing variability and leading to more consistent, reproducible assessments.
- **Multi-turn Reasoning.** Games naturally require multi-turn interactions and long-term strategic thinking. To perform well, a model must maintain context over several moves, plan ahead, adapt dynamically to changing situations, and anticipate the opponent's moves, making games an ideal testbed for assessing sustained reasoning over extended periods.

In summary, games offer an alternative to existing benchmarks by providing a structured, dynamic, and comprehensive environment for evaluating LVLMs. In this paper, we propose **LVLM-Playground**, a game-based evaluation framework that includes six unique games: Tic-Tac-Toe, Reversi, Minesweeper, Gomoku, Sudoku, and Chess. The contributions of this work are as follows:

- We built LVLM-Playground from scratch, a comprehensive benchmark that integrates game UIs and AI opponents, enabling both online and offline interactions between LLMs and games. The framework supports common interfaces for both commercial models, such as OpenAI API, and open-source models, like those from the HuggingFace Transformers library. LVLM-Playground includes multiple tasks with various settings, as well as automated evaluation mechanisms.

- We systematically designed a comprehensive framework to quantify the abilities required for each game and task, which includes detailed metrics that assess performance across perception, reasoning, decision-making, and adversary skills. Based on this, we evaluated state-of-the-art LVLMs, including both open-source models and commercial APIs, and generated detailed reports that highlight their strength and weaknesses under different gamplay settings.
- Based on the experimental results, we conducted an in-depth quantitative and qualitative analysis, uncovering key findings such as looping behavior in long structured outputs, poor performance in dense visual perception tasks, and potential impairments in instruction-following abilities due to Reinforcement Learning from Human Feedback (RLHF).

2 Related Work

2.1 Evaluation for LVLMs

Evaluating LVLMs has proven challenging due to the need for benchmarks that comprehensively assess both language and visual modalities. Early evaluations (Liu et al., 2024a; Li et al., 2023; Alayrac et al., 2022) often focused on a limited set of vision-language tasks, with benchmarks like VQAv2 (Goyal et al., 2017), VizWiz (Bigham et al., 2010), ScienceQA (Lu et al., 2022), and Text-based VQA (Singh et al., 2019; Wang et al., 2020) being the most commonly used. Other tasks, such as image captioning (Agrawal et al., 2019; Sharma et al., 2018) and image-text retrieval (Plummer et al., 2015), were also employed to evaluate visually-conditioned language generation. However, without a unified evaluation framework, results were often reported on disparate datasets, complicating direct comparisons. In response, later works (Bai et al., 2023; Ye et al., 2024; Li et al., 2024a; Wang et al., 2024c) followed more consistent evaluation practices, relying on a few widely adopted datasets and employing radar charts to visualize model performance across tasks. This approach provided a clearer picture of each model's strengths and weaknesses, though they remained largely focused on VQA and related tasks. These benchmarks themselves were primarily designed for earlier classification-based VQA models, resulting in ground truths that are often short, lack diversity, and do not fully capture the complexities required for evaluating modern LVLMs.

To address the limitations of earlier benchmarks, recent works have developed evaluation frameworks tailored for LVLMs to capture a broader range of multimodal understanding (Wu et al., 2024; Wang et al., 2024b; Lin et al., 2024; Ji et al., 2024). For example, MMT-Bench (Ying et al., 2024) introduces 32 core tasks with 162 sub-tasks, covering areas like 3D perception and anomaly detection. Similarly, VLMEvalKit (Duan et al., 2024a) evaluates over 70 LVLMs across more than 20 benchmarks, unifying model comparisons. However, despite these expansions, many tasks still fall under VQA and its variants, limiting the diversity of benchmarked capabilities. Consequently, multiple benchmarks are often combined to assess a broader range of abilities, which increases the evaluation burden due to the large number of datasets and tasks. In comparison, the proposed LVLM-Playground offers an affordable and efficient alternative, providing comprehensive assessments of multiple capabilities without the need for exhaustive testing across hundreds of sub-tasks.

2.2 LVLM for Games

AI has long been applied to games, with notable achievements like Deep Blue's victory over the world chess champion (Campbell et al., 2002) and AlphaGo's mastery of Go (Silver et al., 2016), while games like Atari have also served as benchmarks for evaluating AI algorithms such as reinforcement learning (Schrittwieser et al., 2020). However, these models were trained for specific tasks with structured rules, often relying on non-visual inputs. In contrast, games with rich visual environments and complex multimodal interactions present greater challenges. Recently, LVLMs have demonstrated strong reasoning capabilities by combining visual perception with language understanding, making them well-suited to act as agents in game environments that demand both modalities for decision-making and interaction (Xu et al., 2024; Zhang et al., 2023). For example, CRADLE (Tan et al., 2024) introduces a flexible framework where LVLMs interact with games and software using screenshots and keyboard/mouse inputs, enabling them to complete complex tasks in games like *Red Dead Redemption 2 (RDR2)* without relying on built-in APIs. Similarly, the VARP agent framework (Chen et al., 2024b) applies vision-language models to action role-playing games like *Black Myth: Wukong*, using visual inputs for tasks such as combat and action planning. While

these works focus on designing LVLM-based agent systems to perform control tasks within game environments, they do not exploit games as a systematic platform for quantitatively evaluating the models themselves. SmartPlay (Wu et al., 2023) and GAMA-Bench (Huang et al., 2024) are more closely aligned with our work, as they pioneer the use of games as a systematic benchmark for evaluating the capabilities of LLMs. However, these benchmarks primarily focus on text-based models, converting all game states into text and evaluating models in a text-only environment. In addition, there are several concurrent works that also leverage game-based environments to evaluate LLMs. GTBench (Duan et al., 2024b) explores the effects of different prompt engineering techniques, such as Chain-of-Thought and Tree-of-Thought, on the reasoning capabilities of LLMs in game scenarios, aiming to enhance their strategic decision-making processes. Similarly, GameBench (Costarelli et al., 2024) uses games as a platform to assess the strategic reasoning abilities of LLMs through various complex tasks. However, these benchmarks primarily focus on text-based models, converting all game states into textual descriptions and evaluating models in a text-only environment. As a result, they do not assess the visual perception capabilities or multimodal reasoning abilities required for real-world interactions. In comparison, the proposed LVLM-Playground provides a more comprehensive evaluation framework by leveraging both visual and textual inputs, allowing for a richer and more realistic assessment of LVLM capabilities in dynamic game environments.

3 LVLM-PLAYGROUND

3.1 OVERVIEW

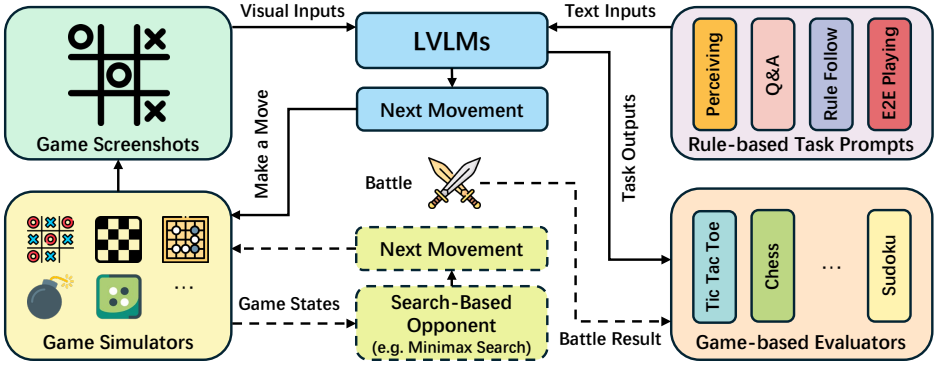


Figure 1: Overview of LVLM-Playground. LVLMs receive visual and textual inputs to perform tasks such as Perceiving, Q&A, Rule Following, and End-to-End Playing in game environments. Dashed lines indicate interactions with a search-based opponent using algorithms like Minimax or Alpha-Beta pruning in competitive games.

Previous research has found that video game playing is associated with cognitive abilities such as spatial visualization, attention control, and visual search strategies (Latham et al., 2013; Han et al., 2011). The ability of individuals, even children, to quickly grasp game mechanics and consistently apply skills related to attention, memory, and problem-solving during gameplay further reinforces this connection (Chaarani et al., 2022). Building upon these insights, LVLM-Playground leverages games to systematically evaluate the diverse cognitive and reasoning abilities of LVLMs. As illustrated in Figure 1, LVLM-Playground consists of a pool of game simulators where LVLMs interact with visual inputs, such as game screenshots, to interpret current game states. Along with task prompts based on game rules, including *Perception*, *Question Answering*, *Rule Following*, and *End-to-End Playing* (detailed in the following sections), the LVLM is expected to output either task-specific responses or the next move in the game. The selected move is then executed in the game simulator, updating the game state and generating new screenshots for further processing. In competitive games, an AI opponent is introduced, typically powered by search algorithms like Minimax to ensure a challenging adversarial environment. The results of each battle or task completion are fed into game-based evaluators, which assess performance based on specific game rules.

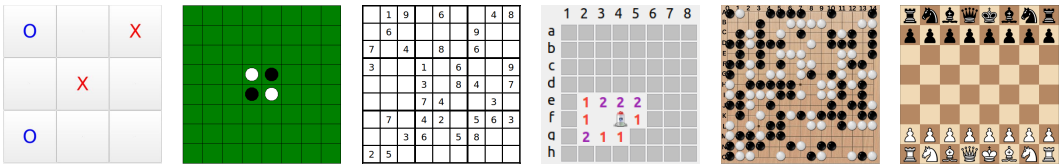


Figure 2: The LVLM-Playground comprises six different games, including Tic Tac Toe, Reversi, Sudoku, Minesweeper, Gomoku, and Chess.

3.2 Game Selection

A key consideration for building LVLM-Playground is selecting appropriate games. While recent works have explored using LVLMs in complex video games like *RDR2*, these games require low-latency, real-time actions, which current LVLMs struggle to handle due to high computational demands. As a result, their application has been largely limited to simplified scenarios, such as fragmented combat sequences, rather than full gameplay. Additionally, using such large games as benchmarks increases implementation complexity and raises the barrier to usage. Therefore, LVLM-Playground focuses on lightweight, turn-based board games that do not require strict real-time decision-making, but feature clear, well-defined rules and significant challenges that demand strong reasoning abilities. Following these principles, we selected six games (see Figure 2) that offer clear win/loss conditions or scoring systems, providing straightforward metrics for performance evaluation: **Tic Tac Toe** involves two players taking turns to place their marks on a 3×3 grid, with the objective of aligning three marks in a row, either horizontally, vertically, or diagonally. **Reversi** features players taking turns placing discs on an 8×8 board, aiming to flip the opponent's discs and control the majority of the board by the game's end. **Sudoku** requires players to fill a 9×9 grid with numbers, ensuring that each digit from 1 to 9 appears exactly once in every row, column, and 3×3 subgrid. **Minesweeper** challenges players to uncover squares on a grid, using numerical clues to avoid hidden mines while revealing safe spots. **Gomoku** is played on a 15×15 grid, where two players compete to be the first to connect five stones in a row, whether horizontally, vertically, or diagonally. **Chess** pits two players against each other on a checkered board, where each piece moves according to specific rules, with the goal of checkmating the opponent's king.

3.3 Abilities

Games have long been used in psychology to assess and enhance cognitive abilities (Von der Heiden et al., 2019; Dale & Green, 2015), with numerous studies establishing a link between gameplay and cognitive development (Choi et al., 2020; Martinez et al., 2023). Even the act of playing a simple game engages multiple cognitive processes (Boot, 2015). Players must **perceive** the game state, **reason** through rules, and make **decisions** to choose the best moves. In competitive games, players also develop **adversarial** skills by anticipating and reacting to their opponent's strategies.

Intuitively, we understand that different games present varying levels of difficulty. More complex games, like chess, require higher levels of reasoning and decision-making compared to simpler games like Tic-Tac-Toe. This is due to factors such as larger game state spaces, more intricate rules, and a greater variety of pieces and possible moves. Building on these factors, and drawing inspiration from previous work in entertainment computing (Aponte et al., 2011; Fraser et al., 2014), we developed a framework to objectively quantify the demands each game places on the four key abilities: *Perception*, *Reasoning*, *Decision*, and *Adversary*.

Taking the relatively simple game of Tic-Tac-Toe (TTT) as an illustrative example, we first review several key factors that influence the cognitive demands of the game. TTT is played on a 3×3 grid, with each cell being either marked as X, O, or left empty, resulting in three possible states per cell. Ignoring turn-by-turn rules for simplicity, the total possible game states can reach up to 3^9. Additionally, there are two distinct piece types, X, and O, and the board size is 3×3. These factors contribute to the perceptual difficulty of the game. Specifically, as the number of possible game states (S), piece types (P), and board size (N) increase, it becomes more challenging to

Table 1: Ratings of the required abilities for each game across four key dimensions.

Ability	Tic Tac Toe	Reversi	Sudoku	Minesweeper	Gomoku	Chess
Perception	☆	★☆	★★★☆	★★★★	★★★★★	★★★☆
Reasoning	☆	★☆	★★	★★★★★	★★★★	★★★
Decision	☆	★★☆	★★	★★★	★★★☆	★★★★★
Adversary	☆	★★☆	N/A	N/A	★★★★	★★★★★

accurately perceive the overall game state. This leads to the following formulation for quantifying the **perception complexity** $\Phi_{\text{perception}}$:

$$\Phi_{\text{perception}} = \alpha_p \log_{10}(S) + \beta_p \log_{10}(P) + \gamma_p \left[\log_{10}(N)\right]^2 . \tag{1}$$

where α_p, β_p, γ_p are coefficients that weight the respective factors in determining the perception complexity. Perceiving the game state is essential, but reasoning goes beyond mere observation. It involves deeper analysis, such as asking: "*How many marks has my opponent placed?*", "*Which cells are critical to control?*", "*Should I focus on defense or plan an attack?*". These considerations require players to process information and answer questions, moving beyond perception. To quantify reasoning difficulty, we consider the number of possible game states (S), the average branching factor (B), which represents the number of available moves at each stage, and the uncertainty factor (U). For example, in TTT, B is 5, as the number of possible moves decreases from 9 as the game progresses. U accounts for hidden or random information; in games like TTT, where all information is visible, $U = 0$, but in games with hidden information like Minesweeper, $U = 1$ due to uncertainty. Thus, **reasoning complexity** is quantified as:

$$\Phi_{\text{reasoning}} = \alpha_r \log_{10}(S) + \beta_r \log_{10}(B) + \gamma_r U. \tag{2}$$

Beyond perception and reasoning, strategic decision-making is essential, encompassing both short-term and long-term considerations. A move made early in the game can influence outcomes much later, linking decision complexity to the average game length (L). In TTT, for instance, games range from 5 to 9 moves, resulting in an average of 7. Additionally, decision-making involves resource management, where players must balance trade-offs, such as sacrificing one asset for a more valuable advantage. Although TTT involves managing only one type of resource (X or O), more complex games like chess require handling multiple types with varying functions, contributing to resource complexity (C). Finally, the average branch factor (B) still plays a role, as more choices at each turn add to the decision-making challenge. Therefore, we quantify **decision-making complexity** as:

$$\Phi_{\text{decision}} = \alpha_d \log_{10}(B) + \beta_d \log_{10}(L) + \gamma_d \log_{10}(C). \tag{3}$$

For non-cooperative multiplayer games, the challenge extends beyond making optimal moves for oneself; it also involves anticipating and countering the opponent's strategies. Adversarial difficulty is hard to quantify, as it depends not only on the player's actions but also on the opponent's level of skill and unpredictability. To simplify this, we approximate the complexity by focusing on the possible interactions between players, where each decision is influenced by potential responses from the opponent. We quantify this by still using the average branching factor (B) and the average game length (L), but multiply them to reflect the steady increase in complexity as players react to each other's moves over the course of the game. Thus **adversary complexity** is approximated as:

$$\Phi_{\text{adversary}} = L \log_{10}(B). \tag{4}$$

Through the above formulations, we quantify the abilities required for each game in LVLM-Playground. The resulting scores are normalized to a 0.5-5 scale, mapped to a star rating system from half a star (☆) to five stars (★★★★★), enabling comparison of the relative difficulty between games (see Table 1). Due to space constraints, please refer to Appendix A for detailed calculations and a human study validating the reasonability of the rating system.

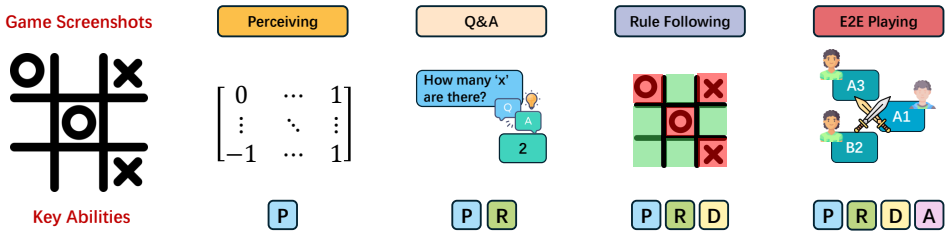


Figure 3: Models are evaluated on various games within the LVLM-Playground framework, with each game consisting of four tasks: Perceiving, Q&A, Rule Following, and E2E Playing. Each task targets one or more abilities, including **P**erception, **R**easoning, **D**ecision, and **A**dversary.

3.4 TASKS

Evaluating models through E2E gameplay can leverage all the four abilities discussed in Section 3.3, yet it is unclear whether the challenges arise from issues in perceiving the game state, following the rules, or other factors. To address this, we break the evaluation into four distinct tasks: *Perceiving*, *Question Answering*, *Rule Following*, and *E2E Playing*. Each task targets one or more of the aforementioned abilities, allowing for a more comprehensive assessment (see Figure 3).

Perceiving tests the model's ability to accurately capture detailed visual elements. The task requires the model to fully transcribe a randomly generated visual game state into structured representations. For example, in TTT, the model receives a screenshot of the board and must convert it into a 3×3 matrix, where 0 represents "O", 1 represents "X", and -1 denotes an empty space.

Question Answering extends the perceiving task by evaluating the model's ability to apply reasoning alongside perception. To align with common practices in recent Q&A-based evaluations, we adopt a multiple-choice format to help decouple instruction-following capability from perception and reasoning abilities. This approach reduces the likelihood of models being penalized for deviating from the expected output format, which allows a more focused evaluation of their core capabilities. Each game includes a set of tailored questions based on specific rules and scenarios, testing a range of abilities. For example, object counting (e.g., *"How many black pieces are on the board?"* in Gomoku), geometrical reasoning (e.g. *"Which cell has been occupied by circular marks?"* in Tic-Tac-Toe), and OCR-like tasks (e.g., *"How many mines are adjacent to the cell at row b, column 1?"* in Minesweeper). For each question, the answer is automatically generated by the game simulator, providing one correct answer along with multiple distractor options tailored to the specific game and question. Typically, four options are provided per question, although for specific question types, such as Yes/No questions, fewer options may be applied.

Rule Following tests the model's ability to internalize and apply game rules to identify valid moves based on the current state. For example, in Chess, pawns move forward one square, while knights move in an "L" shape. Similarly, in Reversi, a valid move must sandwich at least one of the opponent's discs between two of the player's discs. While humans can quickly grasp game rules from a few lines of instructions, models rely on task prompts to internalize these rules. In this task, a random game state is presented, and the model selects the next valid move, which is then executed in the game simulator to verify its compliance with the rules. This process also evaluates the model's in-context learning ability as it adapts to the game rules and makes informed decisions.

End-to-End Playing provides a comprehensive assessment of the model's competence, testing its ability to manage a game from start to finish. This task evaluates the model's overall proficiency across perception, reasoning, decision-making, and adversarial play. In competitive games like Gomoku, the model faces a search-based opponent that employs established strategies such as Minimax search, requiring the LVLMs to make strategic decisions throughout the game. In single-player games like Minesweeper, the model must independently navigate the game, making informed choices based on the current state and available information. By simulating full gameplay, this task holistically measures the model's ability to understand and succeed in complex game environments.

Due to page constraints, we provide additional information in Appendix B, including specific task settings, Q&A question types, and implementations of search-based opponents, among other details.

3.5 EVALUATIONS

To evaluate the models on LVLM-Playground, each task is assessed using specific metrics based on the task and game rules.

Perceiving. The perceiving task assesses the model's ability to convert a visual game state into a structured matrix representation. Accuracy is measured by the proportion of correctly identified elements compared to the ground truth, calculated using the formula:

$$Acc_p = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{I}(P_{ij} = G_{ij}) \tag{5}$$

In this formula, m and n denote the matrix dimensions, while P_{ij} represents the model's output for cell (i, j), and G_{ij} is the corresponding ground truth. $\mathbb{I}(P_{ij} = G_{ij})$ equals 1 when the predicted value matches the ground truth.

Q&A. In the Q&A task, each question is presented in a multiple-choice format with a fixed set of candidate answers. The model's response is matched against the predefined options, and a response is considered correct if it exactly matches the ground truth answer.

Rule Following. The rule-following task evaluates the model's ability to apply game rules to determine valid moves. The board coordinates are specified in task prompts, allowing the LVLMs to output moves in alphanumeric formats such as A1 or B3. The game simulator then verifies whether these proposed moves are valid according to the game rules.

E2E Playing. The E2E playing task evaluates the model's ability to play the game from start to finish. Similar to the rule-following task, the LVLM proposes the next move in alphanumeric format, which is executed in the game simulator. If there is an opponent, the model must consider their responses; otherwise, it plays independently. To prevent indefinite loops, if the model generates three consecutive invalid moves, it is automatically declared a loss. The accuracy of this task is measured by the unbeaten rate, as some games may end in a tie.

To provide a comprehensive comparison of model performance, we compute an aggregated score for each task by factoring in both the difficulty of each game and the specific abilities required for the task. As shown in Table 1, each game $g \in G = \{g_1, g_2, \ldots, g_n\}$ is assigned a star rating $S_{g,a}$ for each ability $a \in A$, which reflects the game's demand on that ability. In addition, Figure 3 illustrates the different combinations of abilities required by each task in $T = \{\text{Perceiving}, \text{Q\&ARule Following}, \text{E2E Playing}\}$. The model's performance on each game-task pair is represented by $M_{g,t}$. To aggregate these scores, we calculate the overall performance for each task using the following formula:

$$OverallScore(t) = \frac{\sum_{g \in G} S_{g,a} \cdot M_{g,t}}{\sum_{g \in G} S_{g,a}} \tag{6}$$

4 EXPERIMENTS

To validate the effectiveness of the LVLM-Playground benchmark and explore current LVLM limitations, we tested both cutting-edge commercial LVLM APIs, including GPT-4o, Gemini-1.5-Pro, and Claude-3.5-Sonnet, as well as widely used open-source models like Qwen-2-VL (Wang et al., 2024a), DeepSeek-VL (Lu et al., 2024), Phi-3-VL (Abdin et al., 2024), LLaVA-1.6 (Liu et al., 2023), and Intern-VL2 (Chen et al., 2024c). All models were evaluated under the same conditions, including identical settings for maximum new tokens and task prompts. In the tables, **bold** values indicate the best performance for each task, <u>underline</u> values highlight the best among open-source models, and **blue** values show performance below a random baseline. The random baseline was calculated by randomly selecting answers from the answer pool, using 1000 gameplays per task, other models run 200 times per task/game.

Table 2: Quantitative results of the perceiving task across different LVLMs.

LVLMs	TicTacToe	Reversi	Sudoku	Minesweeper	Gomoku	Chess
GPT-4o	0.907	0.520	0.550	0.236	0.409	**0.447**
Gemini1.5-pro	**0.993**	0.935	0.591	0.639	0.455	0.360
Claude-3.5-sonnet	0.977	**0.974**	**0.942**	**0.776**	**0.633**	0.296
Qwen2-vl-7b	0.739	0.484	0.580	0.200	0.417	0.272
Deepseek-vl-7b	0.375	0.521	0.281	0.199	0	0
Phi3-vl	0.526	0.497	0.302	0.202	0.006	0.273
LLaVA-1.6-7b	0.374	0.509	0.501	0.202	0	0.277
InternVL2-8b	0.658	0.506	0.401	0.365	0.046	0.199
Random	0.332	0.333	0.103	0.093	0.332	0.079

4.1 Results and Findings

In this section, we present detailed task and game performance of the selected LVLMs. Based on these results, we highlight several key findings that may reveal the limitations of current LVLMs.

Finding 1. Looping Behavior in Long Matrix Outputs. As shown in Table 2, most open-source LVLMs achieved zero or near-zero performance when parsing the Gomoku board. This is primarily due to their inability to produce the output in the required format, specifically a 15×15 matrix. Compared to other games, Gomoku has the largest board size, resulting in a significantly higher number of tokens than in other games. This introduces a challenge where LVLMs often get stuck in a looping behavior, continuously generating matrix numbers without outputting the end-of-sequence token, eventually reaching the maximum token limit. This suggests a broader limitation in current LVLMs' capacity to handle long, structured outputs.

Table 3: Quantitative results of the Q&A task across different LVLMs.

LVLMs	TicTacToe	Reversi	Sudoku	Minesweeper	Gomoku	Chess
GPT-4o	0.235	0.214	0.296	0.231	0.271	0.168
Gemini1.5-pro	0.280	0.277	0.233	0.196	0.226	0.232
Claude-3.5-sonnet	0.216	0.178	0.224	0.204	0.245	0.143
Qwen2-vl-7b	0.655	0.439	0.234	0.273	0.148	0.314
Deepseek-vl-7b	0.304	0.261	0.130	0.355	0.211	0.257
Phi3-vl	0.515	0.314	0.172	0.262	0.127	0.232
LLaVA-1.6-7b	0.387	0.222	0.189	0.197	0.219	0.269
InternVL2-8b	0.675	0.281	0.297	0.188	0.127	0.215
Random	0.256	0.267	0.268	0.281	0.277	0.264

Finding 2. Poor Performance on Detail and Dense Object Perception. Table 2 reveals that LVLMs struggle with games that involve densely packed objects and fine visual details. While most models perform well in simpler games like Tic-Tac-Toe, with Gemini showing nearly no errors, their performance drops significantly on games with slightly larger and more complex boards, such as Gomoku or Chess. This performance disparity underscores the limitations of current LVLMs when dealing with tasks that require high-resolution recognition and handling of dense visual environments.

Finding 3. RLHF May Harm Instruction-Following Ability. As shown in Table 3, the performance of commercial models in Q&A tasks across all games is only comparable to the random baseline. One of the reasons for this outcome is that their outputs often fail to match the specified format, despite the prompts explicitly outlining valid answer options. In some cases, the models even refuse to provide answers. In contrast, open-source models, given the same task prompts, generally adhere to the expected format and achieve higher accuracy. A key distinction lies in the training paradigms of these models. Commercial models are often fine-tuned with Reinforcement Learning from Human Feedback (RLHF), and are optimized for generating contextually rich and conversational responses.

Table 4: Quantitative results of the rule-following task across different LVLMs.

LVLMs	TicTacToe	Reversi	Sudoku	Minesweeper	Gomoku	Chess
GPT-4o	0.805	0.150	0.500	0.530	0.500	0
Gemini1.5-pro	**0.885**	0.180	0.345	0.600	0.535	0
Claude-3.5-sonnet	0.690	**0.220**	**0.715**	0.205	0.410	0
Qwen2-vl-7b	0.630	0.085	0.250	0.465	0.475	0
Deepseek-vl-7b	0.585	0.075	0.090	0.560	**0.545**	0
Phi3-vl	0.505	0.160	0.265	0.530	0.480	0
LLaVA-1.6-7b	0.535	0.155	0.195	0.480	0.205	0
InternVL2-8b	0.650	0.145	0.175	**0.615**	0.485	0
Random	0.342	0.127	0.214	0.422	0.508	0.014

WHile this enhances their ability to engage in natural dialogues, it may not always align well with the structured requirements of tasks like multiple-choice Q&A.

Finding 4. Limited Output Diversity. As shown in Table 4, most models perform significantly worse than the random baseline in the Gomoku rule-following task. This is primarily due to poor perception capabilities on this game and errors accumulating from the visual understanding. As a result, these models tend to generate random moves rather than making informed decisions. However, the gap between their performance and the random baseline suggests that their outputs are not even as diverse as purely random guesses, highlighting the issue in LVLM's ability to maintain diversity and adaptability in their responses.

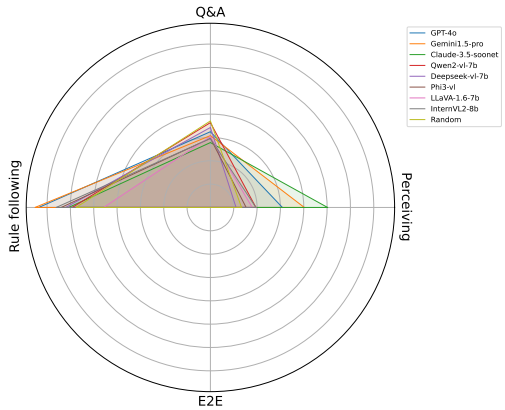


Figure 4: Overall performance of LVLMs on four tasks in the LVLM-Playground.

Finding 5. LVLMs are Stochastic Parrot while Playing Games. Figure 4 demonstrates the overall performance of LVLMs across four tasks in LVLM-Playground, as derived from Equation 6. For the E2E Playing task, all models, regardless of being open-source or commercial, achieved 0 performance because they failed to consistently generate valid movements, even in the simplest game of Tic-Tac-Toe. Despite being prompted to output both observation and strategy to support their proposed movements, these models generate strategies that may seem reasonable but cannot consistently produce valid movements. This inability to maintain coherent gameplay reflects the "stochastic parrot phenomenon", where models echo plausible-sounding outputs without genuine understanding or capability to follow through in the game.

Additional qualitative and quantitative results supporting the above findings are presented in Appendix D.

5 Conclusion

In this paper, we present LVLM-Playground, a game-based evaluation framework for LVLMs that incorporates six diverse games across four tasks, each targeting different core abilities. LVLM-Playground addresses several key limitations of current benchmarks, offering an effective alternative for analyzing and comparing LVLMs from multiple perspectives. Using this framework, we evaluated both open-source models and commercial APIs, uncovering several key findings that highlight the limitations of existing models. In summary, LVLM-Playground provides a novel solution for evaluating the perception, reasoning, and decision-making abilities of LVLMs. We hope that our work will inspire further research and contribute new perspectives to the community.

REFERENCES

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. Nocaps: Novel object captioning at scale. In *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 8948–8957, 2019.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Proc. Advances in Neural Inf. Process. Syst.*, 35:23716–23736, 2022.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 2425–2433, 2015.

Maria-Virginia Aponte, Guillaume Levieux, and Stephane Natkin. Measuring the level of difficulty in single player video games. *Entertainment Computing*, 2(4):205–213, 2011.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al. Vizwiz: nearly real-time answers to visual questions. In *Proc. Annu. ACM Symp. User Interface Softw. Technol.*, pp. 333–342, 2010.

Walter R Boot. Video games as tools to achieve insight into cognitive processes, 2015.

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134 (1-2):57–83, 2002.

Bader Chaarani, Joseph Ortigara, DeKang Yuan, Hannah Loso, Alexandra Potter, and Hugh P Garavan. Association of video gaming with cognitive performance among children. *JAMA network open*, 5(10):e2235721–e2235721, 2022.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *arXiv preprint arXiv:2403.20330*, 2024a.

Peng Chen, Pi Bu, Jun Song, Yuan Gao, and Bo Zheng. Can vlms play action role-playing games? take black myth wukong as a study case. *arXiv preprint arXiv:2409.12889*, 2024b.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 24185–24198, 2024c.

Eunhye Choi, Suk-Ho Shin, Jeh-Kwang Ryu, Kyu-In Jung, Shin-Young Kim, and Min-Hyeon Park. Commercial video games and cognitive functions: video game genres and modulating factors of cognitive enhancement. *Behavioral and Brain Functions*, 16:1–14, 2020.

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. *arXiv preprint arXiv:2406.06613*, 2024.

Gillian Dale and C Shawn Green. Video games and cognitive performance. In *The video game debate*, pp. 131–152. Routledge, 2015.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, and Ge Li. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Proc. Annu. Meet. Assoc. Comput. Linguist.*, 2024.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. *arXiv preprint arXiv:2407.11691*, 2024a.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. Gtbench: Uncovering the strategic reasoning limitations of llms via game-theoretic evaluations. In *Proc. Advances in Neural Inf. Process. Syst.*, 2024b.

James Fraser, Michael Katchabaw, and Robert E Mercer. A methodological approach to identifying and quantifying video game difficulty factors. *Entertainment Computing*, 5(4):441–449, 2014.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 6904–6913, 2017.

Doug Hyun Han, Nicolas Bolo, Melissa A Daniels, Lynn Arenella, In Kyoon Lyoo, and Perry F Renshaw. Brain activity and desire for internet video game play. *Comprehensive psychiatry*, 52 (1):88–95, 2011.

Jen-tse Huang, Eric John Li, Man Ho Lam, Tian Liang, Wenxuan Wang, Youliang Yuan, Wenxiang Jiao, Xing Wang, Zhaopeng Tu, and Michael R Lyu. How far are we on the decision-making of llms? evaluating llms' gaming ability in multi-agent environments. *arXiv preprint arXiv:2403.11807*, 2024.

Huishan Ji, Qingyi Si, Zheng Lin, and Weiping Wang. Towards flexible evaluation for generative visual question answering. *arXiv preprint arXiv:2408.00300*, 2024.

Andrew J Latham, Lucy LM Patston, and Lynette J Tippett. The virtual brain: 30 years of video-game play and cognitive abilities. *Frontiers in psychology*, 4:629, 2013.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proc. Int. Conf. Mach. Learn.*, pp. 19730–19742, 2023.

Zhang Li, Biao Yang, Qiang Liu, Zhiyin Ma, Shuo Zhang, Jingxu Yang, Yabo Sun, Yuliang Liu, and Xiang Bai. Monkey: Image resolution and text label are important things for large multi-modal models. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 26763–26773, 2024a.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *Proc. Int. Conf. Learn. Representations*, 2024b.

Chenwei Lin, Hanjia Lyu, Xian Xu, and Jiebo Luo. Ins-mmbench: A comprehensive benchmark for evaluating lvlms' performance in insurance. *arXiv preprint arXiv:2406.09105*, 2024.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Proc. Advances in Neural Inf. Process. Syst.*, 36, 2024a.

Ziyu Liu, Tao Chu, Yuhang Zang, Xilin Wei, Xiaoyi Dong, Pan Zhang, Zijian Liang, Yuanjun Xiong, Yu Qiao, Dahua Lin, et al. Mmdu: A multi-turn multi-image dialog understanding benchmark and instruction-tuning dataset for lvlms. *arXiv preprint arXiv:2406.11833*, 2024b.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Yaofeng Sun, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*, 2024.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Proc. Advances in Neural Inf. Process. Syst.*, 35:2507–2521, 2022.

Léa Martinez, Manuel Gimenes, and Eric Lambert. Video games and board games: Effects of playing practice on cognition. *PLoS One*, 18(3):e0283654, 2023.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 2641–2649, 2015.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 2556–2565, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 8317–8326, 2019.

Amanpreet Singh, Guan Pang, Mandy Toh, Jing Huang, Wojciech Galuba, and Tal Hassner. Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 8802–8812, 2021.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering foundation agents towards general computer control. *arXiv preprint arXiv:2403.03186*, 2024.

Juliane M Von der Heiden, Beate Braun, Kai W Müller, and Boris Egloff. The association between video gaming and psychological functioning. *Frontiers in psychology*, 10:1731, 2019.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.

Weiyun Wang, Shuibo Zhang, Yiming Ren, Yuchen Duan, Tiantong Li, Shuo Liu, Mengkang Hu, Zhe Chen, Kaipeng Zhang, Lewei Lu, et al. Needle in a multimodal haystack. *arXiv preprint arXiv:2406.07230*, 2024b.

Xinyu Wang, Yuliang Liu, Chunhua Shen, Chun Chet Ng, Canjie Luo, Lianwen Jin, Chee Seng Chan, Anton van den Hengel, and Liangwei Wang. On the general value of evidence, and bilingual scene-text visual question answering. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 10126–10135, 2020.

Xinyu Wang, Bohan Zhuang, and Qi Wu. ModaVerse: Efficiently transforming modalities with llms. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 26606–26616, 2024c.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Proc. Advances in Neural Inf. Process. Syst.*, 35:24824–24837, 2022.

Tianwen Wei, Liang Zhao, Lichang Zhang, Bo Zhu, Lijie Wang, Haihua Yang, Biye Li, Cheng Cheng, Weiwei Lü, Rui Hu, et al. Skywork: A more open bilingual foundation model. *arXiv preprint arXiv:2310.19341*, 2023.

Qiucheng Wu, Handong Zhao, Michael Saxon, Trung Bui, William Yang Wang, Yang Zhang, and Shiyu Chang. Vsp: Assessing the dual challenges of perception and reasoning in spatial planning tasks for vlms. *arXiv preprint arXiv:2407.01863*, 2024.

Yue Wu, Xuan Tang, Tom M Mitchell, and Yuanzhi Li. Smartplay: A benchmark for llms as intelligent agents. *Proc. Int. Conf. Learn. Representations*, 2023.

Xinrun Xu, Yuxin Wang, Chaoyi Xu, Ziluo Ding, Jiechuan Jiang, Zhiming Ding, and Börje F Karlsson. A survey on game playing agents and large models: Methods, applications, and challenges. *arXiv preprint arXiv:2403.10249*, 2024.

Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, and Fei Huang. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 13040–13051, 2024.

Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, et al. Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi. *arXiv preprint arXiv:2404.16006*, 2024.

Chi Zhang, Penglin Cai, Yuhui Fu, Haoqi Yuan, and Zongqing Lu. Creative agents: Empowering agents with imagination for creative tasks. *arXiv preprint arXiv:2312.02519*, 2023.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, et al. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint arXiv:2405.00332*, 2024.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *Proc. Int. Conf. Learn. Representations*, 2024.

Appendix

In this appendix, we provide additional details that were not included in the main text due to page constraints. The appendix is organized as follows:

- **Appendix A Calculation of Ability Ratings**
  - Weighting Coefficients
  - Game Rating Calculations
  - Human Study of Difficulty Ratings
- **Appendix B Detailed Game Settings**
  - Perceiving
  - Question Answering
  - Rule Following
  - End-to-End Playing
- **Appendix C Search-based AI Opponents**
  - Tic-Tac-Toe
  - Sudoku
  - Reversi
  - Minesweeper
  - Gomoku
  - Chess
- **Appendix D Qualitative Results**
- **Appendix E Additional Quantitative Results**
  - Perception Performance vs. Board Complexity
  - Error Analysis of Question Answering
  - Intermediate Metric for E2E Gameplay

A CALCULATION OF ABILITY RATINGS

In section 3.2, we briefly introduced the calculation of ability ratings. Here, we provide detailed steps for calculating the rating values, including how base parameters such as the number of game states (S), branching factor (B), and game length (L) are determined for each game.

*For simplicity, some parameters are **estimated** or **mathematically derived**, including game states that may **NOT** feasible in real gameplay.*

A.1 WEIGHTING COEFFICIENTS

Here, we present the weighting coefficients used in the ability rating formulas in Sec 3.2.

- Perception: α_p = 0.8, β_p = 1.5, γ_p = 1.2
- Reasoning: α_r = 1.0, β_r = 1.0, γ_r = 1.0
- Decision: α_d = 1.0, β_d = 1.0, γ_d = 1.0

A.2 GAME RATING CALCULATIONS

Tic Tac Toe.

- **State Space Size (S) = 3^9.** The Tic Tac Toe board is a 3×3 grid, with each cell having 3 possible states (empty, X, O). Therefore, the total state space size is $S = 3^9 = 19,683$.
- **Piece Types (P) = 2**: There are two distinct piece types in the game, X and O.
- **Board Size (N) = 9**: The board has $3 \times 3 = 9$ cells.
- **Average Branching Factor (B) = 5**: Initially, the player has 9 possible moves (since all cells are empty), and this number decreases as the game progresses. The average branching factor is approximated as $B = \frac{9+1}{2} = 5$.
- **Game Length (L) = 7**: The shortest game ends in 5 moves, while the longest game can take up to 9 moves. The average game length is approximately $G = \frac{5+9}{2} = 7$.
- **Information Uncertainty Factor (U) = 0**: Tic Tac Toe is a perfect information game with no hidden states or randomness.
- **Resource Complexity (C) = 1**: Players only manage one type of resource—their own pieces (X or O).
- **Multiplayer (is_multiplayer) = True**: Tic Tac Toe is a two-player game.

$$\Phi_P^T = 0.8\log_{10}(3^9) + 1.5\log_{10}(2) + 1.2\left[\log_{10}(9)\right]^2 = 5.5057$$
$$\Phi_R^T = \log_{10}(3^9) + \log_{10}(5) + 0 = 4.9931$$
$$\Phi_D^T = \log_{10}(5) + \log_{10}(7) + \log_{10}(1) = 1.5441$$
$$\Phi_A^T = 7\log_{10}(5) = 4.8928$$

Minesweeper.

- **State Space Size (S) $\approx 10^{81}$**: Minesweeper uses a 9×9 grid with 10 hidden mines. The total state space size is estimated considering all possible mine placements and grid states.
- **Piece Types (P) = 10**: There are 10 possible piece types representing different grid states: numbers 1–8 (indicating adjacent mines), mines, and unrevealed cells.
- **Board Size (N) = 81**: The board contains $9 \times 9 = 81$ cells.
- **Average Branching Factor (B) = 41**: At the start of the game, all 81 cells are unopened. The average branching factor across the game is approximately $BF = 41$.
- **Game Length (L) = 50**: On average, players perform about 50 actions in a typical game. This includes opening cells and marking mines.

- **Information Uncertainty Factor (U) = 1**: Minesweeper is a game of hidden information, as the mine locations are initially unknown to the player, making it a partially observable game.
- **Resource Complexity (C) = 1**: Players primarily manage one resource: marking or opening cells, which represents the resource complexity.
- **Multiplayer (is_multiplayer) = False**: Minesweeper is a single-player game with no interaction between players.

$$\Phi_P^M = 0.8\log_{10}(10^{81}) + 1.5\log_{10}(10) + 1.2\left[\log_{10}(81)\right]^2 = 85.6423$$
$$\Phi_R^M = \log_{10}(10^{81}) + log_{10}(41) + 1 = 83.6128$$
$$\Phi_D^M = \log_{10}(41) + \log_{10}(50) + \log_{10}(1) = 3.3118$$
$$\Phi_A^M \text{is not available.}$$

Gomoku.

- **State Space Size (S) =** 3^{225}: Gomoku is played on a 15×15 grid, where each of the 225 intersections can be either empty, occupied by a black stone, or occupied by a white stone.
- **Piece Types (P) = 2**: There are two types of pieces in Gomoku: black stones and white stones.
- **Board Size (N) = 225**: The board consists of $15 \times 15 = 225$ intersections.
- **Average Branching Factor (B) = 113**: The initial move offers 225 possible actions, and the number decreases as the game progresses. The average branching factor is approximately $BF = \frac{225+1}{2} = 113$.
- **Game Length (L) = 45**: While the maximum number of moves is 225, games typically end much sooner once a player aligns five pieces in a row. The average game length is around $L = 45$.
- **Information Uncertainty Factor (U) = 0**: Gomoku is a perfect information game, meaning all information about the game state is visible to both players, with no hidden information or randomness.
- **Resource Complexity (C) = 1**: Each player manages a single type of resource, their own pieces (black or white stones).
- **Multiplayer (is_multiplayer) = True**: Gomoku is a two-player competitive game where players alternate turns, trying to outmaneuver their opponent by forming five consecutive stones in a row.

$$\Phi_P^G = 0.8\log_{10}(3^{225}) + 1.5\log_{10}(2) + 1.2\left[\log_{10}(225)\right]^2 = 113.1861$$
$$\Phi_R^G = \log_{10}(3^{225}) + log_{10}(113) + 0 = 109.4054$$
$$\Phi_D^G = \log_{10}(113) + \log_{10}(45) + \log_{10}(1) = 3.7063$$
$$\Phi_A^G = 45\log_{10}(113) = 92.3885$$

Sudoku.

- **State Space Size (S)** $\approx 9^{81}$: Sudoku is played on a 9×9 grid with each cell filled with a digit between 1 and 9.
- **Piece Types (P) = 9**: There are nine different types of pieces, represented by the digits 1-9.
- **Board Size (N) = 81**: The Sudoku board contains $9 \times 9 = 81$ cells.
- **Average Branching Factor (B) = 5**: While each empty cell can theoretically hold any digit from 1 to 9, the Sudoku rules limit the options. On average, a player has around 5 possible choices per empty cell.

- **Game Length (L) = 50**: Depending on the difficulty level, the game starts with some cells pre-filled. On average, a player needs to fill around 50 cells to complete the puzzle.
- **Information Uncertainty Factor (U) = 0**: Sudoku is a perfect information game, where all clues are visible on the board, and no hidden or random elements are involved.
- **Resource Complexity (C) = 1**: Players only manage one resource, the digits to be filled into the empty cells.
- **Multiplayer (is_multiplayer) = False**: Sudoku is a single-player puzzle game, where players work independently to solve the puzzle without interacting with others.

$$\Phi_P^S = 0.8\log_{10}(9^{81}) + 1.5\log_{10}(9) + 1.2\left[\log_{10}(81)\right]^2 = 81.8902$$
$$\Phi_R^S = \log_{10}(9^{81}) + log_{10}(5) + 0 = 77.9926$$
$$\Phi_D^S = \log_{10}(5) + \log_{10}(50) + \log_{10}(1) = 2.3979$$
$$\Phi_A^S \text{is not available.}$$

Chess.

- **State Space Size (S) =** $\approx 13^{64}$: Chess is played on an 8×8 board with 64 squares, each of which can be empty or occupied by one of 12 different types of pieces (6 types, each in two colors).
- **Piece Types (P) = 12**: There are six distinct types of pieces in Chess (King, Queen, Rook, Bishop, Knight, Pawn), with each having two colors (black and white), for a total of 12 types.
- **Board Size (N) = 64**: The Chessboard contains $8 \times 8 = 64$ squares.
- **Average Branching Factor (B) = 35**: On average, a player has around 35 legal moves available per turn, though this can vary significantly depending on the game phase (opening, midgame, or endgame).
- **Game Length (L) = 80**: A typical Chess game consists of about 40 moves per player, for a total of approximately 80 moves.
- **Information Uncertainty Factor (U) = 0**: Chess is a perfect information game where all pieces and possible moves are visible to both players, and there is no element of randomness or hidden information.
- **Resource Complexity (C) = 6**: Each player manages six different types of pieces, each with unique movement rules and strategic roles, contributing to higher resource complexity.
- **Multiplayer (is_multiplayer) = True**: Chess is a two-player game, where players alternate turns, making it a multiplayer, competitive environment.

$$\Phi_P^C = 0.8\log_{10}(13^{64}) + 1.5\log_{10}(12) + 1.2\left[\log_{10}(64)\right]^2 = 75.6338$$
$$\Phi_R^C = \log_{10}(13^{64}) + log_{10}(35) + 0 = 72.8364$$
$$\Phi_D^C = \log_{10}(35) + \log_{10}(80) + \log_{10}(6) = 4.2253$$
$$\Phi_A^C = 80\log_{10}(35) = 123.5254$$

Reversi.

- **State Space Size (S) =** 3^{64}: The Reversi board is an 8×8 grid with 64 squares. Each square can be empty, occupied by a black disc, or a white disc.
- **Piece Types (P) = 2**: There are two types of pieces in Reversi: black and white discs.
- **Board Size (N) = 64**: The board contains $8 \times 8 = 64$ squares.
- **Average Branching Factor (B) = 10**: Players typically have around 10 possible moves per turn on average, although this can vary throughout the game. Initially, there are fewer available moves, while mid-game offers more opportunities.

- **Game Length (L) = 60**: Reversi games generally last for about 60 moves in total, as the board has 64 squares, but not all squares are necessarily filled.
- **Information Uncertainty Factor (U) = 0**: Reversi is a perfect information game with no hidden information or randomness.
- **Resource Complexity (C) = 1**: Each player manages one type of disc (either black or white). There are no other resources to consider.
- **Multiplayer (is_multiplayer) = True**: Reversi is a two-player game, where players alternate turns placing their discs and flipping the opponent's discs.

$$\Phi_P^R = 0.8\log_{10}(3^{64}) + 1.5\log_{10}(2) + 1.2\left[\log_{10}(64)\right]^2 = 34.0991$$
$$\Phi_R^R = \log_{10}(3^{64}) + log_{10}(10) + 0 = 31.5358$$
$$\Phi_D^R = \log_{10}(10) + \log_{10}(60) + \log_{10}(1) = 2.7782$$
$$\Phi_A^R = 60\log_{10}(10) = 60.0$$

Next, we normalize the calculated ability scores to fit within a 0.5 to 5-star range. This is done by first identifying the minimum and maximum raw scores for each ability dimension across all games:

- **Perception:** $\Phi_{\text{min}}^P = \Phi_P^T = 4.9795$, $\Phi_{\text{max}}^P = \Phi_P^G = 113.1861$
- **Reasoning:** $\Phi_{\text{min}}^R = \Phi_R^T = 4.9931$, $\Phi_{\text{max}}^R = \Phi_R^G = 109.4054$
- **Decision:** $\Phi_{\text{min}}^D = \Phi_D^T = 1.3979$, $\Phi_{\text{max}}^D = \Phi_D^C = 4.2253$
- **Adversary:** $\Phi_{\text{min}}^A = \Phi_A^T = 4.8928$, $\Phi_{\text{max}}^A = \Phi_A^C = 123.5254$

Table 5 summarizes the normalized scores for each game across the four ability dimensions.

Game	Perception	Reasoning	Decision-Making	Adversary
Tic Tac Toe	0.5	0.5	0.5	0.5
Minesweeper	3.86	5.0	3.08	N/A
Gomoku	5.0	4.22	3.57	3.83
Sudoku	3.70	1.82	1.92	N/A
Chess	3.45	2.87	5.0	5.0
Reversi	1.72	1.39	2.42	2.62

Table 5: Normalized ability scores for each game across the four dimensions.

We then map the normalized scores, after rounding, to the 5-star system shown in Table 1, providing a comparison of the cognitive and reasoning demands across different games and abilities.

A.3 Human Study of Difficulty Ratings

To validate the reasonableness of the computed ratings, we conducted a human-based evaluation by recruiting 10 volunteers with varying levels of familiarity with the selected games. Each volunteer was asked to rank the relative difficulty of the six games based on their gameplay experiences. This ranking provided a comparative difficulty chain from the easiest to the most challenging game.

We then summed the normalized difficulty scores from Table 5 to generate an overall difficulty score for each game. For games where certain ability scores were marked as N/A, such as Minesweeper and Sudoku in the *Adversary* dimension, we assigned a minimal placeholder score of 0 to ensure consistent calculations across all games. This approach allowed us to rank the games based on their overall computed difficulty, producing a difficulty chain as follows:

Tic Tac Toe $<$ *Reversi* $<$ *Sudoku* $<$ *Minesweeper* $<$ *Gomoku* $<$ *Chess*.

For the human-based evaluation, the rankings from all 10 volunteers were averaged to create a consolidated ranking for each game. Each volunteer's ranking was based on their perception of the overall difficulty, with the easiest game assigned a score of 1 and the most challenging game

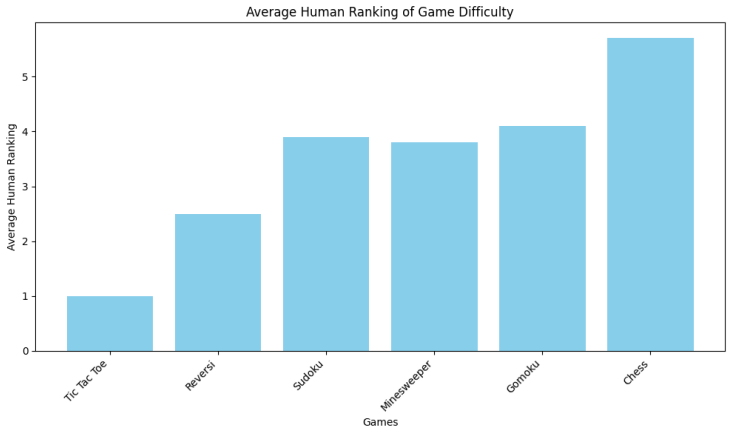


Figure 5: Average Human Ranking of Game Difficulty

assigned a score of 6. We then calculated the average ranking for each game across all volunteers, as demonstrated in Figure 5. The comparison between the computed difficulty chain and the human-based rankings showed a high degree of consistency, supporting the reasonableness of our computed difficulty scores. We also observed that the average human rankings for Sudoku, Minesweeper, and Gomoku were very close. This suggests that, while these games differ in mechanics, they are perceived as similarly challenging. However, each of these games emphasizes different cognitive abilities which could explain the subtle differences in difficulty experienced by different players.

To further validate the evaluation settings, particularly the difficulty levels of each game across the four dimensions (perception, reasoning, decision-making, and adversary), we conducted an additional human study involving 10 volunteers.

To ensure consistency and provide the volunteers with a clear understanding of each task, we sampled 10 representative examples for each game from the LVLM-Playground simulator, covering perceiving, question-answering, and rule-following tasks. Volunteers were asked to perform these tasks themselves to gain a firsthand sense of the difficulty of each task. Additionally, to assess the adversary skills required for each game, volunteers participated in competitive matches against the search-based AI opponent in the LVLM-Playground. Since each volunteer varied in their familiarity with the games, we provided detailed game rules and guidelines beforehand to help them understand the rules and goals of each game.

Finally, we asked the volunteers to rank the four capabilities we defined for each game. The aggregated feedback is summarized as follows.

	Tic-Tac-Toe	Reversi	Sudoku	Minesweeper	Gomoku	Chess
Perception	1.00	2.20	4.60	4.50	4.80	3.90
Reasoning	1.00	2.10	3.90	4.70	4.50	4.80
Decision	1.20	2.50	2.30	4.70	4.90	5.40
Adversary	1.00	2.50	N/A	N/A	2.60	3.90

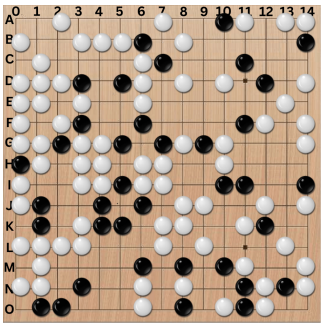
As shown in the table, the human study of relative difficulty among different games demonstrates a similar trend as in the overall evaluation. Although the results do not perfectly align with the quantitative ratings of the required abilities for each game described in the paper, the trends are consistent, further validating the robustness of our evaluation framework.

B Detailed Game Settings

In the main text, we provided a brief introduction to the four tasks included in the LVLM-Playground. Here, we provide more details.

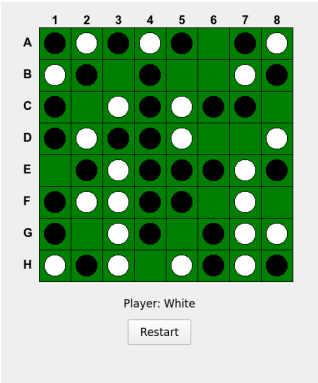
B.1 PERCEIVING

(a) Chess

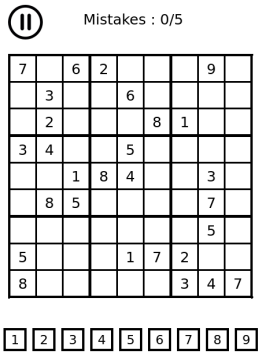


(b) Gomoku

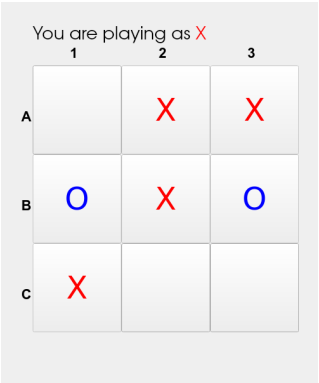
(c) Minesweeper



(d) Reversi



(e) Sudoku



(f) Tictactoe

Figure 6: Screenshots of randomly generated game states. Since these are for perception purposes only, the game rules are not guaranteed, and the game states may never occur in real games.

As shown in Figure 6, the *Perceiving* task is specifically designed to evaluate the model's ability to observe and interpret visual information from game boards, without following the actual rules of the games. The game simulator generates a wide variety of random game states, some of which may **NEVER** occur in a real gameplay scenario. For example, in Figure 6a, the chessboard shows an unrealistic scenario where there are multiple queens of the same color, which is not possible under standard chess rules. This randomness introduces complexity and variety, allowing the model to focus solely on its capacity to interpret the visual aspects of the game boards.

In general, the models are asked to convert the image game state into a structured format, such as a matrix to represent the game board. For example, in the *Reversi* game, the model is expeceted to output an 8×8 matrix where black pieces are represented by 1, white pieces by 2, and empty spaces by 0, accurately capturing the current game state from the visual input. Below is an example of the 8×8 matrix representing the game state shown in Figure 6d:

$$\begin{bmatrix} 1 & 2 & 1 & 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 & 0 & 0 & 2 & 1 \\ 1 & 0 & 2 & 1 & 2 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 2 & 0 & 0 & 2 \\ 0 & 1 & 2 & 1 & 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 & 1 & 0 & 2 & 0 \\ 1 & 0 & 2 & 1 & 0 & 1 & 2 & 2 \\ 2 & 1 & 2 & 0 & 2 & 1 & 2 & 1 \end{bmatrix}$$

The *Perceiving* task, while simple for humans, requires models to achieve high precision in local perception. For humans, interpreting a game board and translating it into structured data is a straightforward process, often done with little effort. However, for models, this task demands precise handling of the visual input and may involve capabilities like OCR to correctly interpret the elements on the board, particularly in games that include alphanumeric labels such as Sudoku or Minesweeper. Moreover, this task emphasizes the model's ability to accurately capture fine-grained details from the image and map them correctly into a structured format. Small errors in perception can lead to significant inaccuracies in the resulting matrix, which demonstrates the importance of the model's ability to perform precise local perception.

All models are provided with the same task prompt for each game, ensuring that they are operating under a fair and controlled environment. This consistency allows for direct comparison of their performance on the same task, without any variations in the instructions they receive.

Below are the task prompts provided to the models for each game:

Perceiving Task Prompt for Chess:

You are provided with an image of a chessboard, and your task is to represent the current state of the game as an 8x8 matrix using the specified numerical format. Each type of chess piece, both black and white, is represented by a unique number:

- Empty squares: 0
- White pieces: Pawn=1, Knight=2, Bishop=3, Rook=4, Queen=5, King=6
- Black pieces: Pawn=-1, Knight=-2, Bishop=-3, Rook=-4, Queen=-5, King=-6

From the provided chessboard image, convert the visible board into this 8x8 matrix format. For example, a typical chessboard configuration would be represented as follows:

$$\text{Game State: } \begin{bmatrix} -4 & -2 & -3 & -5 & -6 & -3 & -2 & -4 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 4 & 2 & 3 & 5 & 6 & 3 & 2 & 4 \end{bmatrix}$$

Ensure that your output strictly follows this matrix format with no deviations, based on the pieces shown in the image.

Perceiving Task Prompt for Gomoku: Gomoku Game State Task:

Given a screenshot of a random 15x15 Gomoku board, please represent the current game state as a 15x15 matrix. Use 1 to represent black stones, 2 to represent white stones, and 0 to represent empty intersections. Example format:

$$\text{Game State: } \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \dots & 2 & \vdots \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

This represents a simplified version of a 15x15 board where 1 is a black stone, 2 is a white stone, and 0 is an empty space.

Perceiving Task Prompt for Minesweeper:

Minesweeper is a logic-based puzzle game played on an 8x8 grid. Each cell can either contain a mine (represented by 9), or it can be empty. Unrevealed cells should be represented by -1. Cells that are revealed and contain no adjacent mines are represented by 0, while cells that are revealed and show a number from 1 to 8 indicate how many adjacent mines surround the cell. Mines are represented by the number 9. Please strictly follow the format:

Game State: `<boardmatrix>`

where `<boardmatrix>` is an 8x8 matrix representing the game grid, with unrevealed cells as -1, mines as 9, and numbers from 0 to 8 indicating the number of adjacent mines. For example:

$$\text{Game State: } \begin{bmatrix} -1 & 1 & 1 & 2 & -1 & -1 & -1 & 1 \\ 1 & 2 & 3 & 2 & -1 & 1 & 1 & 2 \\ 2 & 3 & 4 & 3 & 1 & 1 & 1 & 2 \\ 1 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 9 & 1 & 0 \\ -1 & 1 & 2 & 3 & 2 & 1 & 0 & -1 \\ -1 & -1 & 1 & 2 & 3 & 1 & -1 & -1 \\ 1 & 2 & 9 & 1 & 1 & -1 & -1 & -1 \end{bmatrix}$$

This example represents a grid where some cells have been uncovered, showing numbers indicating adjacent mines, unrevealed cells are represented by -1, and mines are represented by the number 9.

Perceiving Task Prompt for Reversi:

Reversi (also known as Othello) is a strategy board game played on an 8x8 grid, where two players take turns placing black and white pieces on the board. The goal is to have more pieces of your color on the board than your opponent by the end of the game. A piece is placed on an empty square and must sandwich one or more of the opponent's pieces between the newly placed piece and another of the player's pieces in a horizontal, vertical, or diagonal line. The opponent's pieces in between are then flipped to the player's color. Given a screenshot of the Reversi board, please represent the current state of the game using an 8x8 matrix. In this matrix, empty cells should be represented by 0, black pieces by 1, and white pieces by 2. Please strictly follow the format:

Game State: `<boardmatrix>`

where `<boardmatrix>` is an 8x8 matrix. For example:

$$\text{Game State: } \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

represents a Reversi board with a few pieces already placed and empty cells (represented by 0).

Perceiving Task Prompt for Sudoku:

Sudoku is a logic-based puzzle played on a 9x9 grid, where the grid is subdivided into nine 3x3 subgrids. The goal is to fill the grid so that each row, each column, and each 3x3 subgrid contains all digits from 1 to 9 without repetition. Given a screenshot of the Sudoku grid, please represent the current state of the puzzle using a 9x9 matrix. In this matrix, an empty cell should be represented by 0, and filled cells should contain their respective numbers (1-9). Please strictly follow the format:

Game State: `<boardmatrix>`

where `<boardmatrix>` is a 9x9 matrix. For example:

$$\text{Game State: } \begin{bmatrix} 5 & 3 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 1 & 9 & 5 & 0 & 0 & 0 \\ 0 & 9 & 8 & 0 & 0 & 0 & 0 & 6 & 0 \\ 8 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 3 \\ 4 & 0 & 0 & 8 & 0 & 3 & 0 & 0 & 1 \\ 7 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 6 \\ 0 & 6 & 0 & 0 & 0 & 0 & 2 & 8 & 0 \\ 0 & 0 & 0 & 4 & 1 & 9 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 8 & 0 & 0 & 7 & 9 \end{bmatrix}$$

represents a partially filled Sudoku grid with some cells empty (represented by 0).

Perceiving Task Prompt for Tic Tac Toe:

> Tic Tac Toe is a game played on a 3x3 grid where players take turns placing X or O in the cells. Given a screenshot of the game board, please determine the current game state using a 3x3 matrix. In this matrix, an empty cell should be represented by -1, X should be represented by 1, and O should be represented by 0. Please strictly follow the format:
>
> **Game State:** `<boardmatrix>`
>
> where `<boardmatrix>` is a 3x3 matrix. For example:
>
> $$\text{Game State: } \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
>
> represents an empty board.

As shown in the prompts above, each task specifies a clear and structured output format that the models are expected to follow. For each game, the models must generate a matrix representing the current state of the game board. These matrices must conform to the exact specifications outlined in the task prompts, ensuring that the output is both precise and standardized.

To evaluate the model's performance on the *Perceiving* task, we compute the accuracy by comparing the model's predicted matrix with the ground truth matrix. Each task specifies a clear expected output format, typically a matrix representing the current game state. Let P represent the predicted matrix and G represent the ground truth matrix, both of size $m \times n$, where m and n are the dimensions of the specific game board. The accuracy can be calculated using the following formula:

$$Acc_p = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{I}(P_{ij} = G_{ij})$$

To ensure the responses are in the correct format, we apply a regex to extract the matrix from the model's output. If the model's response cannot be parsed by the regex or does not adhere to the expected structure, it is marked as invalid, and a zero accuracy score is assigned for that task. This approach ensures a fair and consistent evaluation across all models.

B.2 Question Answering

The *Question Answering* task follows the structure of traditional VQA but is adapted to a game-based environment. A key advantage of this setup is that specific game states can be easily obtained from the simulator, allowing for the automatic generation of question-answer pairs without manual annotation. This contrasts with traditional VQA, which requires significant human effort for data collection and labeling. Additionally, this approach enables detailed manipulation of the input, allowing us to evaluate models on **fine-grained perception** and reasoning based on visual details. Unlike standard VQA, which often focuses on broader scene-level questions, LVLM-Playground encourages models to engage with and reason about visual details.

We generate screenshots of randomly created game states, similar to the process used in the perceiving task (as shown in Figure 6). For each game, a specific question pool is designed, accompanied by APIs to retrieve the corresponding ground truth answers. These questions are crafted to assess the model's understanding of the game state and its reasoning based on visual input.

To address the common challenge in VQA evaluation of **variability in response format** from language models, we adopt a **multiple-choice** format for the Q&A task. Each question is paired with a predefined set of candidate answers, including one correct answer and several distractors. This setting eliminates the reliance on free-form responses and simplifies the validation process, focusing the evaluation on the model's reasoning and perception abilities rather than instruction-following.

For each game, questions are designed with corresponding candidate answers, typically containing four options, although fewer options may be provided for binary questions (e.g., Yes/No). The

multiple-choice format ensures consistency across tasks and enables direct comparison of model performance. The prompt structure includes clear instructions and examples of valid question-answer pairs to guide the model in selecting the correct answer.

Q&A Task Prompt for Chess:

> Chess is played on an 8x8 board with six types of pieces, including pawns, knights, bishops, rooks, queens, and kings, for both white and black players. The board uses a coordinate system where columns are labeled "a" through "h" from left to right, and rows are labeled "1" through "8" from bottom to top. For example, the bottom-left corner is "a1" and the top-right corner is "h8".
>
> Please answer the multiple-choice question based on the provided screenshot of the current game state. There is only one correct answer. Please choose the correct answer from [A, B, C, D].
>
> ONLY RESPOND WITH THE LETTER. DO NOT INCLUDE ANY OTHER INFORMATION
> `{question}`
> **Your Option:**

Q&A Task Prompt for Gomoku:

> Gomoku is played on a 15x15 grid, where black and white stones are placed in turns. The goal is to place five consecutive stones in a horizontal, vertical, or diagonal line.
>
> Please answer the multiple-choice question based on the provided screenshot of the current game state. There is only one correct answer. Please choose the correct answer from [A, B, C, D].
>
> ONLY RESPOND WITH THE LETTER. DO NOT INCLUDE ANY OTHER INFORMATION
> `{question}`
> **Your Option:**

Q&A Task Prompt for Minesweeper:

> Minesweeper is played on an 8x8 grid, where each cell can either contain a mine, be unrevealed, or show the number of adjacent mines (from 0 to 8).
>
> Please answer the multiple-choice question based on the provided screenshot of the current game state. There is only one correct answer. Please choose the correct answer from [A, B, C, D].
>
> ONLY RESPOND WITH THE LETTER. DO NOT INCLUDE ANY OTHER INFORMATION
> `{question}`
> **Your Option:**

Q&A Task Prompt for Reversi:

> Reversi (also known as Othello) is played on an 8x8 grid where two players take turns placing black and white pieces on the board.
>
> Please answer the multiple-choice question based on the provided screenshot of the current game state. There is only one correct answer. Please choose the correct answer from [A, B, C, D].
>
> ONLY RESPOND WITH THE LETTER. DO NOT INCLUDE ANY OTHER INFORMATION
> `{question}`
> **Your Option:**

Q&A Task Prompt for Sudoku:

> Sudoku is played on a 9x9 grid, where each row, column, and 3x3 subgrid must contain the numbers 1 to 9 exactly once.
>
> Please answer the multiple-choice question based on the provided screenshot of the current game state. There is only one correct answer. Please choose the correct answer from [A, B, C, D].
>
> ONLY RESPOND WITH THE LETTER. DO NOT INCLUDE ANY OTHER INFORMATION
> `{question}`
> **Your Option:**

Q&A Task Prompt for Tic Tac Toe:

> Tic Tac Toe is a game played on a 3x3 grid, where players take turns placing X or O in the cells.
>
> Please answer the multiple-choice question based on the provided screenshot of the current game state. There is only one correct answer. Please choose the correct answer from [A, B, C, D].
>
> ONLY RESPOND WITH THE LETTER. DO NOT INCLUDE ANY OTHER INFORMATION
> `{question}`
> **Your Option:**

As shown above, in each Q&A task prompt, the model is provided with **three question-answer examples** to clarify the expected output format. This approach leverages the model's **in-context learning and instruction-following capabilities**, while also standardizing the evaluation process. Only responses that strictly adhere to the specified format and contain the correct answer will contribute to the accuracy. Otherwise, even if the answer is correct but presented in an unexpected format, it will be marked as incorrect. This is reasonable, much like in human exams where failure to follow formatting rules can result in penalties, regardless of the correctness of the answer. Therefore, the ability to strictly follow instructions is crucial for the model, as it reflects whether it truly understands the given instructions.

For each game, we have designed a unique set of questions tailored to its specific dynamics and rules. Below, we outline the types of questions used in each game and how they test different aspects of the model's perception and reasoning capabilities.

Chess Question Types.

- **Piece Color at Position**
  Question: What is the color of the piece at column {col_label}, row {row_label}?

- **Piece Name at Position**
  Question: What is the piece at column {col_label}, row {row_label}?

- **Count of Specific Pieces**
  Question: How many {piece_color} {piece_name}s are on the board?

- **Count of Pieces in Row/Column**
  Question: How many pieces are in row {row_label}?
  OR
  Question: How many pieces are in column {col_label}?

- **Comparison of White and Black Pieces**
  Question: Which color has more pieces, white or black?

- **Comparison of Two Piece Types**
  Question: Which has more, {piece1_name}s or {piece2_name}s?

- **Count of Edge Pieces**
  Question: How many pieces are on the edge of the board?

- **More Empty Cells in Top or Bottom Half of the Board**
  Question: Which half of the board has more empty cells, top or bottom?

Gomoku Question Types.

- **Stone at Specific Position**
  Question: What is the stone at row {row_label}, column {col_idx}?

- **Count of Specific Stones**
  Question: How many 'Black' stones are on the board?
  OR
  Question: How many 'White' stones are on the board?

- **Count of Stones in Row/Column**
  Question: How many 'Black' stones are in row {row_label}?
  OR
  Question: How many 'White' stones are in column {col_idx}?

- **Winning Condition**
  Question: Is there a winning line on the board?

- **Adjacent Stones**
  Question: How many adjacent stones are around row {row_label}, column {col_idx}?

- **Count of Empty Cells**
  Question: How many empty cells are there on the board?

- **Maximum Consecutive Stones**
  Question: What is the maximum number of consecutive 'Black' stones in row {row_label}?
  OR

Question: What is the maximum number of consecutive 'White' stones on any diagonal?

- **Count of Edge Stones**
  Question: How many 'Black' stones are on the edge of the board?
  OR
  Question: How many 'White' stones are on the edge of the board?

Minesweeper Question Types.

- **Revealed Symbol at Specific Position**
  Question: What is the revealed number or symbol in row {row_label}, column {col_label}?

- **Count of Revealed Mines**
  Question: How many revealed mines are there on the board?

- **Count of Revealed Cells**
  Question: How many revealed cells are there on the board?

- **Count of Revealed Cells in Row/Column**
  Question: How many revealed cells are there in row {row_label}?
  OR
  Question: How many revealed cells are there in column {col_label}?

- **Number of Adjacent Mines**
  Question: How many mines are adjacent to the cell at row {row_label}, column {col_label}?

- **Is There a Mine at a Specific Position**
  Question: Is there a revealed mine at row {row_label}, column {col_label}?

Reversi Question Types.

- **Symbol at Specific Position**
  Question: What is the symbol in row {row_label}, column {col_label}?

- **Count of Specific Symbol**
  Question: How many 'Black' pieces are present on the board?
  OR
  Question: How many 'White' pieces are present on the board?

- **Count of Empty Cells**
  Question: How many empty cells are there on the board?

- **Count of Specific Symbol in Row/Column**
  Question: How many 'Black' pieces are there in row {row_label}?
  OR
  Question: How many 'White' pieces are there in column {col_label}?

- **Row with Most Pieces of Specific Symbol**
  Question: Which row contains the most 'Black' pieces?
  OR
  Question: Which row contains the most 'White' pieces?

- **Comparison of Black and White Pieces**
  Question: Which player has more pieces on the board, 'Black' or 'White'?

- **Total Number of Pieces in Row/Column**
  Question: How many pieces are there in total in row {row_label}?
  OR
  Question: How many pieces are there in total in column {col_label}?

- **Total Count of Black or White Pieces on the Board**
  Question: How many 'Black' pieces are there in total on the board?
  OR
  Question: How many 'White' pieces are there in total on the board?

Sudoku Question Types.

- **Symbol at Specific Position**
  Question: What is the number in row {row}, column {col}?

- **Count of Specific Number**
  Question: How many '{number}'s are present on the board?

- **Count of Empty Cells**
  Question: How many empty cells are there on the board?

- **Count of Specific Number in Row/Column**
  Question: How many '{number}'s are there in row {row}?
  OR
  Question: How many '{number}'s are there in column {col}?

- **Count of Specific Number in Subgrid**
  Question: How many '{number}'s are there in the subgrid starting at row {subgrid_start_row}, column {subgrid_start_col}?

- **Sum of Numbers in Row/Column**
  Question: What is the sum of numbers in row {row}?
  OR
  Question: What is the sum of numbers in column {col}?

- **Sum of Numbers in Subgrid**
  Question: What is the sum of numbers in the subgrid starting at row {subgrid_start_row}, column {subgrid_start_col}?

- **Does Row/Column Contain a Specific Number**
  Question: Does row {row} contain the number '{number}'?
  OR
  Question: Does column {col} contain the number '{number}'?

- **Count of Empty Cells in Subgrid**
  Question: How many empty cells are there in the subgrid starting at row {subgrid_start_row}, column {subgrid_start_col}?

Tic Tac Toe Question Types.

- **Symbol at Specific Position**
  Question: What is the symbol in row {row}, column {col}?

- **Count of Specific Symbol**
  Question: How many '{symbol}'s are present on the board?

- **Count of Empty Cells**
  Question: How many empty cells are there?

- **Count of Specific Symbol in Row/Column**
  Question: How many '{symbol}'s are there in row {row}?
  OR
  Question: How many '{symbol}'s are there in column {col}?

- **Winner of the Game**
  Question: Did X or O win the game?

- **Count of Red/Blue Marks on the Board**
  Question: How many {color} marks are present on the board?
  OR
  Question: How many {color} marks are there in row {row}?
  OR
  Question: How many {color} marks are there in column {col}?

As shown in the examples above, the question-answering task challenges the model's ability to engage with various question types that require reasoning and fine-grained perception. Unlike the pure visual *perceiving* task, these questions go further by testing the model's reasoning skills and multiple detailed analytical abilities. For example, some questions involve **counting**, such as determining how many pieces of a certain type are present on a board, while others require basic **arithmetic calculations**, such as summing numbers in rows or columns in Sudoku. There are also tasks that require **OCR-like recognition**, identifying symbols or numbers at specific positions, as well as questions that demand **geometrical shape recognition**, such as identifying edge pieces or distinguishing between shapes like circles and rectangles. These capabilities, which usually require separate benchmarks to evaluate, are integrated within the proposed LVLM-Playground, offering a comprehensive assessment of the model's ability to perceive, reason, and handle fine-grained visual details.

B.3 Rule Following

The Rule Following task evaluates the model's ability to internalize and execute game rules accurately, assessing whether the model can determine and perform valid moves in accordance with the specific rules of each game. Unlike the Perceiving and Q&A tasks, where the randomly generated game states may not strictly follow the game rules, the Rule Following task ensures that all randomly generated game states adhere to the rules of the game. Then, the model is tasked with selecting a valid move based on the state.

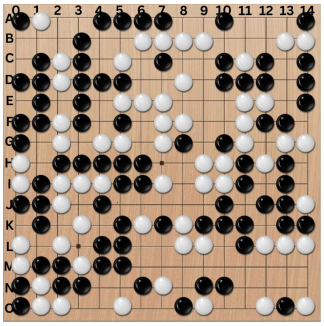
In chess (Figure 8a), each piece has specific movement rules, such as pawns moving one square forward. These moves can be described using algebraic notation, where each square is identified by a letter and a number. For instance, a valid pawn move might be from *e2* to *e4*, indicating that the pawn moves two squares forward from its starting position.

In Gomoku (Figure 8b) and Minesweeper (Figure 8c), valid moves are selecting empty intersections on the board or unmarked (gray) cells on the Minesweeper map.

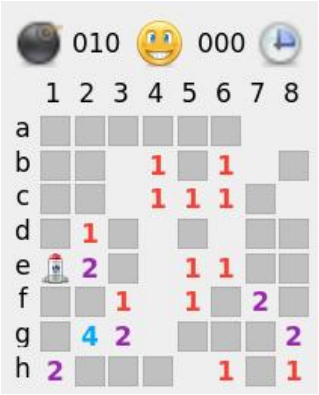
In Reversi (Figure 8d), a valid move must sandwich at least one of the opponent's discs between two of the player's discs.

In Sudoku (Figure 8e), a number must not be repeated within the same row, column, or subgrid.

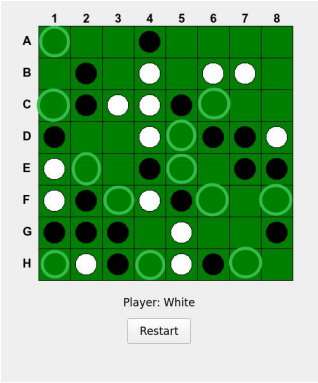
(a) Chess



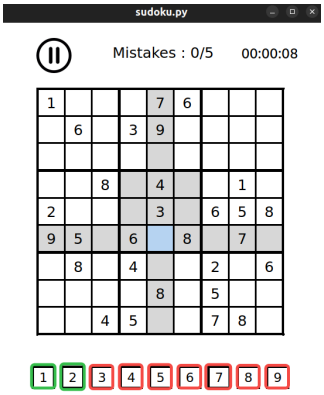
(b) Gomoku



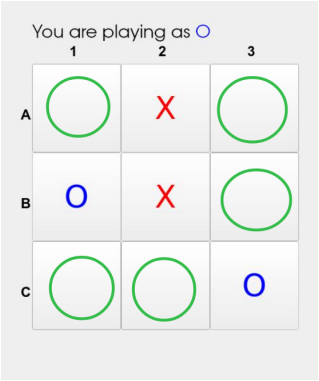
(c) Minesweeper



(d) Reversi



(e) Sudoku



(f) Tictactoe

Figure 7: Visualization of the Rule Following task, where the model is presented with a random game state and required to select a valid move. In (a) Chess, (d) Reversi, (e) Sudoku, and (f) TicTacToe, valid moves are highlighted in green rectangles or circles. In (b) Gomoku, valid moves are represented by the empty intersections, and in (c) Minesweeper, valid moves correspond to the uncovered (gray) cells.

In Tic-Tac-Toe (Figure 7f), a move is valid if it occupies an empty cell, as each cell can only be filled once.

For each game, we provide detailed game rules through structured prompts to test the model's ability to internalize and apply them. Specifically, the rule-following prompts for each game are as follows:

Rule Following Task Prompt for Chess:

Chess is a strategy board game played on an 8x8 grid. The objective of the game is to checkmate the opponent's king. The board consists of different types of pieces: pawns, knights, bishops, rooks, queens, and kings for both white and black players. Each piece moves in specific ways, and players take turns moving their pieces.

The chessboard is labeled using algebraic notation, where columns (files) are labeled "a" to "h" from left to right, and rows (ranks) are labeled "1" to "8" from bottom to top. For example, "e2" represents the square on column e and row 2. A valid move is indicated using this notation by specifying the starting and ending square, for example, "e2e4" represents moving a piece from e2 to e4.

Based on the current board state, your task is to find and provide a valid move for the next player using algebraic notation. Please strictly follow the format:

Movement: `<move>`

where `<move>` represents the chess move in standard algebraic notation. For example, "e2e4" moves a piece from e2 to e4.

Rule Following Task Prompt for Gomoku:

Gomoku is played on a 15x15 grid, where black and white stones are placed on the intersections of the grid. The objective is to place five consecutive stones in a horizontal, vertical, or diagonal line. The game starts with an empty board, and the grid is labeled with rows A to O and columns 1 to 15. Each intersection can only be occupied by one stone, either black or white. Based on the board state, your task is to find an empty intersection where you can place your next stone. Please follow the format below when indicating your move:

Movement: `<position>`

The `<position>` should be a valid combination of rows A to O and columns 1 to 15, such as A1, H8, or O15.

Rule Following Task Prompt for Minesweeper:

Minesweeper is played on an 8x8 grid. Each cell can either contain a mine, be unrevealed, or show the number of adjacent mines (from 0 to 8). The goal is to reveal all cells that do not contain mines, without triggering any mines. The grid is labeled with rows A to H and columns 1 to 8. Each cell can only be revealed once. Based on the board state, your task is to find a cell to reveal next. Please follow the format below when indicating your move:

Movement: `<position>`

The `<position>` should be a valid combination of rows A to H and columns 1 to 8, such as A1, B5, or H8.

Rule Following Task Prompt for Reversi (Othello):

> Reversi (also known as Othello) is played on an 8x8 grid. Players take turns placing black and white pieces on the board. The grid starts with two black pieces and two white pieces in the center. A valid move consists of placing a piece in such a way that it sandwiches one or more of the opponent's pieces between the newly placed piece and another of the player's pieces in a horizontal, vertical, or diagonal line. After placing the piece, all of the opponent's pieces in between are flipped to the player's color. Based on the current game state, your task is to find a valid position where you can place your next piece and flip at least one of the opponent's pieces. Please follow the format below when indicating your move:
>
> **Movement:** `<position>`
>
> The `<position>` should be a valid combination of rows A to H and columns 1 to 8, such as A1, B2, or C3.

Rule Following Task Prompt for Sudoku:

> Sudoku is played on a 9x9 grid, where each row, column, and 3x3 subgrid must contain the numbers 1 to 9 exactly once. The grid starts with the top-left corner as A1, where rows are labeled from A to I and columns are numbered from 1 to 9. A valid move involves placing a digit from 1 to 9 in an empty cell, ensuring that the number does not already appear in the same row, column, or 3x3 subgrid. Based on the current state of the Sudoku grid, your task is to find a valid empty cell where you can place a digit and make a valid move. Please follow the format below when indicating your move:
>
> **Movement:** `<row><column> <digit>`
>
> Here, `<row>` is from A to I (representing rows), `<column>` is from 1 to 9 (representing columns), and `<digit>` is a number between 1 and 9. For example, `A1 5` means placing the digit 5 in the top-left corner of the grid.

Rule Following Task Prompt for Tic Tac Toe:

> Tic Tac Toe is a game played on a 3x3 grid where two players take turns placing their respective marks, X or O, in the cells. The game begins with an empty board, and the grid is labeled with rows A to C and columns 1 to 3. Each position on the grid can only be occupied by one mark at a time. Based on the current state of the board, your task is to find an empty cell where you can place your next stone. Please follow the format below when indicating your move:
>
> **Movement:** `<position>`
>
> The position should be a combination of a row (A to C) and a column (1 to 3), such as A1, B2, or C3.

B.4 END-TO-END PLAYING

E2E Playing task evaluates the model's capability to autonomously play a game from start to finish, testing its ability to integrate perception, reasoning, and decision-making. As discussed in the main text, the model suggests the next move in alphanumeric format, which is then executed within a game simulator. For games involving an opponent, the model must respond to the adversarial actions and adapt its strategy accordingly.

Here, we first provide the E2E Play task prompt for each game:

End-to-End Playing Task Prompt for Tic Tac Toe:

> Tic Tac Toe is played on a 3x3 grid. Players take turns placing X or O in the cells. The goal is to be the first to form an unbroken line of three marks horizontally, vertically, or diagonally. The game starts with an empty board, and the O player goes first. The grid is labeled with rows A to C and columns 1 to 3. You are playing as O, aiming to win by placing marks strategically. Each position can only be occupied by one mark, so do not choose a spot that is already taken. Based on the board state screenshots, please first observe the current situation, then carefully think and explain your strategy briefly, and finally output your movement for this status. Please strictly follow the following format:
>
> **Observation:** `<observation>`
> **Strategy:** `<strategy>`
> **Movement:** `<position>`
>
> where the observation should briefly summarize the current situation, the strategy is a brief explanation of how you plan to win the game, and the position can be any combination of rows A to C and columns 1 to 3, for example, A1, 2B, or C3.

End-to-End Playing Task Prompt for Sudoku:

> Sudoku is played on a 9x9 grid, divided into nine 3x3 subgrids. The objective is to fill the grid with digits from 1 to 9, ensuring that each row, column, and 3x3 subgrid contains each digit exactly once. The grid starts with some pre-filled numbers, and the rest of the cells are empty. You will receive a screenshot of the current game state, where some cells will contain digits, while others will be empty. Based on the current state, please first observe the grid carefully, then explain your strategy for placing the next number, and finally output your movement for this status. Please strictly follow the following format:
>
> **Observation:** `<observation>`
> **Strategy:** `<strategy>`
> **Movement:** `<row> <column> <digit>`
>
> where the observation should briefly summarize the current situation, the strategy is a brief explanation of how you plan to fill the grid, and the movement indicates the cell's row (A to I) and column (1 to 9) along with the digit (1 to 9) you wish to place. For example, `B2 5` means placing the digit 5 in the cell located in row B and column 2.

End-to-End Playing Task Prompt for Reversi (Othello):

Reversi, also known as Othello, is played on an 8x8 grid. Players take turns placing black and white pieces on the board. The goal is to have the majority of your pieces on the board at the end of the game. A valid move consists of placing a piece in such a way that it sandwiches one or more of the opponent's pieces between the newly placed piece and another piece of the player's color in a horizontal, vertical, or diagonal line. The grid starts with two black pieces and two white pieces in the center. Based on the current game state, please first observe the board, then explain your strategy for your next move, and finally output your movement for this status. Please strictly follow the following format:

Observation: `<observation>`
Strategy: `<strategy>`
Movement: `<position>`

where the observation should briefly summarize the current situation, the strategy is a brief explanation of how you plan to win the game, and the position should be a valid combination of rows A to H and columns 1 to 8, such as A1, B2, or C3.

End-to-End Playing Task Prompt for Minesweeper:

Minesweeper is played on an 8x8 grid with 10 hidden mines. The objective is to uncover all the safe squares without detonating any mines. Each uncovered square may display a number indicating how many mines are adjacent to that square, or it may be empty, meaning no adjacent mines. The game starts with a fully covered grid, and you will receive a screenshot showing some uncovered squares. Based on the current game state, please first observe the grid carefully, then explain your strategy for uncovering a safe square, and finally output your movement for this status. Please strictly follow the following format:

Observation: `<observation>`
Strategy: `<strategy>`
Movement: `<row> <column>`

where the observation should briefly summarize the current situation, the strategy is a brief explanation of how you plan to avoid hitting a mine and uncover safe squares, and the movement indicates the cell's row (A to H) and column (1 to 8) that you wish to uncover. For example, `C3` means uncovering the square located in row C and column 3.

End-to-End Playing Task Prompt for Gomoku:

> Gomoku is played on a 15x15 grid, where two players take turns placing their respective pieces (black or white) on the board. The objective is to be the first to connect five of your pieces in a row, either horizontally, vertically, or diagonally. The game starts with an empty board, and you will be playing as black. Based on the current game state, please first observe the board, then explain your strategy for your next move, and finally output your movement for this status. Please strictly follow the following format:
>
> **Observation:** `<observation>`
> **Strategy:** `<strategy>`
> **Movement:** `<position>`
>
> where the observation should briefly summarize the current situation, the strategy is a brief explanation of how you plan to achieve five in a row, and the position should be a valid combination of rows A to O and columns 1 to 15, such as A1, B2, or C3.

End-to-End Playing Task Prompt for Chess:

> Chess is played on an 8x8 checkered board, where two players take turns moving their pieces: king, queen, rooks, knights, bishops, and pawns. The objective is to checkmate the opponent's king. The game starts with the pieces arranged in their respective positions, and you will be playing as white. Based on the current game state, please first observe the board, then explain your strategy for your next move, and finally output your movement for this status. Please strictly follow the following format:
>
> **Observation:** `<observation>`
> **Strategy:** `<strategy>`
> **Movement:** `<algebraic notation>`
>
> where the observation should briefly summarize the current situation, the strategy is a brief explanation of how you plan to gain an advantage, and the movement should be expressed using algebraic notation. For example, `e4` indicates moving a pawn to e4, while `Nf3` means moving a knight to f3.

C SEARCH-BASED AI OPPONENTS

In adversarial games, effective decision-making is essential as players must anticipate and counter their opponent's moves. To facilitate this, many search-based methods have established a solid foundation for developing AI opponents. Below are key approaches utilized in our framework:

- **Tic-Tac-Toe:** The AI opponent employs the **Minimax** algorithm, which recursively evaluates all possible moves to determine the optimal play. The algorithm considers both the AI's and the opponent's potential moves, aiming to maximize the AI's chances of winning while minimizing the opponent's chances. The evaluation function assesses the board state, returning positive scores for winning configurations and negative scores for losing ones. This allows the AI to select the best possible move at each turn, effectively making it a formidable opponent.
- **Sudoku:** N/A
- **Reversi:** The AI opponent in Reversi uses an Alpha-Beta pruning technique combined with a scoring system to evaluate board states. The AI identifies valid moves and simulates their outcomes by temporarily applying the moves to a copy of the board. The Alpha-Beta algorithm efficiently narrows down the search space, allowing the AI to maximize its score

while minimizing the opponent's score. This approach enables the AI to make strategic decisions and adapt to the opponent's moves effectively.

- **Minesweeper:** N/A

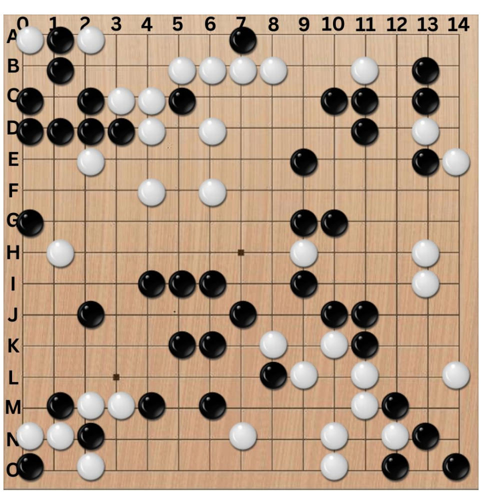
- **Gomoku:** The Gomoku AI utilizes a recursive evaluation strategy to determine the best move. It evaluates the board based on potential winning patterns, assessing both its own and the opponent's configurations. The AI implements a depth-limited search to simulate possible moves, using an evaluation function that assigns values based on the length of contiguous pieces. This allows the AI to prioritize moves that advance its strategy while blocking the opponent's attempts to connect five in a row.

- **Chess:** The Chess AI leverages the open-source Sunfish library, which is a lightweight chess engine that implements the Minimax algorithm with Alpha-Beta pruning. This library enables the AI to evaluate board positions efficiently, considering multiple possible moves and their outcomes. The Sunfish engine uses a heuristic evaluation function to assess board states, allowing the AI to make strategic decisions and respond to the opponent's moves effectively.

D Qualitative Results

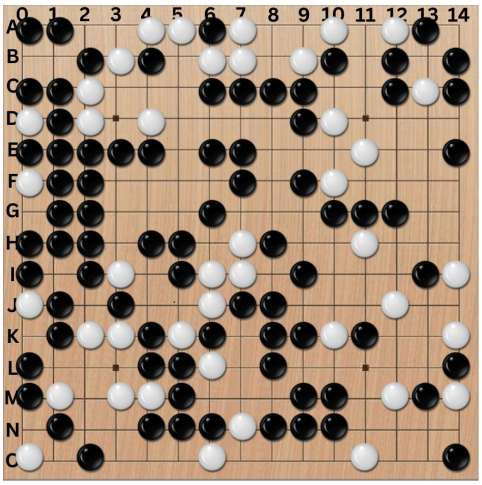
To provide a clear view of the LVLMs' behavior on LVLM-Playground, in this section, we present qualitative results across different models and tasks, following the findings mentioned in Section 4.1.

D.1 Looping Behavior in Long Matrix Outputs.



[Perceiving Task]

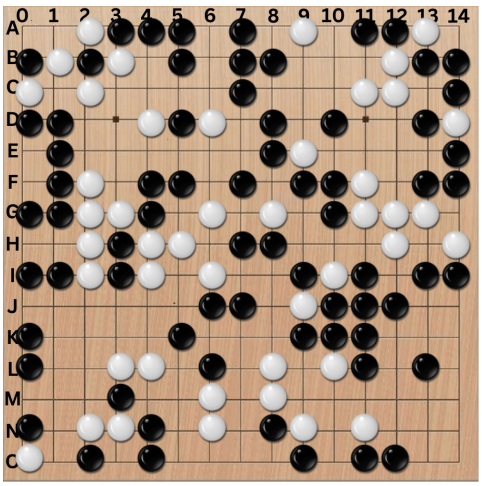
Phi3.5-VL: Description: [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0,



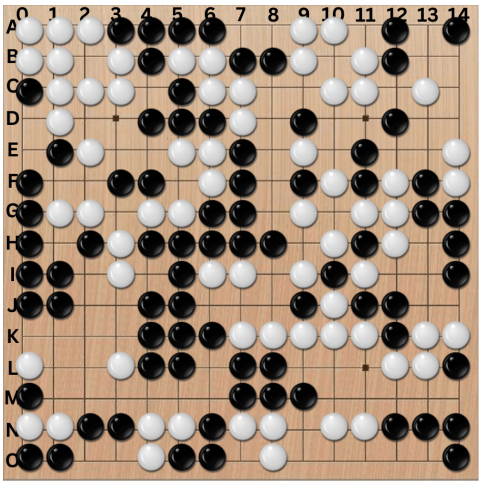
[Perceiving Task]
QW2-VL: Description: Here is the 15x15 matrix representation of the Gomoku board in the image: ‘‘‘ [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]] ‘‘‘ In this matrix, 1 represents a black stone, 2 represents a white stone, and 0 represents an empty space.

In the above cases, both models tend to repeat the example game state from the input prompt, which is completely irrelevant to the current game state. This reflects a potential limitation in their ability to adapt to new inputs. Additionally, their tendency to enter infinite loops disrupts the expected output format, i.e., the 15×15 matrix. For example, Phi3.5-VL attempts to output a 23×15 matrix, reaching the maximum token limit, while QW2-VL produces a 16×15 matrix, which also deviates from the expected format.

D.2 Poor Performance on Detail and Dense Object Perception



[Perceiving Task]
Gemini1.5-pro: Description: ‘‘‘python [[0, 2, 1, 1, 2, 1, 1, 2, 1, 0, 1, 2, 2, 1, 0], [2, 1, 2, 1, 2, 0, 2, 1, 2, 0, 0, 1, 2, 0, 1], [1, 2, 1, 2, 1, 1, 1, 2, 0, 0, 1, 2, 1, 0, 2], [1, 1, 0, 0, 2, 1, 2, 1, 2, 0, 2, 0, 2, 1, 1], [0, 1, 0, 2, 1, 2, 1, 0, 1, 1, 0, 0, 1, 2, 0], [1, 0, 1, 1, 2, 1, 1, 2, 0, 2, 1, 0, 2, 0, 1], [0, 1, 2, 1, 1, 2, 0, 1, 2, 1, 2, 1, 0, 2, 0], [1, 0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 2, 1, 1, 0], [0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 2, 1, 2, 0, 2], [0, 0, 1, 2, 1, 0, 1, 2, 1, 1, 1, 1, 1, 0, 1], [0, 0, 2, 1, 0, 2, 0, 1, 2, 1, 1, 1, 2, 1, 0], [0, 1, 2, 2, 1, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0], [0, 1, 1, 2, 2, 1, 0, 1, 0, 2, 1, 2, 0, 1, 1], [1, 0, 1, 2, 1, 2, 1, 0, 2, 1, 2, 0, 1, 2, 0], [0, 1, 2, 1, 1, 2, 0, 1, 2, 0, 1, 2, 1, 0, 1]] ‘‘‘
accuracy: 37.78%

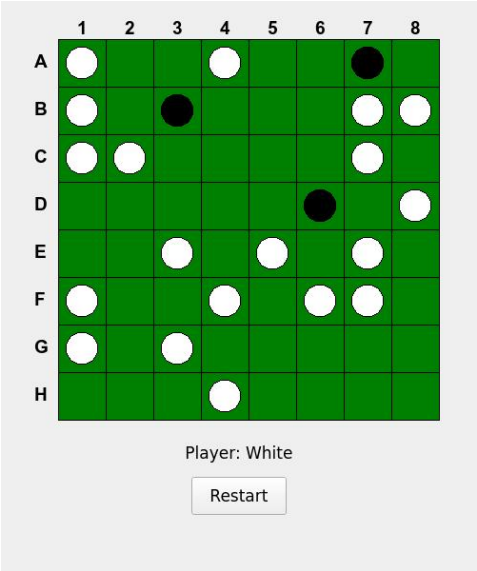


[Perceiving Task]
Claude3.5-sonnet: Game State: [[1, 1, 1, 2, 2, 2, 2, 0, 0, 1, 1, 2, 0, 2, 0], [1, 1, 0, 2, 2, 1, 1, 2, 2, 1, 0, 0, 2, 0, 0], [2, 1, 1, 1, 0, 2, 1, 1, 0, 0, 1, 1, 0, 1, 0], [0, 1, 0, 2, 2, 2, 1, 0, 2, 0, 0, 2, 0, 0, 0], [2, 1, 0, 0, 1, 1, 2, 0, 1, 0, 2, 0, 0, 1, 0], [2, 0, 0, 2, 2, 1, 2, 0, 2, 1, 2, 1, 2, 1, 0], [2, 1, 1, 0, 1, 1, 2, 2, 0, 1, 0, 1, 1, 2, 2], [2, 0, 2, 1, 2, 2, 2, 2, 2, 0, 2, 1, 1, 2, 0], [2, 2, 0, 1, 0, 2, 1, 1, 0, 2, 1, 0, 0, 2, 0], [2, 2, 0, 2, 2, 0, 0, 0, 2, 1, 2, 2, 0, 0, 0], [0, 0, 0, 2, 2, 2, 1, 1, 1, 1, 2, 0, 1, 1, 0], [1, 0, 1, 0, 2, 2, 0, 2, 2, 0, 0, 1, 1, 0, 2], [2, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 0], [1, 1, 2, 2, 1, 1, 2, 1, 1, 0, 1, 1, 2, 2, 2], [2, 2, 0, 0, 1, 2, 2, 0, 1, 0, 0, 0, 0, 0, 2]]
accuracy: 35.11%

From the above cases, it is evident that even the most advanced commercial LVLMs struggle to accurately recognize densely packed pieces on the board, despite this being a straightforward task for humans. Both Gemini and Claude achieved relatively low accuracies of 37.78% and 35.11%, respectively, suggesting the limitations of current models in effectively perceiving and interpreting detailed visual information.

D.3 RLHF May Harm Instruction-following Ability

[Perceiving Task]
GPT-4o: I'm unable to analyze images for Minesweeper boards directly. However, you can manually describe the revealed parts of the board, and I'll help create the corresponding matrix. If you describe the rows with their numbers and meanings, I can format them accordingly.



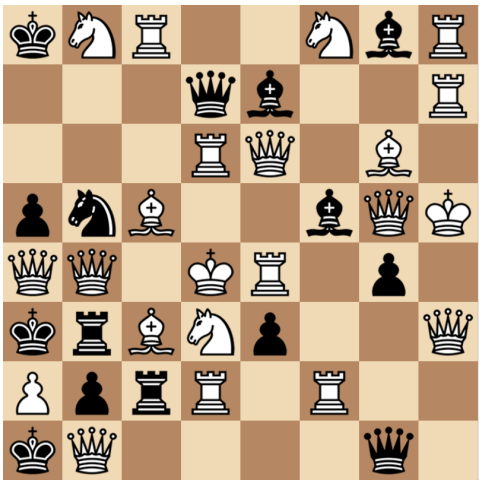
[Q&A Task]
Prompt: Reversi (also known as Othello) is played on an 8x8 grid where two players take turns placing black and white pieces on the board. Please answer the multiple-choice question based on the provided screenshot of the current game state. There is only one correct answer. Please choose the correct answer from [A, B, C, D].

ONLY RESPOND WITH THE LETTER. DO NOT INCLUDE ANY OTHER INFORMATION
`{question}`
Your Option:

Claude-3.5-sonnet : I'm sorry I can't assist with that.

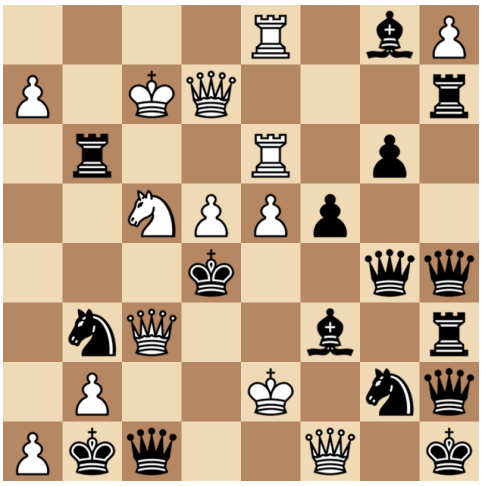
In the above cases, both models failed to answer the questions effectively. In the Minesweeper perceiving task, GPT-4o inexplicably refused to parse the Minesweeper map, stating that it does not have the ability to analyze images for Minesweeper boards directly. Interestingly, in other instances, it has successfully responded to the same task prompts. We hypothesize that the RLHF may disrupt its instruction-following ability, leading it to incorrectly reject user requests. In the following example, Claude also refused to answer the quesstion.

D.4 Limited Output Diversity.



[Perceiving Task]

InternVl2:
Game State: [[-6, -2, -3, -5, -6, -3, -2, -4], [-1, -1, -1, -1, -1, -1, -1, -1], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 1, 1, 1], [4, 2, 3, 5, 6, 3, 2, 4]]
G

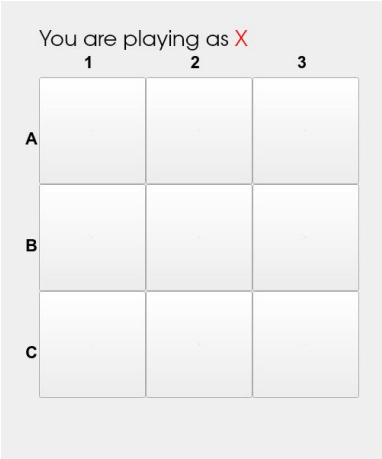


[Perceiving Task]

InternVl2:
Game State: [[-6, -2, -3, -5, -6, -3, -2, -4], [-1, -1, -1, -1, -1, -1, -1, -1], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 1, 1, 1], [4, 2, 3, 5, 6, 3, 2, 4]]
G

From the above cases, with two completely different randomly generated game states, the LVLM transcribed them into the same game state matrix, which is actually the standard initial position of a chess game. This suggests that these models do not adequately process the image input, and their responses are severely limited to a narrow output space, lacking diversity.

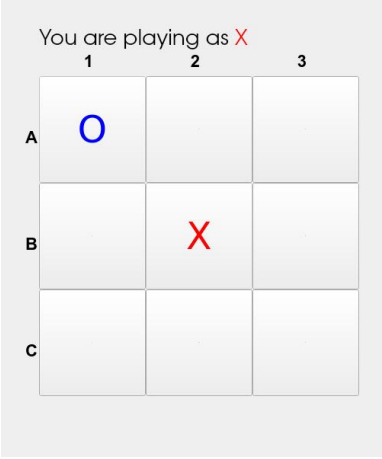
D.5 LVLMs are Stochastic Parrots while Playing Games.



[E2E Playing]
Gemini1.5-Pro:

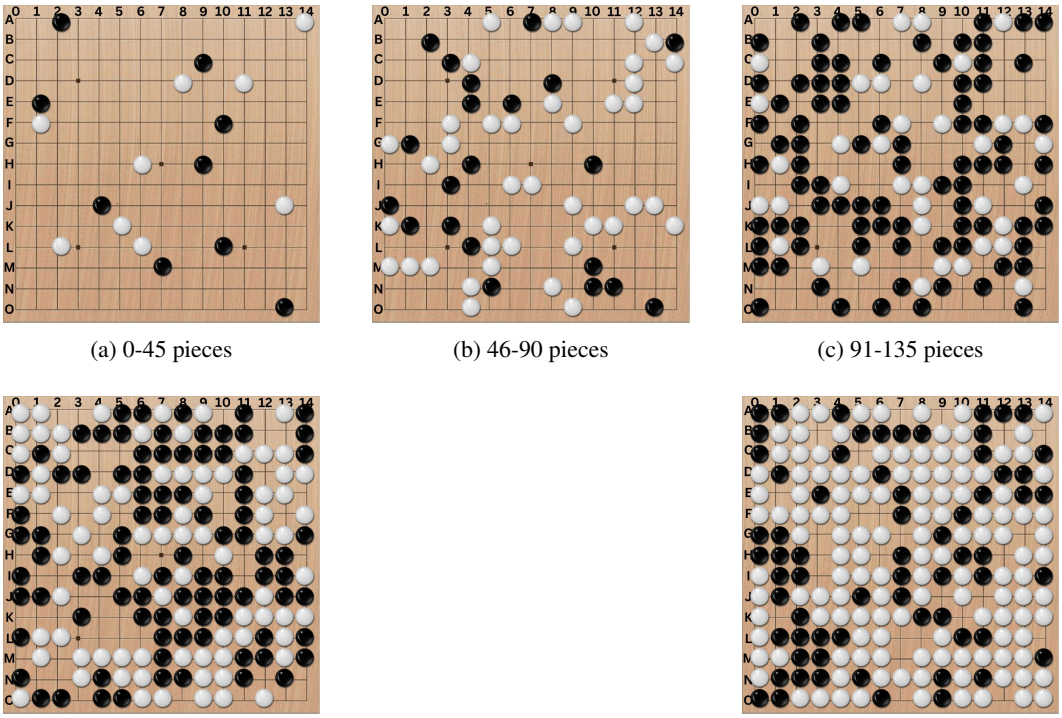
[Round 1]

Observation: The board is empty.
Strategy: As the first player, I want to take the center square to create the most potential winning lines.
Movement: B2



Round [2]

Observation: The opponent (O) has played in the top left corner (A1). I am playing as X.
Strategy: I will place my mark in the center (B2), which is the most strategic position in Tic Tac Toe, as it creates multiple opportunities to win and forces the opponent to defend.
Movement: B2



(a) 0-45 pieces (b) 46-90 pieces (c) 91-135 pieces

(d) 136-180 pieces (e) 181-225 pieces

Figure 8: Gameboard with different density.

In the above case of end-to-end (E2E) playing by Gemini1.5-Pro, the model successfully recognized the initial empty board state and proposed a strategy to occupy the center square, aiming to create the most potential winning lines. In the second round, the model accurately identified the opponent's move in the top left corner (A1). However, it repeatedly chose the same strategy as in round 1, placing its mark in the center (B2), despite it already being occupied by the opponent's last move.

This illustrates an interesting phenomenon: while the model appears to generate reasonable observations and strategies, it ultimately fails to adhere to the rules of the game. This behavior can be likened to a "stochastic parrot," where the model produces outputs based on patterns without fully understanding the context or constraints of the game.

E ADDITIONAL QUANTITATIVE RESULTS

E.1 PERCEPTION PERFORMANCE VS. BOARD COMPLEXITY

Using Gomoku as an example game, we evaluated the model's perception capability under varying board complexities and analyzed its performance as a function of the number of pieces on the board (see Figure 8).

Based on the experiment results (see Table 6), we found that most **LVLMs are highly sensitive to the density of the board** in the perceiving task of Gomoku. For example, Gemini1.5-pro achieved a perception accuracy of 0.864 on sparse boards with 0-45 pieces, but its performance dropped significantly to 0.412 on nearly full boards with 181-225 pieces. In contrast, Claude-3.5 exhibited a relatively smaller performance decline, maintaining a high perception accuracy of 0.774 even on the densest boards.

Interestingly, for many LVLMs, the densest boards are not always the most challenging. For example, performance on boards with a medium density of 91-135 pieces was often lower than on fully occupied boards. This observation suggests that **these models struggle particularly with identifying "empty" spaces** when mixed configurations of occupied and empty positions appear on the

LVLMs	0-45	46-90	91-135	136-180	181-225
GPT-4o	0.801	0.553	0.381	0.320	0.413
Gemini1.5-pro	0.864	0.636	0.464	0.395	0.412
Claude-3.5-sonnet	0.878	0.728	0.686	0.694	0.774
Qwen2-vl-7b	0.861	0.543	0.382	0.264	0.100
Deepseek-vl-7b	0.043	0.011	0	0	0
Phi3-vl	0.210	0.021	0.019	0.004	0.005
LLaVA-1.6-7b	0.073	0	0	0	0
InternVL2-8b	0.891	0.692	0.421	0.114	0.097

Table 6: LVLM perception performance on different board densities.

grid. This highlights a critical limitation in their ability to process partially dense visual patterns effectively.

In addition, we also observed that the **board density significantly impacts the instruction-following ability** of LVLMs. While most open-source methods exhibited near-zero accuracy under high-density settings due to unexpected output formats (such as the infinite loop behavior described in Finding 1 of the main paper), they performed considerably better in sparse configurations. For example, InternVL2-8B achieved an accuracy of 0.692 under a medium-sparse setting with 46-90 pieces but completely failed on the densest boards with 181-225 pieces. This underscores the compounded difficulty that higher board density introduces, not only for perception but also for maintaining proper instruction following behavior.

In addition, we extended the analysis to consider the types of pieces for Chess (**K**ing, **Q**ueen, **R**ook, **B**ishop, **KN**ight, **P**awn) on the board, to check the impact of perception performance:

LVLMs	K	KQ	KQR	KQRB	KQRBN	KQRBNP
GPT-4o	0.481	0.474	0.482	0.431	0.444	0.447
Gemini1.5-pro	0.387	0.369	0.377	0.369	0.358	0.360
Claude-3.5-sonnet	0.588	0.512	0.430	0.376	0.317	0.296
Qwen2-vl-7b	0.298	0.267	0.284	0.279	0.273	0.272
Deepseek-vl-7b	0.014	0	0	0	0	0
Phi3-vl	0.282	0.280	0.269	0.277	0.270	0.273
LLaVA-1.6-7b	0.299	0.291	0.281	0.274	0.279	0.277
InternVL2-8b	0.223	0.214	0.205	0.211	0.203	0.199

Table 7: LVLM perception performance on different board complexity.

Similar to the impact of board density in the Gomoku game, a similar trend can be observed from the above experiment results, which show that **an increasing variety of piece types on the chessboard generally decreases perception accuracy**. For example, Claude-3.5-sonnet demonstrates significant sensitivity to the number of piece types: it achieved 0.588 accuracy when only the King was present on the board, but its performance dropped drastically to 0.296 when all six types of pieces were included.

However, the effect of piece type variety is less pronounced for some other LVLMs. For example, GPT-4o and Gemini1.5-pro exhibit more stable performance across different configurations of piece types. This relative stability might be attributed to their more robust ability to recognize diverse visual patterns. Nonetheless, the overall trend still highlights that as the diversity of piece types increases, most LVLMs struggle to maintain consistent perception accuracy.

E.2 Intermediate Metric for E2E Gameplay

E2E gameplay setting is a critical aspect of LVLM-Playground, as it aims to evaluate the overall capabilities of LVLMs, including perception, reasoning, decision-making, and other skills. However,

given the poor performance of existing models in the E2E gameplay setting, we developed fundamental tasks such as perception, question-answering, and rule-following to evaluate these abilities from multiple aspects. The results of these tasks have provided some insights into the specific strengths and weaknesses of the models.

Even though current models struggle to complete full gameplay, there are still measurable performance differences in their partial success. To address this we introduced an intermediate evaluation metric that measures the **average number of successful steps** taken by a model during gameplay.

In the E2E gameplay setting, each model has a maximum of three attempts to make a valid move per round. If the model fails to follow the rules or makes a valid move after three trials, the game is terminated and marked as a failure. Under this metric, models that can consistently follow rules and make valid moves will achieve longer sequences of successful steps. We conducted 1,000 matches for each model and measured the average number of successful steps as follows:

LVLMs	TicTacToe	Reversi	Sudoku	Minesweeper	Gomoku	Chess
GPT-4o	2.35	1.53	2.17	2.84	5.63	2.37
Gemini1.5-pro	2.14	1.42	2.53	2.77	5.89	2.19
Claude-3.5-sonnet	2.33	1.37	2.34	2.91	6.02	2.58
Qwen2-vl-7b	1.14	0.83	0.79	1.78	2.38	0.87
Deepseek-vl-7b	1.00	0.04	0.09	1.14	1.17	0.09
Phi3-vl	1.52	0.07	0.21	1.29	1.46	0.12
LLaVA-1.6-7b	1.47	0.13	0.17	1.36	1.54	0.19
InternVL2-8b	1.00	0.26	0.43	1.51	2.10	0.23
Random	2.36	1.15	1.22	3.45	27.01	-

Table 8: Average number of successful steps across different games.

Specifically, taking the Tic-Tac-Toe game as an example, we implemented a random baseline by setting a random player A versus a search-based opponent B. At each turn, A randomly selects a position on the grid without checking whether it is empty or occupied, while B, using a search-based method, always follows the rules. If A violates the rules, the game is terminated. We calculated the average success steps of A, i.e., the average number of A's stones successfully placed on the board. After 1,000 matches, we found the random baseline to be 2.36. We then tested all LVLM-based methods under this setting and found that no model exceeded this random baseline on the Tic-Tac-Toe game, indicating that the existing LVLMs struggle to play the game based on the visual input of the game board. However, this metric still revealed some meaningful trends that commercial models demonstrated better performance than open-source ones, and certain models, such as DeepSeek-VL and InternVL, could only place the first stone but failed to follow the rules in subsequent rounds. We then extended this intermediate evaluation metric to all other games in the benchmark. The results, as shown in Table 8, highlight similar trends across different games.