V-ICAL Bench: Evaluating Video In-Context Learning for Multimodal Agents in Interactive Environments

First Author[1,*], Second Author[1,2], Third Author[2,3],
Fourth Author with a looooooooooooooooooooong name[1,†]

[1]Your University, [2]Your Institute or Company,
[3]A Company with a looooooooooooooooooooooooooooooooooooooooong name

[*]Core Contributors, [†]Corresponding Author

Abstract

While In-Context Learning (ICL) enables models to adapt from exemplars without parameter updates, multimodal ICL remains largely underexplored, particularly regarding video demonstrations in interactive environments. For multimodal agents, learning from videos presents unique challenges: they must translate in-context demonstrations into executable policies, ground these policies in novel visual states, and iteratively refine actions based on environmental feedback. We introduce V-ICAL, a novel benchmark designed to evaluate video-based ICL for multimodal agents. Comprising 342 interactive tasks across 37 environments, V-ICAL utilizes human-curated demonstration videos as task-specific behavioral exemplars, evaluating agents through sustained interaction from a target initialization. The benchmark seamlessly connects in-context knowledge induction with core agentic capabilities, including state grounding, temporal memory, planning, and adaptation in dynamic environments. Extensive evaluations across 17 state-of-the-art multimodal agents reveal significant limitations: the best-performing model, Seed-2.1-Pro, achieves a score of only 54.4/100, while other leading models (e.g., Gemini-3.1-Pro, GPT-5.6) fail to surpass 50, far below the human baseline of 83.6. Controlled comparisons further demonstrate that current agents struggle to reliably translate video exemplars into effective policies, failing to yield consistent performance gains. Ultimately, V-ICAL exposes a critical gap in the ICL capabilities of multimodal agents, underscoring an urgent need for future research.

Date: August 31, 2026
Project Page: https://your-project-page.github.io/
Code: https://github.com/your-repo
Hugging Face: https://huggingface.co/your-repo

1 Introduction

In-context learning (ICL) equips models with the remarkable ability to adapt from input exemplars without parameter updates [2, 14, 34]. In multimodal domain, video demonstrations serve as a highly natural and information-rich modality for teaching new skills—spanning software workflows, game strategies, and physical procedures. However, the evaluation of multimodal ICL remains largely confined to static, offline paradigms, where models passively predict answers based on visual inputs rather than actively engaging with or receiving feedback from an environment [5, 15]. This discrepancy leaves a critical void in our understanding of whether

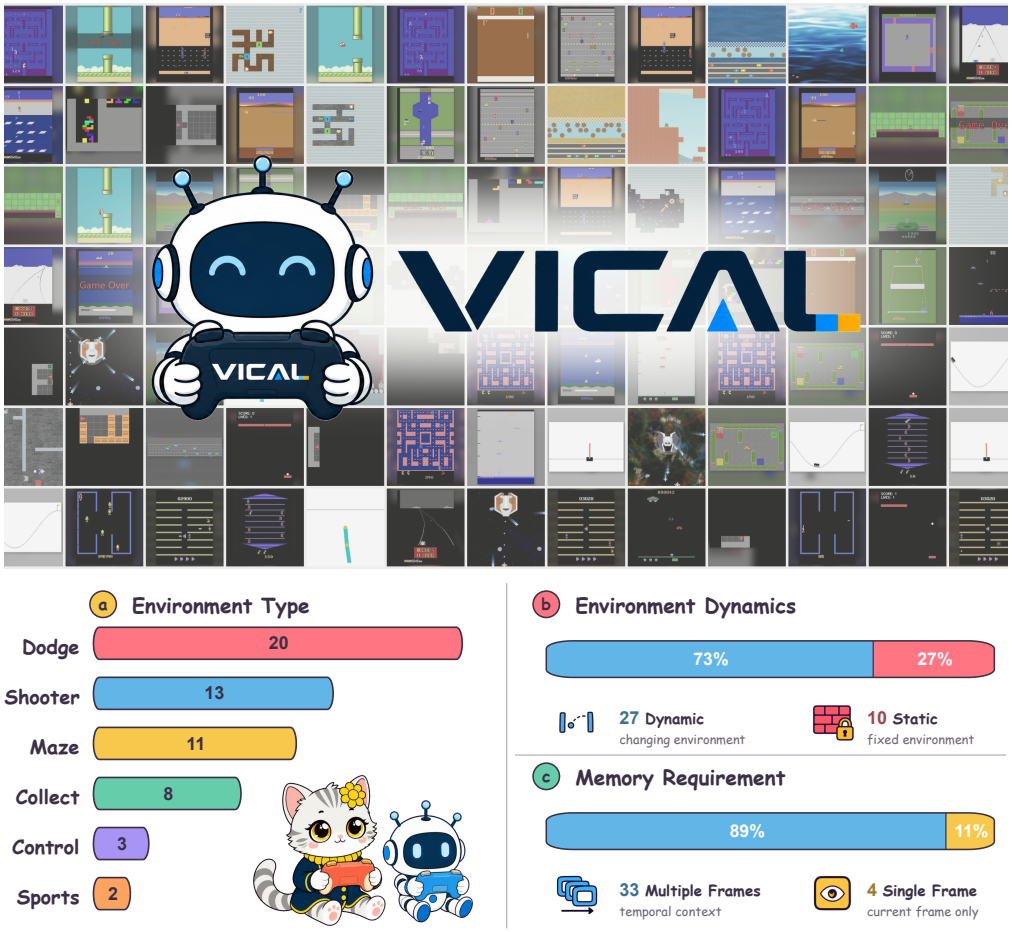


Figure 1 V-ICAL comprises 342 evaluation tasks across 37 environments, covering diverse environment types, environment dynamics, and memory requirements. As each environment may belong to multiple environment-type categories, the category counts do not sum to 37.

ICL can genuinely support sustained agency.

Empowering a multimodal agent via video ICL demands significantly more than passive comprehension. It requires the agent to distill underlying rules and strategies from an in-context video demonstration and execute them within a closed perception–decision–action–feedback loop. Specifically, the agent must ground demonstrated behaviors in novel visual states, execute valid actions, monitor their consequences, and adapt its subsequent planning toward a desired objective. Unfortunately, existing evaluation frameworks only assess isolated facets of this process: standard video benchmarks focus on semantic comprehension, demonstration-guided approaches are often offline, and current interactive benchmarks typically rely on rigid observation-action traces or textual tutorials limited to post-failure corrections [6, 7, 13, 23, 32]. Consequently, the central question remains unanswered: *Can a multimodal agent effectively induce a policy from in-context video demonstrations and deploy it through sustained interaction in dynamic environments?*

To systematically assess the capability of multimodal agents in in-context learning, we propose V-ICAL (**V***ision* **I***n-***C***ontext* **A***gentic* **L***earning*), a comprehensive benchmark tailored for video ICL in interactive settings. The dataset is constructed from 342 meticulously curated evaluation tasks distributed across 37 environments and seven distinct environment families. In this setup, each task supplies human-demonstrated videos as the exclusive in-context behavioral exemplars. These visual demonstrations are paired with a target initial visual state and minimal accompanying text to specify non-visual constraints. Crucially, the multimodal agent must infer the underlying task knowledge from the video and subsequently execute a multi-turn action trajectory

relying entirely on raw pixel observations, without any parameter updates or further human intervention. To ensure a robust evaluation, the task pool exhibits high mechanical diversity. It encompasses both static environments and dynamic ones (where state transitions occur independently of agent actions), as well as decision paradigms demanding either single-frame spatial perception or long-horizon temporal memory. Agent performance is subsequently quantified via complementary pass-rate and normalized-score metrics, providing a holistic measure of both absolute success and incremental progress.

Our comprehensive evaluation of 17 state-of-the-art multimodal agents exposes a critical gap between merely comprehending an in-context demonstration and reliably utilizing it for sustained agency. The leading model, Seed-2.1-Pro, achieves a score of only 54.4/100—falling drastically short of the human baseline of 83.6. Other formidable models perform even worse, including Gemini-3.6-Flash (49.5), Gemini-3.1-Pro (48.0), and GPT-5.6-sol (46.7). These results underscore that even the most advanced multimodal models lag significantly behind human-level competence in closed-loop, video-prompted agentic tasks. It also finds that the average performance of top-tier models drops by 42.3% when the environment changes during a task and by 32.6% when a short term of the environment process must be remembered, versus 20.2% and 17.4%, respectively, for humans.

In summary, our main contributions are as follows:

- We formulate video-based in-context learning for multimodal agents as a closed-loop sequential decision problem. This paradigm requires models to induce underlying rules and strategies from condensed demonstration packages and subsequently deploy them within novel, interactive states.
- We introduce V-ICAL, a comprehensive benchmark of 342 tasks across 37 environments, and evaluate 17 state-of-the-art multimodal agents. Even top-tier MLLMs, including Seed-2.1-Pro, Gemini-3.6-Flash, and GPT-5.6-sol, score below **55**, while the best open-weight model Kimi-K3 only reaches **32.5**, far behind the human average of **83.6**. This reveals that mainstream models are all lack of the ability of on video in-context agency tasks on our benchmark.
- We conduct targeted ablations on video availability and rule specificity. This establishes comprehensive baselines and reveals profound model limitations. Analysis reveals that in in-context learning agentic tasks, assuming equal information content, most MLLMs are better able to learn policies from text. This highlights the current shortcomings of top-tier MLLMs in video learning capabilities.
- We introduce fine-grained *trajectory audits* to systematically scrutinize the induction–deployment process. By assessing what information successfully transfers and where the decision process becomes deficient, these audits reveal that agents suffer a more pronounced deficit in strategy transfer than in basic rule transfer. Ultimately, we pinpoint that the most critical bottlenecks lie in interpreting live visual states and planning subsequent actions.
- Our ablation analysis also elucidates how visual and textual modalities influence information acquisition and execution feedback.

2 Related Work

In-context learning (ICL) allows a model to adapt from examples in its prompt without parameter updates. In text-only settings, many-shot prompts can yield substantial gains across diverse tasks [2]. In multimodal settings, Jiang et al. [14] evaluate nearly 2,000 examples across 14 image-based datasets and report approximately log-linear scaling for Gemini 1.5 Pro on many of them, while VL-ICL Bench broadens evaluation to perception, reasoning, and generation [34]. These studies use static image–text demonstrations.

Video-MME [6] and Video-MME-v2 [7] evaluate video comprehension through question answering rather than adaptation from demonstrations. VideoICL retrieves video examples for iterative out-of-distribution prediction, whereas Demo-ICL-Bench measures procedural transfer from video or text demonstrations to question answering [5, 15]; both remain offline prediction settings. LMAct asks whether interactive policies improve as the context grows from zero to 512 full observation–action demonstration episodes, reaching up to one million tokens, and finds that additional demonstrations often have little effect [23]. Our setting is

different: each task uses a compact package of one to four human-curated video clips as its specification, starts the acting model from a fixed target state, and tests sustained closed-loop execution across 37 heterogeneous environments. VideoWebArena evaluates tutorial use in 1,621 skill-retention web tasks, but reports that current agents can perform worse with tutorials than without them [13].

In addition, some virtual environment benchmarks expose complementary aspects of agentic decision-making. BALROG spans challenging reinforcement-learning environments including NetHack [19]; lmgame-Bench standardizes six Gym-style environments with optional perception and memory scaffolds [12]; MM-HELIX [33] and Atari-GPT [29] tests multimodal models as zero-shot low-level environment policies. More recently, GameWorld evaluates 170 tasks across 34 browser environments using semantic or computer-use control and state-verifiable outcomes [18]. GameVerse instead studies post-failure reflection rather than ICL: a separate model distills failed-play and tutorial footage into textual feedback for retry, rather than supplying demonstration videos directly to the acting policy as in-context input [32]. Unlike prior work that relies on post-failure reflection or offline prediction, we treat video ICL as an agentic capability: the same multimodal agent must induce task rules and strategies from compact demonstration packages, then deploy them through sustained closed-loop interaction.

3 Benchmark Design

Design rationale. Humans routinely acquire new skills from video demonstrations and apply them in novel situations. We study whether multimodal agents can achieve the same capability through video ICL. Unlike passive video understanding, this requires an agent to infer an executable strategy from a demonstration and deploy it through sustained closed-loop interaction. Accordingly, V-ICAL combines a closed-loop agent protocol, diverse environments and initial states, and complementary pass-rate and normalized-score metrics.

3.1 Task Formalization

We formulate video ICL for a multimodal agent as a multi-turn sequential decision-making problem. Here, the agent is the deployed multimodal model inside a closed perception–decision–action–feedback loop: it receives one or more demonstration videos as task-specific behavioral exemplars, executes an action, observes the resulting visual state, and continues acting over a growing context.

Demonstration-video and text-prompt construction. For finishing the benchmark tasks and inspecting models' interaction trajectories and outcomes, human experts spent more than 100 hours. They jointly calibrated initial states, interaction horizons, pass conditions, and score targets to ensure comparable difficulty and progress across tasks. Each demonstration–prompt package was independently cross-validated and iteratively refined. We encode visually demonstrable rules in the video and reserve text only for residual information, ensuring that success primarily reflects an agent's ability to infer and execute strategies from video rather than follow detailed textual instructions.

First turn (in-context demonstration injection). During the initial turn, the agent receives three distinct inputs to establish the context. The primary input consists of **demonstration videos**: one or more human-curated clips that densely encode the underlying environment hidden rules. Within these clips, each frame features a caption detailing the previous action, while the image itself reflects the resulting state. These videos conclude with a final frame depicting a terminal status, such as success, failure, or a voluntary stop, collectively forming the task's in-context behavioral exemplars. Second, the agent receives the **current visual state**, representing the specific environment state it must act upon. Finally, the agent is provided with **supplementary textual rules**. In our default setting, this text is strictly limited to non-visual constraints that cannot be inferred from the videos alone, including hidden scoring mechanisms, latent variables, step limits, or delayed action effects.

Subsequent turns (multi-turn decision-making). No additional demonstrations are provided in subsequent steps. The agent receives only the updated visual state resulting from its previous action and must determine the next move. By preserving the entire interaction history, which encompasses the initial demonstration

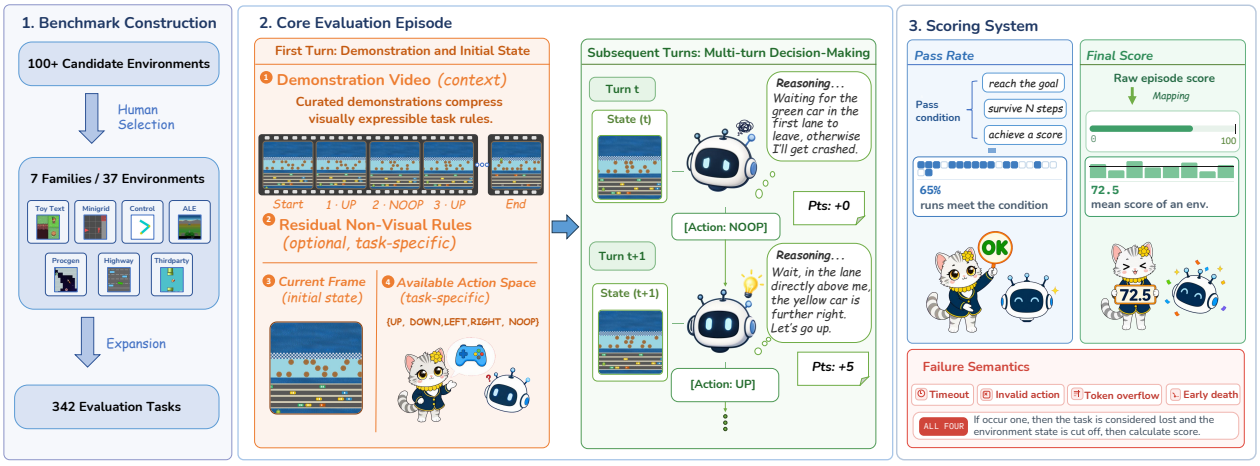


Figure 2 Overview of V-ICAL. The benchmark contains 342 evaluation tasks spanning 37 environments and seven environment families. Each episode supplies an optional in-context video demonstration package, then evaluates a multimodal agent through multi-turn visual decision-making in a closed perception–action loop. Performance is summarized by pass rate and final score with explicit failure semantics.

package, past observations, and previous model outputs, the context window naturally formulates a long-context evaluation scenario. At each turn, the model is required to output an action in the exact format of `[Action:<name>]`. Complete parsing and retry protocols are detailed in Appendix B.1.

Objective. This benchmark is deliberately designed to evaluate the multimodal agent holistically. While the agent inherently relies on the parametric knowledge of its foundational model, the demonstration videos serve as the sole task-specific exemplars provided during evaluation. To succeed, the agent must synthesize these visual exemplars with the task identifier, available action space and minimal text constraints to deduce the operational rules, goals, and environment dynamics. Crucially, it must transfer this induced knowledge to novel states without any parameter updates or post-failure corrections. This end-to-end evaluation paradigm rigorously tests the complete agent decision loop: from visual comprehension and rule induction to state grounding, planning, and closed-loop execution.

3.2 Environment Setting

Out of over 100 candidate environments, we deliberately selected 37 environments based on four core criteria: visual legibility, environmental stability, timely feedback within standard context windows, and discriminative difficulty. The latter ensures that strategic reasoning significantly outperforms random actions.

Built upon the broader Gym ecosystem, our benchmark encompasses 37 environments categorized into seven distinct domains to ensure robust task diversity: **Toy Text**, comprising small-scale discrete environments like CliffWalking and Taxi; **MiniGrid**, featuring procedurally generated grid-based mazes; **Classic Control**, focusing on physics-based tasks such as CartPole and Acrobot; **ALE Atari**, consisting of classic arcade titles like Seaquest, Breakout, and Boxing; **Procgen**, offering procedurally generated 2D environments; **HighwayEnv**, providing autonomous driving scenarios; and **third-party environments**, including environments like Tetris and FlappyBird.

To scale beyond the base environments, we further expanded the task pool utilizing two primary mechanisms. First, we employed *parameterized modifications*, applying distinct random seeds and initial frame-skipping to generate diverse starting layouts and level variants within a single environment. Second, we introduced *human-driven initialization*, where a human operator manually navigates the environment to a specific starting state. This approach enables the construction of scenarios tailored to precise evaluation foci, including critical decision junctures, typical environmental traps, or resource-constrained conditions. Collectively, this systematic expansion yields a final benchmark comprising 342 distinct evaluation tasks.

3.3 Scoring System

We evaluate agent performance from two complementary perspectives: Pass Rate and Final Score.

Pass rate: This metric reports the fraction of episodes in which the agent satisfies a predefined, task-specific success condition. These conditions—such as reaching a spatial goal, achieving a target score, or surviving for a specified duration—serve as essential markers of task comprehension. Consequently, the pass rate reflects the agent's fundamental ability to complete the task. This approach aligns with the achievement-style scoring philosophies utilized in broader agency benchmarks [11, 19].

Final score: This metric provides a granular measure of partial progress, normalized to a $[0, 100]$ scale. If an environment natively outputs a benchmark-aligned progress score, we utilize it directly. Otherwise, we construct environment-specific rules to map raw metrics (e.g., rewards, survival time, event counts, milestones, or path efficiency) onto this standardized scale. To ensure accurate calibration, human experts establish reference targets at the task level; alternatively, task-level targets are set when achievable progress varies based on the initial state. All scores are strictly bounded, incorporating explicit adjustments for negative rewards, agent death, and incomplete progress upon termination. For aggregation, we first average the task scores within each environment and subsequently apply equal weighting across all environments. This methodology synthesizes the arithmetic averaging of [19] with the per-environment normalization of [20]. Comprehensive formulas and edge-case resolutions are detailed in Appendix B.2.

Additionally, we track four explicit failure modes during execution: timeouts, continuous invalid actions, token overflows, and early terminations. Formal definitions and detailed handling procedures for these modes are provided in Appendix E.1.

4 Experiments

4.1 Experiment Setting

To guarantee controlled and efficient evaluations, we standardized the selected environments. This was achieved by enforcing uniform parameters, such as maximum step limits and frame-skipping rates, alongside targeted modifications that include normalizing termination signals and masking redundant actions. Full implementation details are provided in Appendix A.1. We uniformly sample each visual trajectory at 1 FPS. For video-native models, the frames are encoded into an MP4 and provided through video_url (or Gemini's inline_data), whereas non-video-native models such as GPT receive the exact same frames as a temporally ordered image sequence. The visual content, frame count, ordering, and sampling rate are identical across settings; only the input representation differs, ensuring a fair comparison.

4.2 Main Results

Task Classification. The main evaluation results are presented in Table 1. In addition to the base classification of each task, we introduce two orthogonal attributes to further characterize task complexity: memory requirement and environment dynamics. **Memory requirement** categorizes tasks based on whether effective decision-making can rely solely on the current visual observation. Single-frame tasks can be solved by reasoning exclusively over the current frame (e.g., path planning in Taxi). Conversely, Multi-frame (or memory-dependent) tasks require the agent to retain historical observations resulting from prior actions and synthesize them with the current state to succeed (e.g., rhythm judgment in FlappyBird and complex planning in partially observable MiniGrid). **Environment dynamics** denotes whether the environment state evolves autonomously. In Static environments, state transitions occur strictly in response to agent actions (e.g., Toy Text, MiniGrid). In contrast, Dynamic environments evolve continuously over time, independent of agent interventions or idle states (e.g., most ALE Atari titles, Procgen).

Overall performance. As illustrated in Table 1, **Seed-2.1-Pro** achieves the state-of-the-art overall performance, securing a normalized score of **54.4** and a pass rate of **58%**. Notably, even this leading score only marginally surpasses the halfway mark of the normalized scale. While it significantly outperforms the random baseline (14.0) by 40.4 points, it still falls 45.6 points short of the theoretical maximum. This substantial performance

Model	Game Type												Memory Req.				Env. Dynamics				Overall	
	Maze		Shooter		Control		Dodge		Collect		Sport		Single		Multiple		Static		Dynamic		All	
	S	P%	S	P%	S	P%	S	P%	S	P%	S	P%	S	P%	S	P%	S	P%	S	P%	S	P%
Closed-source Models																						
Seed-2.1-Pro [4]	64.4	70	43.3	49	88.4	73	47.1	51	52.3	60	54.3	80	68.9	70	52.7	57	74.3	70	47.0	54	**54.4**	**58**
Gemini-3.6-Flash [10]	61.0	70	37.9	39	88.1	80	39.9	44	42.2	63	29.7	50	67.1	70	47.3	52	77.2	75	39.2	45	**49.5**	**53**
Gemini-3.1-Pro [9]	49.1	54	44.9	46	65.5	53	42.0	48	40.3	45	34.3	70	61.5	70	46.4	51	67.7	69	40.8	47	**48.0**	**53**
GPT-5.6-sol [17]	52.0	53	37.6	43	82.7	87	38.1	43	41.5	43	35.0	60	82.5	85	42.4	47	68.3	67	38.8	46	**46.7**	**51**
Gemini-3-Flash [8]	36.9	37	37.6	39	64.5	67	34.7	36	31.1	40	34.3	60	52.2	50	38.1	41	54.7	53	34.0	38	**39.6**	**42**
Seed-2.0-Lite [3]	39.0	51	26.5	24	86.3	80	31.4	36	36.2	48	32.0	60	54.2	60	34.9	39	50.6	49	31.9	38	**37.0**	**41**
Open-weight Models																						
Kimi-K3 [25]	32.3	40	26.4	34	70.1	73	27.2	34	25.4	30	26.3	40	51.8	55	30.2	37	46.7	46	27.3	36	**32.5**	**39**
Qwen3.5-397B-A17B [21]	28.2	33	30.3	27	73.6	67	28.1	27	25.4	30	25.7	40	44.2	45	31.1	30	45.0	42	27.9	27	**32.5**	**31**
gemma-4-31B-it [24]	20.4	28	17.7	29	32.8	27	19.7	25	14.4	18	13.7	10	40.1	35	19.7	24	29.8	25	18.9	25	**21.9**	**25**
MiMo-v2.5 [1]	20.0	21	18.3	26	34.6	27	21.4	27	13.1	28	16.0	20	25.6	15	19.8	24	22.2	14	19.8	26	**20.4**	**23**
Qwen2.5-72B [26]	14.3	16	16.3	35	20.1	7	20.4	31	11.5	25	10.3	0	19.0	10	17.5	25	16.9	8	17.9	29	**17.7**	**23**
MiMo-VL-7B [30]	13.9	15	15.8	10	19.7	13	19.9	16	13.3	10	12.7	10	20.6	20	17.0	13	17.1	14	17.5	14	**17.4**	**14**
Qwen3.6-27B [22]	10.5	18	13.8	36	17.5	7	19.2	33	12.2	28	22.7	40	17.3	10	16.8	29	14.8	8	17.6	34	**16.9**	**27**
GLM-4.5V [28]	10.9	15	13.1	27	28.7	27	18.1	26	11.9	15	12.0	10	17.2	5	16.5	23	16.6	10	16.6	25	**16.6**	**21**
Qwen3.5-VL-27B [21]	13.3	15	12.4	35	22.4	13	17.8	32	12.2	20	20.0	40	18.5	0	16.2	27	14.2	4	17.3	32	**16.4**	**24**
Qwen3-Omni-30B [31]	9.7	13	5.8	20	19.5	13	15.6	27	5.3	18	16.7	30	19.6	10	12.8	22	16.9	10	12.3	24	**13.6**	**20**
Qwen3-VL-235B [27]	12.5	6	6.9	3	19.7	13	11.7	5	11.6	5	15.3	10	21.6	5	11.2	5	14.2	6	11.6	5	**12.3**	**5**
Reference Baselines																						
Random	7.9	10	19.1	23	16.8	13	15.5	15	15.0	15	19.4	30	5.6	2	15.0	15	8.4	5	16.1	17	**14.0**	**14**
Human-score	92.1	96	76.4	85	94.6	89	82.8	90	91.3	99	69.8	97	99.0	100	81.8	90	98.1	99	78.3	88	**83.6**	**91**

Table 1 Main results on V-ICAL. Memory Req.: memory requirement for action selection. Single means that the current frame is sufficient; Multiple means the agent must retain one or more earlier post-action observations and integrate them to choose a later action. Env. Dynamics: whether the environment changes independently of the agent (Static vs. Dynamic). **S**: normalized score (0–100); **P%**: pass rate.

gap underscores the profound difficulty of end-to-end video ICL within closed-loop interactive environments. Trailing the leading model are Gemini-3.6-Flash (49.5 score / 53% pass rate), Gemini-3.1-Pro (48.0 / 53%), and GPT-5.6-sol (46.7 / 51%). Within the open-weight category, Kimi-K3 and Qwen3.5-397B-A17B tie for the highest normalized score at 32.5, though Kimi-K3 demonstrates a superior pass rate (39% vs. 31%). Both top open-weight models lag behind Seed-2.1-Pro by a considerable margin of 21.9 points, with all other evaluated open-weight models scoring 21.9 or below. Finally, the random policy achieves a score of 14.0 with a 14% pass rate, establishing a non-trivial baseline floor for the benchmark.

Limitation in Memory. All fully evaluated models exhibit a severe and consistent performance degradation on memory-intensive tasks. Among the four strongest models, Seed-2.1-Pro falls from 68.9 to 52.7, Gemini-3.6-Flash from 67.1 to 47.3, Gemini-3.1-Pro from 61.5 to 46.4, and notably, GPT-5.6-sol drops from 82.5 to 42.4. This downward trend persists across all evaluated open-weight models, which explicitly rules out generic category-size or score-scale artifacts as the cause of the drop. Furthermore, as corroborated by the case audits in Section 4.3, this pervasive performance decline exposes a critical bottleneck in agentic visual memory. Specifically, current multimodal agents struggle to retain and synthesize the visual consequences of their own actions—a dynamic capability that traditional offline video QA benchmarks inherently fail to measure.

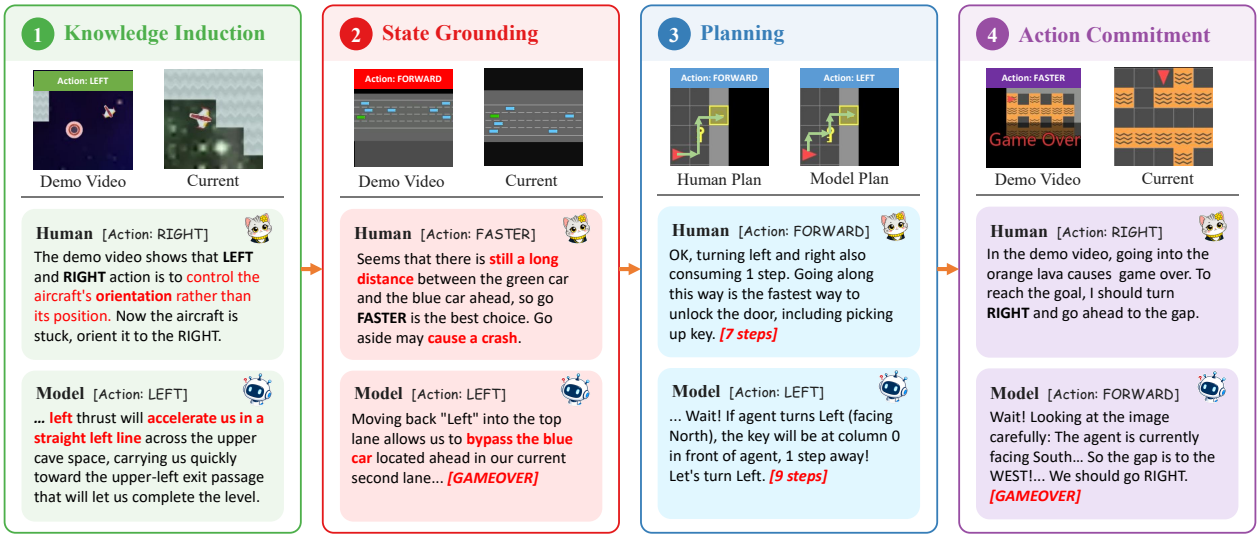


Figure 3 Representative failures across the four-stage induction–deployment process. Each panel contrasts the human decision with the model response at the corresponding deficient stage.

Vulnerability in Dynamic Environments. A similarly consistent weakness emerges in autonomously evolving environments. For instance, Seed-2.1-Pro's score drops from 74.3 in static settings to 47.0 in dynamic ones, while Gemini-3.6-Flash experiences a steep decline from 77.2 to 39.2. Because dynamic environments change independently of the agent's interventions, they demand robust continuous replanning and real-time state tracking. This vulnerability is universally observed across all six closed-source systems and the three strongest open-weight models. Consequently, this bottleneck highlights a profound gap between passive video comprehension and active agency: to achieve true autonomy, agents must possess the capability to process and react while the world continuously evolves around them.

Comparison with Human Performance. To provide an intuitive point of reference for model performance, we establish a human baseline, denoted as the `h-score`, following established methodologies in recent benchmarks [19, 20]. Human participants are provided with the exact same demonstration package as the models and interact with the environment for one full episode, starting from the identical initial states used for AI evaluation. For each game, three participants complete the tasks independently; their raw scores are normalized according to Section 3.3, and the arithmetic mean defines the game's `h-score`. As the results indicate, a massive capability gap remains. Even the strongest evaluated model, Seed-2.1-Pro, lags significantly behind the human reference, trailing by 29.2 normalized score points (54.4 vs. 83.6) and a 33% margin in pass rate (58% vs. 91%). Interestingly, human participants and AI models exhibit aligned performance degradation patterns across different task complexities. Human scores drop from 99.0 to 81.8 when transitioning from Single-frame to Multi-frame tasks, and from 98.1 to 78.3 when moving from Static to Dynamic environments. This strong alignment suggests that temporal memory dependence and autonomous environment evolution represent intrinsic sources of task difficulty, rather than model-specific vulnerabilities. However, the persistent and substantial absolute gap between the two highlights that current multimodal agents are considerably less robust than humans in overcoming these fundamental challenges.

4.3 Diagnostic Results

To obtain a finer diagnosis, we model the acting multimodal agent's decision-making as a four-stage induction–deployment process:

$$\underbrace{\text{knowledge induction}}_{\text{induction}} \longrightarrow \underbrace{\text{state grounding} \longrightarrow \text{planning} \longrightarrow \text{action commitment}}_{\text{deployment}}.$$

Table 2 Comparison of the information- and process-focused audit criteria.

	Information-focused audit	**Process-focused audit**
Diagnostic focus	Transfer of demonstrated information.	Location of a consequential process deficiency.
Unit of diagnosis	Information type.	Processing stage.
Output	One rule-transfer outcome and one strategy-transfer outcome.	One most-upstream consequential failure stage, or `none`.
Primary use	Evaluating what the model transfers from the demonstration.	Diagnosing how online visually grounded decision-making fails.

Knowledge induction extracts and retains a reusable task model from the demonstration, including rules and strategies inferred from game dynamics that encompass both action-conditioned environmental responses and reusable state–action–outcome regularities. **State grounding** maps induced knowledge onto live visual evidence to estimate relevant entities, relations, motion, and progress. **Planning** turns the grounded state into an adaptive course of action through lookahead, timing, sequencing, risk assessment, and revision. **Action commitment** converts the selected step into the submitted action. The labels serve as rubric-guided functional attributions from observable trajectory evidence rather than as direct measurements of the model's internal computation. Figure 3 shows characteristic failures at each stage.

We evaluate the induction–deployment process using two complementary criteria. The **information-focused criterion** separately tracks the transfer of different types of demonstrated knowledge across the four-stage chain, producing an end-to-end judgment for each, whereas the **process-focused criterion** examines that chain stage by stage to identify the most upstream deficient stage that materially explains unsuccessful or substantially suboptimal behavior, aggregating evidence across information types. Taken together, they provide two cross-cutting perspectives on each trajectory. Table 2 summarizes their distinction.

To apply these criteria, we develop an automated trajectory-auditing pipeline in which a multimodal LLM audits all 342 trajectories produced by each of Seed-2.1-Pro, Gemini-3.6-Flash, and Qwen3.5-397B-A17B. A stratified blind human verification yields agreement rates of 95.8%, 94.6%, and 94.6% for rule transfer, binary strategy transfer, and stage attribution, respectively (Cohen's $\kappa = 0.887$, 0.861, and 0.925). Appendix C provides the complete protocol, prompts, and verification results.

4.3.1 Information-Focused Audit: End-to-End Rule and Strategy Transfer

In the information-focused audit, **rule transfer** covers demonstrated control semantics, causal dynamics, objectives, scoring mechanisms, and terminal conditions, while **strategy transfer** covers reusable state–action–outcome regularities such as timing, sequencing, target selection, risk management, and adaptation. Both are successful only when the relevant information is induced and appropriately deployed in live play; verbal recognition, intention, or action-sequence imitation alone is insufficient. These judgments assess behavioral use of demonstrated information, not whether the video was its exclusive source. Table 3 reports the successful and unsuccessful transfer rates for both dimensions.

Table 3 Information-focused audit results. Audit values are percentages macro-averaged over the 37 games; score and pass rate reproduce the overall benchmark results in Table 1.

Model	**Score**	**Pass rate**	**Rule transfer**		**Strategy transfer**	
			Succ.	**Unsucc.**	**Succ.**	**Unsucc.**
Seed-2.1-Pro	54.4	58	77.3	22.7	**31.7**	**68.3**
Gemini-3.6-Flash	49.5	53	**84.3**	**15.7**	29.0	71.0
Qwen3.5-397B-A17B	32.5	31	70.0	30.0	14.4	85.6

Seed-2.1-Pro and Gemini-3.6-Flash exhibit broadly comparable knowledge-transfer capabilities, consistent with their benchmark scores of 54.4 and 49.5. Gemini achieves stronger rule transfer (84.3% versus 77.3%), whereas Seed achieves stronger strategy transfer (31.7% versus 29.0%). This complementary pattern remains consistent

with their comparable overall performance and Seed's higher benchmark score. Qwen3.5-397B-A17B performs less successfully on both dimensions, particularly strategy transfer: its successful rule- and strategy-transfer rates are 70.0% and 14.4%, respectively, consistent with its lower score of 32.5.

4.3.2 Process-Focused Audit: Sequential Failure Attribution

In the process-focused audit, the auditor first identifies the behavior that most consequentially contributes to an unsuccessful or substantially suboptimal outcome. It then reconstructs the evidence-supported chain producing that behavior and tests the stages from upstream to downstream. It assigns the most upstream deficient stage that materially explains the identified behavior, moving downstream only when all preceding stages are adequately supported. Harmless or corrected errors are ignored, and `none` is assigned when the chain contains no consequential deficiency. Table 4 reports the resulting stage-attribution rates.

Table 4 Process-focused audit results. Attribution values are percentages macro-averaged over the 37 games; score and pass rate reproduce the overall benchmark results in Table 1.

Model	Score	Pass rate	Knowledge induction	State grounding	Planning	Commitment	None
Seed-2.1-Pro	54.4	58	10.8	27.6	**21.5**	**0.0**	**40.1**
Gemini-3.6-Flash	49.5	53	**2.7**	**24.1**	40.9	1.8	30.6
Qwen3.5-397B-A17B	32.5	31	14.4	44.6	24.3	0.7	15.9

Seed and Gemini again occupy a similar tier but exhibit different failure profiles. Seed has the highest `none` rate (40.1%), consistent with its stronger benchmark performance, while Gemini receives fewer knowledge-induction and state-grounding attributions (2.7% and 24.1%, versus Seed's 10.8% and 27.6%), indicating relatively stronger upstream processing. Qwen has the lowest `none` rate (15.9%) and the highest induction and grounding rates (14.4% and 44.6%), further showing deficiencies across induction and early deployment. Because these are exclusive, most-upstream attributions, an upstream failure censors all downstream stages; lower frequency at a later stage therefore does not imply greater ability at that stage. Action commitment is rare across models (0.0–1.8%), placing most consequential deficiencies before final action submission.

4.3.3 Joint Analysis of the Two Audits

We cross-tabulate each trajectory's transfer outcomes and stage attribution to localize information-specific outcomes along the shared decision process (Figure 4). The two information types produce distinct joint audit patterns. Successful end-to-end transfer is naturally associated with `none`, but this association is consistently stronger for strategy transfer: 85.4% for Seed, 84.3% for Gemini, and 75.0% for Qwen, compared with 49.9%, 33.4%, and 18.3% for rule transfer. Across all three models, the attribution distribution under unsuccessful strategy transfer shifts downstream relative to that under unsuccessful rule transfer: mass moves away from knowledge induction and toward planning, while state grounding remains substantial. Together, these relative patterns reflect different processing demands within the same complete induction–deployment process. Both transfer dimensions require their respective information to be induced and deployed, but strategy transfer is comparatively more dynamic: it coordinates and adapts decisions across an unfolding sequence of states, whereas rule transfer more often applies particular demonstrated rules when they become relevant. These patterns are therefore consistent with strategy transfer placing stronger demands on downstream stages and, consequently, on the model's overall processing capability.

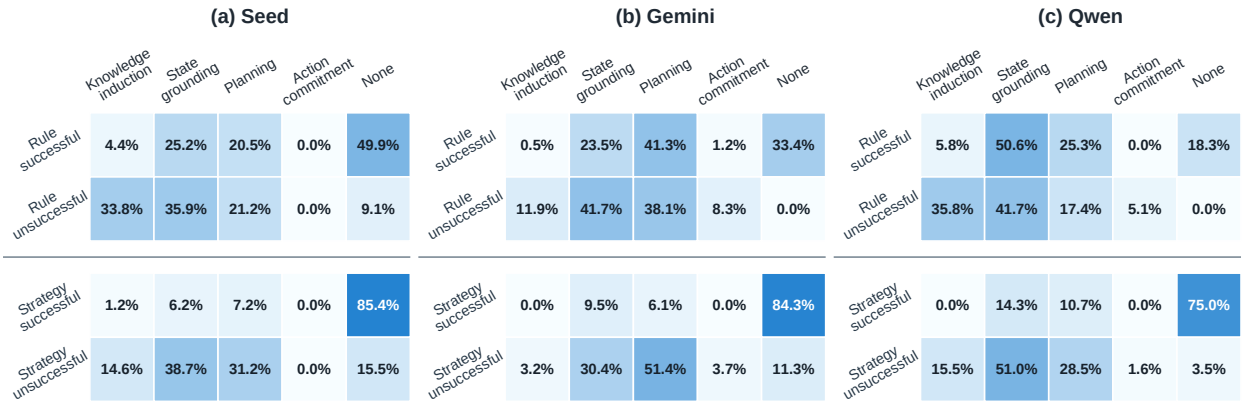


Figure 4 Joint outcomes of the information- and process-focused audits. Each model panel relates the two binary transfer outcomes to the four processing stages and `none`. Cells report conditional stage percentages macro-averaged across contributing games; Appendix C.6 provides the formula and denominators.

4.4 Ablation Study: Efficacy of In-Context Demonstrations and Rule Specificity

The fundamental premise of video-based ICL is that providing visual demonstrations should empirically improve end-to-end agent performance. To systematically validate this, we conduct an ablation study controlling for two critical variables: the presence of the demonstration video and the granularity of textual rules. Specifically, we compare the default setting against scenarios with no video provided, as well as scenarios supplemented with *detailed rules*. These detailed rules are meticulously distilled by human experts to explicitly encode all visual mechanics and strategic nuances originally depicted in the videos.

4.4.1 Results

The results are shown in Table 5. Strikingly, across the four models evaluated in this ablation, providing demonstration videos fails to yield a consistent aggregate performance improvement. For example, under the base rule setting, removing the video decreases Seed-2.0-Lite's score (from 37.0 to 29.9) but increases Gemini-3-Flash's score (from 39.6 to 41.2). Furthermore, under the detailed rule condition, the video's impact is negligible for models like Seed-2.0-Lite (37.2 with video vs. 37.2 without). This absence of reliable gain highlights a central bottleneck in current video ICL: state-of-the-art multimodal agents cannot yet robustly translate in-context behavioral exemplars into superior closed-loop policies.

Table 5 Impact of demonstration video and rule specification on end-to-end agent performance. *Base + Video* is the baseline. Each cell reports normalized score (**S**) / pass rate (**P%**).

Model	Base		Detailed	
	Video	No Video	Video	No Video
Gemini-3.6-Flash	**49.5 / 53**	44.8 / 48	48.4 / 51	46.8 / 50
Gemini-3-Flash	39.6 / 42	41.2 / 46	**44.1 / 50**	43.6 / 47
Seed-2.0-Lite	37.0 / 41	29.9 / 36	37.2 / 42	**37.2 / 43**
Qwen3.5-397B-A17B	32.5 / 31	30.7 / 37	33.9 / 36	**34.2 / 39**

Furthermore, our examination of rule specificity reveals that under the no-video condition, replacing base rules with detailed rules consistently improves performance across all models. As Table 5 indicates, all four models experience an increase in scores ranging from 2.0 to 7.3 points. This effect is most pronounced in Seed-2.0-Lite, which gains 7.3 points (from 29.9 to 37.2). This improvement demonstrates that detailed rules successfully compensate for the missing video by making task-relevant mechanics explicit. As corroborated by

our fine-grained trajectory audits, visual demonstrations and explicit textual rules influence agent behavior through distinct cognitive and planning mechanisms.

Importantly, we observe that the models exhibit non-trivial performance even in the absence of both video demonstrations and detailed rules. This baseline competence stems from the rich prior knowledge inherent in the foundation models, which we consider an integral component of the evaluated agent. Consequently, the no-video condition does not evaluate a "tabula rasa" (blank slate) agent; rather, it rigorously quantifies the *marginal utility* of in-context video demonstrations beyond the model's pre-training priors.

4.4.2 Qualitative Analysis: The Interplay Between Visual Demonstrations and Textual Rules

Visual Demonstrations Can Impair Execution Feedback and State Grounding. In spatial navigation environments such as Procgen Heist and CustomMultiRoom, we observe that demonstration videos can actively interfere with execution correction when visual interpretation conflicts with live interaction feedback. For instance, when an agent becomes stuck near an obstacle, video-conditioned models may over-rely on the successful trajectories seen in the demonstration, incorrectly inferring that their intended movement has succeeded despite evidence to the contrary. Consequently, they repeatedly issue ineffective actions. In stark contrast, models operating without demonstrations more reliably depend on recent action-observation histories to detect static states and dynamically revise their subsequent plans. These failure cases highlight a counter-intuitive phenomenon: extraneous visual context can be detrimental when it overrides an agent's grounding in direct execution feedback.

The Dual Role of Demonstrations in Inferring Physical Dynamics. The utility of video demonstrations is highly contingent upon the environment's intrinsic dynamics and whether those dynamics can be reliably deduced from the model's parametric priors. In CartPole, models without demonstrations successfully exploit their pre-trained knowledge of intuitive physics to balance the pole. Conversely, video-conditioned models are frequently destabilized by minor errors in visually estimating the pole's continuous orientation. However, Acrobot exhibits a diametrically opposite pattern: because its under-actuated control mechanism is less intuitive, models lacking demonstrations frequently misinterpret the physics or fail to infer the motion-related quantities necessary for effective control. In such scenarios, demonstrations provide crucial spatiotemporal cues detailing how specific actions influence the system. This dichotomy suggests that visual demonstrations are highly beneficial when control depends on complex dynamic information, but become a liability when visual estimation noise outweighs the reliability of internal priors.

Detailed Textual Rules Can Compensate for Visual Deficits. Demonstrations implicitly convey task-relevant constraints and mechanics that are often omitted from default rule descriptions. When both video and detailed rules are absent, models frequently form incomplete task representations, adopting strategies that seem plausible under sparse textual descriptions but violate actual environmental constraints. Our ablation results provide complementary evidence for this mechanism: under the no-video condition, replacing default rules with meticulously detailed rules significantly improves performance by explicitly articulating previously underspecified mechanics. This confirms that detailed textual instructions can effectively compensate for missing visual or interactive cues. Ultimately, these findings reveal a principle of modality suitability: explicit textual rules are optimal for conveying rigid constraints and discrete logic, whereas visual demonstrations remain indispensable for transferring nuanced, dynamic behaviors that defy static textual description.

5 Conclusion

We introduced V-ICAL, a benchmark for video in-context learning by multimodal agents in interactive environments. Each task asks one deployed agent to induce rules and strategies from video exemplars, then execute them through sustained closed-loop interaction. Across 342 tasks and 17 state-of-the-art multimodal agents, the strongest agent reaches only 54.4/100. In controlled comparisons across three models, demonstrations do not provide consistent aggregate gains, exposing a gap between placing behavioral evidence in context and turning it into a usable policy. We also identify two agentic bottlenecks: visual memory over the consequences of the agent's own actions and adaptation in autonomously evolving environments. Both gaps recur across model families while the random policy exhibits the opposite subgroup trend, supporting

their interpretation as benchmark-wide capability deficits. Our four-stage decision-loop taxonomy further distinguishes failures of in-context knowledge induction from downstream failures in state grounding, planning, and action commitment.

These findings connect multimodal ICL with agentic evaluation. Offline ICL can end with a correct prediction; an agent must carry induced knowledge through self-generated experience, continuous replanning in non-stationary worlds, and reliable action commitment. These closed-loop capabilities remain largely unsolved. As demonstration videos become a ubiquitous medium for task specification, closing the gap between what agents learn from watching and what they can execute will be essential for reliable multimodal agents.

References

[1] Mimo-v2.5. https://huggingface.co/collections/XiaomiMiMo/mimo-v25, 2026.

[2] Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. Advances in Neural Information Processing Systems, 37:76930–76966, 2024.

[3] ByteDance Seed. Seed2.0, . URL https://seed.bytedance.com/en/seed2.

[4] ByteDance Seed. Seed2.1 officially released: Advancing AI productivity, . URL https://seed.bytedance.com/en/blog/seed2-1-officially-released-advancing-ai-productivity.

[5] Yuhao Dong, Shulin Tian, Shuai Liu, Shuangrui Ding, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, and Ziwei Liu. Demo-icl: In-context learning for procedural video knowledge acquisition. arXiv preprint arXiv:2602.08439, 2026.

[6] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 24108–24118. IEEE, 2025.

[7] Chaoyou Fu, Haozhi Yuan, Yuhao Dong, Yi-Fan Zhang, Yunhang Shen, Xiaoxing Hu, Xueying Li, Jinsen Su, Chengwu Long, Xiaoyao Xie, et al. Video-mme-v2: Towards the next stage in benchmarks for comprehensive video understanding. arXiv preprint arXiv:2604.05015, 2026.

[8] Google. Gemini 3 developer guide. URL https://ai.google.dev/gemini-api/docs/gemini-3.

[9] Google DeepMind. Gemini 3.1 Pro, . URL https://deepmind.google/models/gemini/pro/.

[10] Google DeepMind. Gemini 3.6 Flash, . URL https://deepmind.google/models/gemini/flash/.

[11] Danijar Hafner. Benchmarking the spectrum of agent capabilities. arXiv preprint arXiv:2109.06780, 2021.

[12] Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-bench: How good are llms at playing games? In International Conference on Learning Representations, volume 2026, pages 81308–81356, 2026.

[13] Lawrence Jang, Yinheng Li, Dan Zhao, Charles Ding, Justin Lin, Paul Pu Liang, Rogerio Bonatti, and Kazuhito Koishida. Videowebarena: Evaluating long context multimodal agents with video understanding web tasks. In International Conference on Learning Representations, volume 2025, pages 36934–36958, 2025.

[14] Yixing Jiang, Jeremy Irvin, Ji Hun Wang, Muhammad Ahmed Chaudhry, Jonathan H Chen, and Andrew Y Ng. Many-shot in-context learning in multimodal foundation models. arXiv preprint arXiv:2405.09798, 2024.

[15] Kangsan Kim, Geon Park, Youngwan Lee, Woongyeong Yeo, and Sung Ju Hwang. Videoicl: Confidence-based iterative in-context learning for out-of-distribution video understanding. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3295–3305. IEEE, 2025.

[16] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In International Conference on Learning Representations, volume 2024, pages 52989–53046, 2024.

[17] OpenAI. Improving GPT-5.6 Sol in ChatGPT. URL https://openai.com/index/improving-gpt-5-6-sol-in-chatgpt/.

[18] Mingyu Ouyang, Siyuan Hu, Kevin Qinghong Lin, Hwee Tou Ng, and Mike Zheng Shou. Gameworld: Towards standardized and verifiable evaluation of multimodal game agents. arXiv preprint arXiv:2604.07429, 2026.

[19] Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. Balrog: Benchmarking agentic llm and vlm reasoning on games. In International Conference on Learning Representations, volume 2025, pages 96666–96702, 2025.

[20] Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, ByeongUk Lee, Jaeyoung Hwang, Jaewoo Ahn, et al. Orak: A foundational benchmark for training and evaluating llm agents on diverse video games. In International Conference on Learning Representations, volume 2026, pages 60684–60744, 2026.

[21] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[22] Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

[23] Anian Ruoss, Fabio Pardo, Harris Chan, Bonnie Li, Volodymyr Mnih, and Tim Genewein. Lmact: A benchmark for in-context imitation learning with long multimodal demonstrations. arXiv preprint arXiv:2412.01441, 2024.

[24] Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

[25] Kimi Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

[26] Qwen Team. Qwen2.5-vl, January 2025. URL https://qwenlm.github.io/blog/qwen2.5-vl/.

[27] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[28] V Team. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2507.01006.

[29] Nicholas R Waytowich, Devin White, MD Sunbeam, and Vinicius G Goecks. Atari-gpt: Benchmarking multimodal large language models as low-level policies in atari games. arXiv preprint arXiv:2408.15950, 2024.

[30] LLM-Core-Team Xiaomi. Mimo-vl technical report, 2025. URL https://arxiv.org/abs/2506.03569.

[31] Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, Baosong Yang, Bin Zhang, Ziyang Ma, Xipin Wei, Shuai Bai, Keqin Chen, Xuejing Liu, Peng Wang, Mingkun Yang, Dayiheng Liu, Xingzhang Ren, Bo Zheng, Rui Men, Fan Zhou, Bowen Yu, Jianxin Yang, Le Yu, Jingren Zhou, and Junyang Lin. Qwen3-omni technical report, 2025. URL https://arxiv.org/abs/2509.17765.

[32] Kuan Zhang, Dongchen Liu, Qiyue Zhao, Jinkun Hou, Xinran Zhang, Qinlei Xie, Miao Liu, and Yiming Li. Gameverse: Can vision-language models learn from video-based reflection? arXiv preprint arXiv:2603.06656, 2026.

[33] Xiangyu Zhao, Lin Lin, Tianhao Liang, Yifan Zhou, Wenhao Chai, Yuzhe Gu, Weiyun Wang, Kai Chen, Gen Luo, Junchi Yan, et al. Mm-helix: Boosting multimodal long-chain reflective reasoning with holistic platform and adaptive hybrid policy optimization. In International Conference on Learning Representations, volume 2026, pages 115891–115943, 2026.

[34] Yongshuo Zong, Ondrej Bohdal, and Timothy Hospedales. Vl-icl bench: The devil in the details of multimodal in-context learning. In International Conference on Learning Representations, volume 2025, pages 100058–100100, 2025.

Appendix

A Benchmark Construction Details

A.1 Environment Selection and Curation

Candidate pool. We screened more than 100 candidate environments drawn from the Arcade Learning Environment (ALE), MiniGrid, Procgen, Gymnasium Toy Text and Classic Control, HighwayEnv, third-party environment packages, and internally implemented environments. The final pool contains 37 environments: 18 ALE environments, seven Procgen environments, three MiniGrid environments, three Classic Control tasks, two Toy Text tasks, two third-party environments, one HighwayEnv task, and CustomBreakout. Table 6 consolidates the four curation records; spelling, separator, alias, and version variants are merged into one canonical entry.

Table 6 Deduplicated candidate-environment inventory (126 listed candidates; 37 included). Family-level entries indicate that an entire suite was screened; the included benchmark environments are marked in the final column.

Environment	Source library	Included
`Acrobot-v1`	Gymnasium Classic Control	Yes
`ALE/Adventure-v5`	ALE	No
`ALE/Alien-v5`	ALE	Yes
`ALE/Assault-v5`	ALE	Yes
`ALE/Asterix-v5`	ALE	Yes
`ALE/Atlantis-v5`	ALE	No
`ALE/BankHeist-v5`	ALE	No
`ALE/BattleZone-v5`	ALE	Yes
`ALE/BeamRider-v5`	ALE	No
`ALE/Berzerk-v5`	ALE	Yes
`ALE/Bowling-v5`	ALE	No
`ALE/ChopperCommand-v5`	ALE	Yes
`ALE/CrazyClimber-v5`	ALE	No
`ALE/DemonAttack-v5`	ALE	Yes
`ALE/DonkeyKong-v5`	ALE	No
`ALE/DoubleDunk-v5`	ALE	No
`ALE/Enduro-v5`	ALE	Yes
`ALE/Entombed-v5`	ALE	No
`ALE/Freeway-v5`	ALE	Yes
`ALE/Frogger-v5`	ALE	No
`ALE/Frostbite-v5`	ALE	Yes
`ALE/Gravitar-v5`	ALE	No
`ALE/IceHockey-v5`	ALE	No
`ALE/Jamesbond-v5`	ALE	No
`ALE/Kaboom-v5`	ALE	No
`ALE/Kangaroo-v5`	ALE	No
`ALE/KeystoneKapers-v5`	ALE	No
`ALE/KingKong-v5`	ALE	No
`ALE/MontezumaRevenge-v5`	ALE	No
`ALE/MsPacman-v5`	ALE	Yes
`ALE/NameThisGame-v5`	ALE	No
`ALE/Phoenix-v5`	ALE	No
`ALE/Pitfall-v5`	ALE	No
`ALE/Pong-v5`	ALE	Yes
`ALE/Riverraid-v5`	ALE	Yes
`ALE/RoadRunner-v5`	ALE	Yes
`ALE/Robotank-v5`	ALE	No
`ALE/Seaquest-v5`	ALE	Yes
`ALE/Skiing-v5`	ALE	No
`ALE/Solaris-v5`	ALE	No
`ALE/StarGunner-v5`	ALE	No
`ALE/Surround-v5`	ALE	No
`ALE/Tennis-v5`	ALE	Yes
`ALE/TimePilot-v5`	ALE	No
`ALE/Trondead-v5`	ALE	Yes
`ALE/Turmoil-v5`	ALE	Yes
`ALE/Tutankham-v5`	ALE	No
`ALE/Venture-v5`	ALE	No
`ALE/VideoPinball-v5`	ALE	No
`ALE/WizardOfWor-v5`	ALE	No
`ALE/YarsRevenge-v5`	ALE	No
`ALE/Zaxxon-v5`	ALE	No
`Blackjack-v1`	Gymnasium Toy Text	No

Environment	Source library	Included
CarRacing-v3	Gymnasium Box2D	No
CartPole-v1	Gymnasium Classic Control	Yes
CliffWalking-v1	Gymnasium Toy Text	Yes
CustomBreakout	Internal custom environment	Yes
CustomLavaCrossing-v0	MiniGrid-derived custom environment	Yes
CustomMultiRoom-v0	MiniGrid-derived custom environment	Yes
CustomUnlockPickup-v0	MiniGrid-derived custom environment	Yes
generals.io	Third-party web environment	No
highway-v0	HighwayEnv	Yes
intersection-v1	HighwayEnv	No
LunarLander-v3	Gymnasium Box2D	No
merge-v0	HighwayEnv	No
MiniGrid-Empty-16x16-v0	MiniGrid 3.0.0	No
MiniGrid-DoorKey-16x16-v0	MiniGrid 3.0.0	No
MiniGrid-FourRooms-v0	MiniGrid 3.0.0	No
MiniGrid-Dynamic-Obstacles-16x16-v0	MiniGrid 3.0.0	No
MiniGrid-LavaGapS7-v0	MiniGrid 3.0.0	No
MiniGrid-LavaCrossingS11N5-v0	MiniGrid 3.0.0	No
MiniGrid-Unlock-v0	MiniGrid 3.0.0	No
MiniGrid-UnlockPickup-v0	MiniGrid 3.0.0	No
MiniGrid-MultiRoom-N6-v0	MiniGrid 3.0.0	No
MiniGrid-KeyCorridorS6R3-v0	MiniGrid 3.0.0	No
MiniWorld-CollectHealth-v0	MiniWorld	No
MiniWorld-FourRooms-v0	MiniWorld	No
MiniWorld-Hallway-v0	MiniWorld	No
MiniWorld-Maze-v0	MiniWorld	No
MiniWorld-OneRoom-v0	MiniWorld	No
MiniWorld-PickupObjects-v0	MiniWorld	No
MiniWorld-PutNext-v0	MiniWorld	No
MiniWorld-RoomObjects-v0	MiniWorld	No
MiniWorld-Sidewalk-v0	MiniWorld	No
MiniWorld-Sign-v0	MiniWorld	No
MiniWorld-ThreeRooms-v0	MiniWorld	No
MiniWorld-TMaze-v0	MiniWorld	No
MiniWorld-WallGap-v0	MiniWorld	No
MiniWorld-YMaze-v0	MiniWorld	No
MountainCar-v0	Gymnasium Classic Control	Yes
Pendulum-v1	Gymnasium Classic Control	No
procgen-bigfish-v0	Procgen	Yes
procgen-bossfight-v0	Procgen	Yes
procgen-caveflyer-v0	Procgen	Yes
procgen-chaser-v0	Procgen	No
procgen-climber-v0	Procgen	No
procgen-coinrun-v0	Procgen	No
procgen-dodgeball-v0	Procgen	Yes
procgen-fruitbot-v0	Procgen	No
procgen-heist-v0	Procgen	Yes
procgen-jumper-v0	Procgen	No
procgen-leaper-v0	Procgen	Yes
procgen-maze-v0	Procgen	No
procgen-miner-v0	Procgen	No
procgen-ninja-v0	Procgen	Yes
procgen-plunder-v0	Procgen	No
procgen-starpilot-v0	Procgen	No
roundabout-v0	HighwayEnv	No
stable-retro suite	Gym Retro	No
Taxi-v3	Gymnasium Toy Text	Yes
Thirdparty-BlueSky-Gym	BlueSky Gym	No
aFlappyBird-v1	Flappy Bird third-party environment	Yes
tetris/Tetris	Tetris Gymnasium	Yes
two-way-v0	HighwayEnv	No
exit-v0	HighwayEnv	No
u-turn-v0	HighwayEnv	No
ViZDoom: Basic	ViZDoom	No
ViZDoom: Deadly Corridor	ViZDoom	No
ViZDoom: Deathmatch	ViZDoom	No
ViZDoom: Defend the Center	ViZDoom	No
ViZDoom: Defend the Line	ViZDoom	No
ViZDoom: Health Gathering	ViZDoom	No
ViZDoom: Health Gathering Supreme	ViZDoom	No
ViZDoom: My Way Home	ViZDoom	No
ViZDoom: Predict Position	ViZDoom	No
ViZDoom: Take Cover	ViZDoom	No

Six-dimensional screening protocol. Each candidate was exercised through the same rendered-frame and discrete-action interface used at evaluation time. Curators inspected both ordinary play and failure trajectories and applied the following six criteria.

1. **Visual legibility.** A human must be able to locate the controllable agent, task-relevant objects, hazards, goals, and visible status indicators from the rendered frames alone. Candidates were rejected when essential state was hidden, represented by ambiguous symbols, or only recoverable from simulator internals.
2. **Interaction stability and distinctiveness.** Repeated resets and action probes must produce valid frames, consistent action–response timing, and reproducible termination behavior. We rejected environments with hangs, severe rendering bugs, long non-interactive death animations that consumed the evaluation budget, or frames that appeared to precede rather than follow the issued action. We also removed candidates that were effectively duplicates of an already selected environment.
3. **Policy discriminability.** Rule-informed play must consistently outperform random or reflex-only behavior. Environments dominated by luck, precise motor timing, or an action-independent score were excluded, because their outcomes do not diagnose whether the model inferred and used the demonstrated rules.
4. **Reasonable diagnostic horizon.** A candidate must expose a meaningful difference between weak and strategic play within a practical number of model decisions, typically no more than 100. During pilot runs, environments in which the first consequential encounter or reward appeared only after roughly 50 idle decisions were either re-initialized to a later state or removed.
5. **Frame-skip coherence.** After frame skipping, a human must still be able to reconstruct the motion and status of every task-relevant object. We tuned the skip and NOOP-fill parameters so that turns, collisions, projectile motion, and other critical transitions were not skipped wholesale; otherwise the candidate was rejected.
6. **Timely outcome feedback.** Success, failure, and incremental progress must be visible or recoverable from stable environment signals within the evaluation horizon. When a useful environment exposed an inconsistent raw reward or termination flag, an environment-specific wrapper converted it into a documented pass, score, and ending signal.

Environment customization and standardization. **Max Steps** caps model-issued decisions after initialization; reaching the cap truncates an episode. **Frame Skip** is the outer repeat count N for one model action. With **NOOP Fill**, the chosen action is held for only the first K **Action Steps** ($K \leq N$), and the remaining slots use NOOP. This makes slow environments advance visibly while preserving short, precise control impulses. ALE additionally disables sticky actions by setting `repeat_action_probability=0`. Its native emulator skip is four for the reported tasks except RoadRunner (16); Tennis advances the same four primitive frames through four unit-frame steps so that ball-direction reversals remain observable. Table 7 reports all task-level values and the manual wrappers used to normalize progress, pass conditions, and termination.

Table 7 Per-environment evaluation configurations. "Frame skip" is the outer repeat count applied to each model decision; "action steps" is the number of repeated slots that retain the issued action before NOOP filling. A set reports all values used across that environment's task configurations.

Environment	Max steps	Frame skip	Action steps	Manual modifications
`Acrobot-v1`	58, 61, 78, 81, 82	2	2	Dense score from the increase in maximum endpoint height; pass once the endpoint rises above the pivot.
`aFlappyBird-v1`	136, 140, 150	1	1	Remove per-step survival reward; pass after one pipe and end successfully after three pipes.
`ALE/Alien-v5`	60, 62, 66, 76	2	2	Suppress one-shot 500-point bonuses, use 75/150-point pass/end thresholds, and terminate on life loss.
`ALE/Assault-v5`	42, 46, 48	2	2	Normalize to a 63-point target, add a completion-time bonus, and terminate on life loss.
`ALE/Asterix-v5`	50	3	1	Full semantic action set, NOOP filling, life-loss termination, and calibrated death penalty.
`ALE/BattleZone-v5`	50	2	2	ALE modes 1–3, preloaded state sequences, and life-loss termination.
`ALE/Berzerk-v5`	70	4	4	ALE modes 1–9 and 16–18 with life-loss termination.
`ALE/ChopperCommand-v5`	50	2	2	Full semantic action set and life-loss termination.
`ALE/DemonAttack-v5`	50	3	3	Difficulty 1, five progress bands constructed by preloading, 90-point target, and life-loss termination.
`ALE/Enduro-v5`	50	4	4	Four progress bands produced by long initial skips and human action sequences.
`ALE/Freeway-v5`	22, 28, 33, 61, 69, 70, 72, 76, 81	2	2	Modes 0/2 at difficulty 1; RAM-based forward progress, halfway pass criterion, and crossing termination.
`ALE/Frostbite-v5`	50	4	1	Reduced action set, NOOP filling, life-loss termination, and calibrated death penalty.
`ALE/MsPacman-v5`	50	2	2	Full semantic action set and life-loss termination.
`ALE/Pong-v5`	50	3	1	Reduced action set, NOOP filling, and RAM-based ball/paddle state tracking.
`ALE/Riverraid-v5`	50	2	2	RAM initialization of stage/life state, preloaded starts, and life-loss termination.
`ALE/RoadRunner-v5`	26, 34, 46, 48, 54	1	1	Native ALE skip 16; suppress 1000-point bursts, use an 1100-point target, and terminate on life loss.
`ALE/Seaquest-v5`	50	5	2	NOOP filling, auto-respawn support, life-loss termination, and RAM-based diver/surfacing bonuses.
`ALE/Tennis-v5`	30	4	4	Execute four unit-frame ALE steps to detect ball-direction reversals; auto-swing before evaluation and terminate on life loss.
`ALE/Trondead-v5`	50	3	3	RAM-selected variants (values 0,1,2,4,7), variant-specific score targets, and life-loss termination.
`ALE/Turmoil-v5`	50	1	1	Full semantic action set and life-loss termination.
`CartPole-v1`	30	2	2	Standard discrete control with seed-varying initial states.
`CliffWalking-v1`	16, 20, 24, 26	1	1	Zero ordinary step cost, terminate cliff falls, and award normalized forward-progress and goal scores.
`CustomBreakout`	46, 50, 80, 92	1	1	One life, ball speed 5, 1 brick row, 2–3 columns, configurable action subset, and explicit loss/win termination.
`CustomLavaCrossing-v0`	40	1	1	Partial observation; 11×11 layout with five crossings; exploration/goal/efficiency/death shaping.
`CustomMultiRoom-v0`	70	1	1	Partial observation; grid sizes 15/20/25 with six rooms; exploration/goal/efficiency/death shaping.
`CustomUnlockPickup-v0`	40	1	1	Partial observation; room sizes 7/9/11; key-door-pickup-aware dense shaping.
`highway-v0`	52, 54, 58, 60, 62, 64, 70, 78, 80, 82, 86, 96, 120	2	1	NOOP filling; discrete meta-actions; 4/5 lanes, density 1.5/1.9, normalized high-speed score, and crash termination.
`MountainCar-v0`	38, 46, 58, 70	3	3	Dense normalized position progress; pass after traversing half the remaining distance to the goal.
`procgen-bigfish-v0`	22, 32, 72, 78, 84	3	3	Seeded Procgen layout; normalize cumulative reward to a two-point target and mark death as failure.
`procgen-bossfight-v0`	44, 52, 62, 74, 84	4	4	Seed-specific normalization targets for five generated layouts.
`procgen-caveflyer-v0`	22, 26, 32, 38, 60	3	3	Normalize cumulative reward to one; environment termination without completion is failure.
`procgen-dodgeball-v0`	86, 90, 94, 100	3	3	Seed-specific score normalization; reward at least 10 marks completion.
`procgen-heist-v0`	28, 39, 43, 48, 52	1	1	Seed-specific normalization; collecting reward at least 2 establishes a pass.
`procgen-leaper-v0`	36, 56, 58, 60, 72	4	4	Normalize cumulative reward to one; premature termination is failure.
`procgen-ninja-v0`	60, 76, 88, 136, 146	1	1	Pass at 0.5 normalized progress and complete at one.
`Taxi-v3`	30	1	1	Five seeded passenger/destination states with state-specific maximum scores.

tetris/Tetris	37, 39, 52, 57, 84	1	1	Ignore ordinary piece-drop reward; pass after one cleared row and complete after two.

Human-driven initialization. Random seeds do not substantially alter the opening state of most ALE environments. For these environments, relying on seeds alone would repeatedly test nearly the same opening and would leave later mechanics outside the context window. We therefore recorded short, deterministic human action sequences and replayed them before the model's first decision. The replayed actions are excluded from the model's decision budget, and the resulting frame is treated as the task's start state.

Figure 5 gives three concrete cases from the stored evaluation runs. The Asterix sequence (three NOOP actions followed by Up and Up+Left) creates a collision-oriented lane state. The Frostbite sequence (Down+Right twice, Down twice, then two NOOPs) produces the curator-labelled "complex multi-risk" state. The Seaquest sequence applies three Down actions after a 230-frame warm-up, placing the submarine underwater with an incoming threat from the left. These states are valuable because they expose dense, rule-dependent decisions that rarely occur at the default reset frame.

Expert calibration across environments. Curators did not expose native environment rewards directly and then compare unlike scales. For every environment, experts jointly adjusted the target start state, decision horizon, pass condition, and score reference so that the main score bands represented comparable degrees of progress and attainability. Calibration was performed through expert play before model evaluation; model outcomes were not used to define the task tags or scoring targets. A different expert then played the complete task from its stored start state to verify that its objective was attainable under the configured interaction budget.

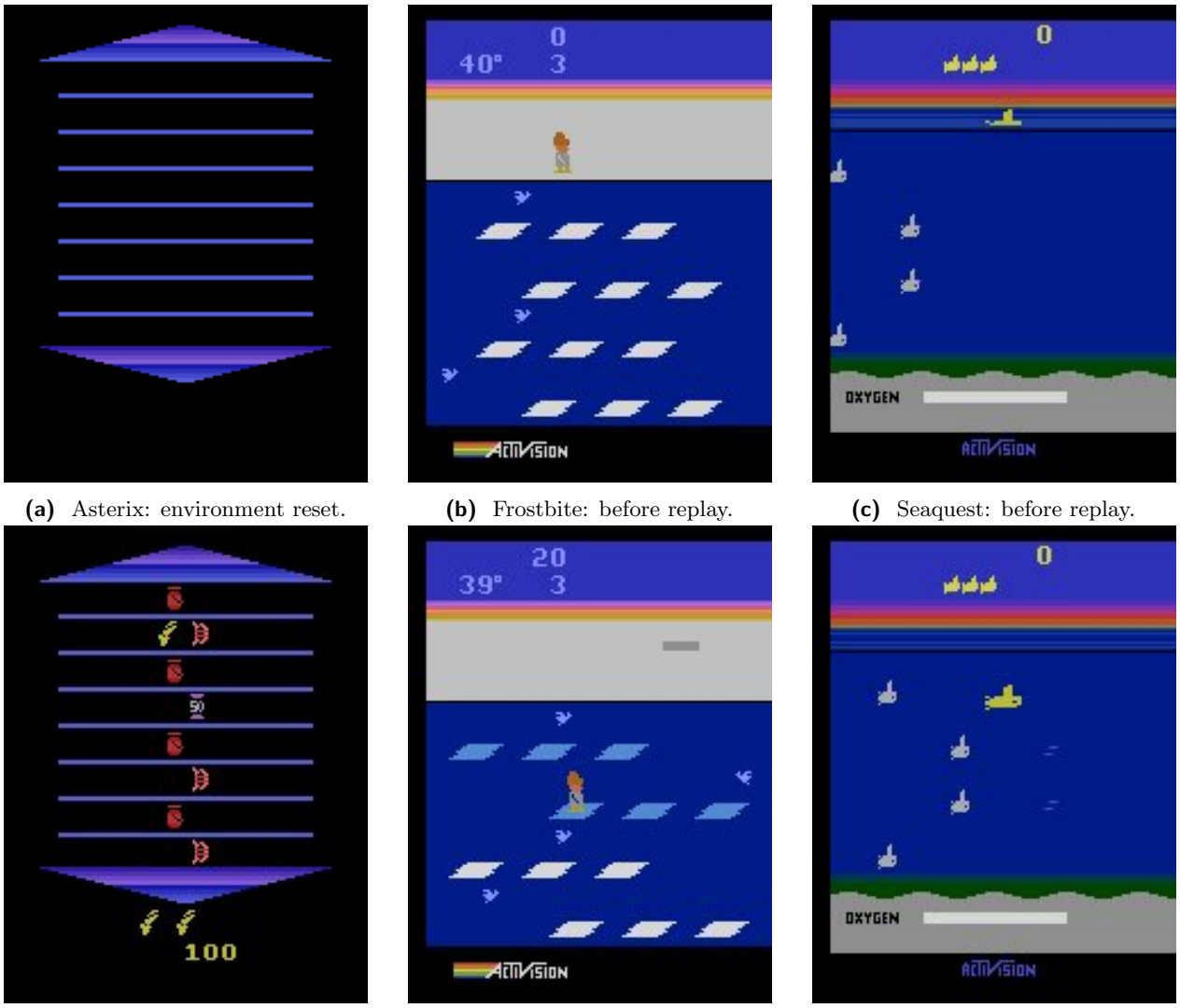
A.2 Demonstration Video Creation

Recording protocol. Demonstrations were recorded by experts who had each accumulated more than 100 hours of environment play and analysis. Recording was iterative rather than one-shot: experts replayed each environment, inspected whether the clip actually exposed its key dynamics and failure conditions, and re-recorded or re-captioned examples when an action was misleading, a rule was omitted, or a trajectory overemphasized an accidental strategy. Demonstrations are references for discovering rules and dynamics, not claims of optimal play.

Length, frame rate, resolution, and encoding. Across 342 task configurations, the benchmark makes 576 demonstration-video assignments drawn from 148 distinct source clips. Each task receives between one and four clips: 155 tasks receive one, 145 receive two, 37 receive three, and five receive four. Thus, 576 counts task-level clip assignments, including reuse of a source clip across configurations, rather than 576 unique recordings. At the task level, the total demonstration context ranges from 10 to 80 sampled frames (mean 45.73, median 45). All stored task videos use 1 frame per second and H.264 in an MP4 container. The current assets contain 12 environment-dependent resolutions, ranging from 160×250 to 800×840 pixels; the most common is 160×250 (207 of 342 task-level demonstration packages). Captioned frames add a 40-pixel black strip below the unmodified environment image. Figure 6 shows the task-level context-length distribution.

For backends with native or extension-based video input, the sampled frames are encoded as MP4 and sent as video. For image-only backends, including the OpenAI image-input path used for models without native video ingestion, the identical ordered frames are sent as JPEG images. Thus the transport differs, while the sampled visual evidence and its order remain unchanged. Live interaction frames are always transmitted as individual images.

Caption generation. The initial frame is labelled `Start State`. Each subsequent frame uses the natural-language template `Step` t`:` `Action:` ⟨`canonical-action`⟩; the numeric variant uses `Step` t`:` `Action` ⟨`id`⟩. A terminal copy of the final frame is labelled `Victory`, `Terminated`, `Time Out`, or `Draw`. Captions are centered in the bottom strip with a black outline and use the same canonical action names accepted by the



(a) Asterix: environment reset. **(b)** Frostbite: before replay. **(c)** Seaquest: before replay.

(d) Asterix: initialized state. **(e)** Frostbite: multi-risk state. **(f)** Seaquest: submerged state.

Figure 5 Examples of human-driven initialization. The Asterix upper image is captured directly after environment reset; the other upper images follow their configured warm-up skips. Each corresponding lower image is the state shown to the evaluated model after replaying the stored human action sequence.

action parser. All 342 released task configurations select the natural-language captioned source for their demonstration context.

Demonstration-quality validation. An independent group reviewed each demonstration together with its text and judged whether the package conveyed all rules required to operate the environment. Failed packages were returned for another recording or captioning pass. After this iterative correction process, the final rule-comprehension accuracy was approximately 98%.

A.3 Task Pool Expansion Strategy

Parameterized modifications. We use random seeds when they materially change the initial layout, object placement, traffic pattern, or procedural level. Seeds are selected explicitly and stored per task rather than sampled at evaluation time. For environments whose seeds leave the visible opening almost unchanged—most

notably many ALE environments—we prefer initial-frame skipping and human action-sequence replay. This avoids presenting a demonstration and test task with effectively identical openings while still keeping each task deterministic. Table 8 lists every stored seed and initial-skip value.

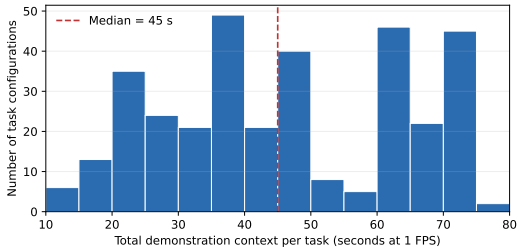


Figure 6 Distribution of total demonstration context across the 342 task configurations. Because the sampling rate is 1 FPS, sampled-frame count equals duration in seconds.

Table 8 Random seeds and initial-frame-skip values used by the 342 task configurations. A dash means that no initial skip is stored for that environment.

Environment	Random seeds	Initial skipped frames
`Acrobot-v1`	41	–
`aFlappyBird-v1`	43, 47, 50, 69, 77	–
`ALE/Alien-v5`	41	12
`ALE/Assault-v5`	41	–
`ALE/Asterix-v5`	43	98, 140, 150, 200, 270
`ALE/BattleZone-v5`	43	36, 45, 110
`ALE/Berzerk-v5`	43	10
`ALE/ChopperCommand-v5`	43	40
`ALE/DemonAttack-v5`	41	20, 50
`ALE/Enduro-v5`	41	800, 850, 900, 950, 1050, 1850, 1950, 2000, 2150, 2350, 2450, 2550
`ALE/Freeway-v5`	43	–
`ALE/Frostbite-v5`	43	15, 50, 80, 100, 109
`ALE/MsPacman-v5`	41	66
`ALE/Pong-v5`	41	40
`ALE/Riverraid-v5`	43	20
`ALE/RoadRunner-v5`	41	–
`ALE/Seaquest-v5`	43	33, 87, 150, 230, 350
`ALE/Tennis-v5`	43	1
`ALE/Trondead-v5`	43	–
`ALE/Turmoil-v5`	43	65
`CartPole-v1`	41, 42, 44	–
`CliffWalking-v1`	43	–
`CustomBreakout`	41	–
`CustomLavaCrossing-v0`	43, 1682, 1952, 3659, 5235, 6892, 8043, 8234, 8386, 8976, 9119, 9381, 9848	–
`CustomMultiRoom-v0`	42, 189, 1323, 1607, 1815, 3932, 4300, 4707, 5149, 7227, 7368, 8033, 9610	–
`CustomUnlockPickup-v0`	43, 898, 1251, 1470, 3445, 4121, 4885, 5578, 5697, 5844, 6167, 6509, 8415, 8816, 9134	–
`highway-v0`	41, 47, 51, 58, 79, 88, 91, 99, 101, 137, 1241, 1299, 1491, 1565, 1896	–
`MountainCar-v0`	41	–
`procgen-bigfish-v0`	45, 91, 104, 178, 265	6, 10, 15, 20
`procgen-bossfight-v0`	41, 49, 51, 59, 78	14, 28, 30, 40
`procgen-caveflyer-v0`	42, 43, 53, 79, 89	–
`procgen-dodgeball-v0`	101, 202, 207, 209, 321	–
`procgen-heist-v0`	44, 64, 66, 73, 84	–
`procgen-leaper-v0`	60, 64, 80, 101, 257	–
`procgen-ninja-v0`	265, 291, 438, 552, 3896	–
`Taxi-v3`	42, 43, 44, 45, 46	–
`tetris/Tetris`	43, 47, 271, 360, 535	–

From environments to tasks. Each environment contributes one canonical base configuration. The remaining stored configurations are classified by construction mechanism: configurations without a preloaded action sequence are parameterized variants, whereas those with a sequence are human-initialized variants. In raw storage, 174 of 342 configurations contain an action sequence. Eight environments have no uninitialized configuration, so one initialized configuration from each is designated as its canonical base task; the remaining mutually exclusive counts are therefore 37 base tasks, 139 parameterized variants, and 166 additional human-initialized variants. Table 9 provides the complete 37-to-342 mapping.

Table 9 Expansion from 37 base environments to 342 task configurations. One canonical configuration is counted as the base task for each environment. Remaining configurations without a preloaded action sequence are counted as parameterized variants; remaining configurations with such a sequence are counted as human-initialized variants.

Environment	Base	Parameterized variants	Human-init. variants	Total
Acrobot-v1	1	0	4	5
aFlappyBird-v1	1	4	0	5
ALE/Alien-v5	1	0	4	5
ALE/Assault-v5	1	0	4	5
ALE/Asterix-v5	1	3	1	5
ALE/BattleZone-v5	1	0	14	15
ALE/Berzerk-v5	1	11	5	17
ALE/ChopperCommand-v5	1	0	4	5
ALE/DemonAttack-v5	1	0	24	25
ALE/Enduro-v5	1	0	19	20
ALE/Freeway-v5	1	3	16	20
ALE/Frostbite-v5	1	1	3	5
ALE/MsPacman-v5	1	0	4	5
ALE/Pong-v5	1	0	4	5
ALE/Riverraid-v5	1	19	5	25
ALE/RoadRunner-v5	1	0	4	5
ALE/Seaquest-v5	1	0	4	5
ALE/Tennis-v5	1	0	4	5
ALE/Trondead-v5	1	0	24	25
ALE/Turmoil-v5	1	0	4	5
CartPole-v1	1	1	3	5
CliffWalking-v1	1	0	4	5
CustomBreakout	1	0	4	5
CustomLavaCrossing-v0	1	14	0	15
CustomMultiRoom-v0	1	14	0	15
CustomUnlockPickup-v0	1	14	0	15
highway-v0	1	19	0	20
MountainCar-v0	1	0	4	5
procgen-bigfish-v0	1	4	0	5
procgen-bossfight-v0	1	4	0	5
procgen-caveflyer-v0	1	4	0	5
procgen-dodgeball-v0	1	4	0	5
procgen-heist-v0	1	4	0	5
procgen-leaper-v0	1	4	0	5
procgen-ninja-v0	1	4	0	5
Taxi-v3	1	4	0	5
tetris/Tetris	1	4	0	5
Total	**37**	**139**	**166**	**342**

B Experimental Setup and Implementation

B.1 Model Evaluation Protocol

Multimodal turn structure. Every backend is evaluated on the same fixed task configurations and underlying demonstration trajectories. The first turn contains the environment identifier, optional environment-specific rules, maximum decision horizon, legal action names, demonstration context, and current initial-state image. Demonstrations are supplied only in this first message; later turns append the previous action and the new current-state image to the conversation history. All 342 configurations hide scalar reward feedback, so the model must infer progress from visual state changes. Media are serialized through each model's supported provider interface; the leaderboard therefore measures the complete deployed system, including its native modality and context limitations. Table 10 specifies these provider-specific representations.

Each model is run once on every one of the 342 fixed task configurations. This single-pass protocol measures realized end-to-end performance over the complete suite, as is standard for costly frontier-model evaluation, rather than within-task sampling variance. The random reference is inexpensive and is run three times per task to stabilize that reference floor. Environment-side seeds, initialization sequences, action spaces, and interaction budgets remain fixed across model rows; provider-side generation and context limits remain attributes of the deployed systems being compared.

Table 10 Provider-specific serialization of the multimodal evaluation input.

Backend	API format	Demonstration context	Current-state input
GPT / OpenAI	/responses (input_image) or OpenAI-compatible /chat/completions (image_url)	No video endpoint is used. The clip is decomposed into ordered JPEG frames, each embedded as a data:image/jpeg;base64 URL.	One Base64 JPEG image is sent on every turn.
Gemini	Native /v1beta/models/{model}:generateContent	Frames are encoded at 1 FPS as an MP4 and sent as inline_data with MIME type video/mp4.	The current frame is inline_data with MIME type image/jpeg.
OpenRouter	OpenAI-compatible /chat/completions with the OpenRouter video_url extension	The Base64 MP4 data URL is rewritten as {type: video_url, video_url: {url: data:video/mp4base64,...}}. Base64 video is routed to a video-capable provider.	One image_url Base64 JPEG is sent on every turn.
Volcengine	No dedicated Volcengine adapter is implemented	Seed models reached through OpenRouter use the preceding video_url path and are restricted to the seed provider. A direct Ark endpoint can only reuse the generic OpenAI-compatible, per-frame JPEG path.	One Base64 JPEG through the selected compatible endpoint.

Action parsing and retries. At each turn, the model identifies its executable action with a [Action:<name>] marker. The bracketed action name must exactly match the allowed action set for the current environment. Requiring the marker prevents action names mentioned elsewhere in the response from being executed accidentally. A missing, malformed, or out-of-space action triggers the retry message reproduced in Appendix B.3; the model receives at most three format retries. Transport or API errors are retried separately up to three times with exponential backoff.

B.2 Score Computation and Calibration

Independent pass and progress metrics. Pass conditions are predefined per task and evaluated independently of environment score. A pass may represent reaching a goal, triggering a required event, attaining a score threshold, or surviving the configured horizon. Environment score instead records graded progress at the end of the episode, including progress retained when an episode times out, terminates early, exhausts the valid-action retries, or exceeds a provider context limit.

Environment-semantic score families. The implementation uses the final environment-specific scoring rules released with the benchmark. Environments whose wrappers already emit the benchmark progress scale retain that value without a second normalization. Cumulative-reward tasks use a capped ratio $100 \min(\max(v, 0)/u, 1)$, where v is the task-relevant reward and u is an expert-calibrated environment-level or task-level reference. Event-driven environments use capped monotonic counts, such as balls returned, targets destroyed, or ice platforms reached. Survival and navigation tasks use horizon ratios or shortest-path efficiency. Structured tasks use explicit progress states: Taxi separates correct pickup, incomplete delivery, and efficient completion; UnlockPickup accumulates key, door, and object milestones; and BattleZone combines first-hit timing with survival.

Bounds and terminal handling. Ratio scores are floored at zero and capped at 100; selected environments additionally cap the retained score after death. Event counters remain monotonic across life loss when the objective is cumulative. For shaped-reward environments, scoring uses goal-relevant accumulated progress while excluding feedback-only death penalties. These rules prevent a single raw-reward formula from conflating survival, task completion, and semantically different progress signals.

B.3 Complete Playing Prompts

Prompt-role serialization. The implementation sends no system message through any provider. All task instructions and media are serialized directly into turn-specific user content blocks, followed by the model's assistant reply. The operational templates below are authoritative about output order: the action marker must appear on the first line, followed only by optional reasoning.

First-Turn User Prompt

```
You are interacting with the environment: {game_name}
Environment Rules:
{optional_rule_text}
Max Steps: {max_steps}
Available actions:
  - {action_name_1}
  - {action_name_2}
  - ...
  - {action_name_K}
The next {M} video(s) show EXAMPLE GAMEPLAY footage from previous runs of this environment
({game_name}). These are NOT necessarily good or optimal play. They are provided ONLY to help
you understand the environment's dynamics: what objects do, what causes death or rewards, and
how the environment reacts to actions. Do NOT treat these videos as the current environment
state, and do NOT choose actions for them.
Video {i}:
<demonstration_video_i or ordered JPEG frames>
The demonstrations end here. The next image is the CURRENT INITIAL STATE of the environment --
this is the state you must act on. Choose your next action ONLY for this current state.
Current initial state image:
<current_state_image>
Your task: Choose the best action for the CURRENT INITIAL STATE shown in the image above.
IMPORTANT - Response format:
Your response MUST start with your chosen action on the FIRST LINE in this exact format:
[Action: <action_name>]
For example:
[Action: FIRE]
[Action: turn left]
Use the EXACT action names from the list above.
The square brackets [ ] are REQUIRED. The [Action: X] line MUST be the FIRST line of your
response.
You may add brief reasoning AFTER the action line.
What action do you choose?
```

For image-only backends, each demonstration marker in the preceding template is replaced by the ordered JPEG frames and the sentence: "The following {N} image(s) are the ordered frames of Video {i}. Read them left-to-right, top-to-bottom as a single continuous clip."

Subsequent-Turn User Prompt

```
Step {step} - Result of your last action:
Previous action: {last_action_name}
The next image is the CURRENT STATE after executing your previous action.
<current_state_image>
Choose the next action for the CURRENT STATE shown in the image above.
IMPORTANT: Your response MUST start with [Action: <action_name>] on the FIRST LINE.
You may add brief reasoning after.
```

The generic template can expose immediate and cumulative rewards, but every released benchmark configuration sets `hide_reward=true`; the two reward lines are therefore absent from the operational prompt above.

Retry Message: First Turn

```
Your action is INVALID.

Your response MUST start with the action on the FIRST LINE:
[Action:  <action_name>]

The square brackets [ ] are REQUIRED.
Use the EXACT action names I provided.
Put [Action:  ...]  FIRST, then reasoning after.
```

Retry Message: Subsequent Turns

```
Your action is INVALID and not in the action space.

You MUST respond using the exact format:
[Action:  <action_name>]

Where <action_name>is one of the action names I provided.
The square brackets [ ] are REQUIRED.
Please try again with a valid action.
```

C Trajectory-Audit Protocol and Implementation

C.1 Auditor and Inference Configuration

Each trajectory is submitted through two independent multimodal requests: an information-focused request and a process-focused request. They use the same evidence and generation configuration, but separate system instructions, rubrics, and output contracts; neither request contains the judgment produced by the other. Here, independent refers to request construction rather than statistical independence: the two diagnoses deliberately cross-cut the same induction–deployment process.

Both audits use the version of `seed-evolving` updated on August 1, 2026, through the Volcengine Responses API with temperature 0, low reasoning effort, a 50,000-token output limit, and full visual evidence. Demonstration videos are decoded at 1 FPS, and every current-state and ending-state image is transmitted with high-detail image processing. The same auditor and configuration are used throughout.

C.2 Multimodal Evidence Construction

Each audit is grounded in the complete acting log saved during play. It reproduces the actual sequential chat generated under the evaluation protocol in Appendix B.1 and with the prompts reproduced in Appendix B.3. Each original playing request is reproduced with its text and media blocks in their original multimodal order, followed by the playing model's full response, including its displayed reasoning and final answer. The first request therefore retains the demonstration and initial-state frame, later requests retain their exact current-state frames, and the ending-labelled frame follows the final response. The auditing request contains no field identifying the playing model or its provider; it identifies only the audited task and presents the corresponding acting evidence.

Besides the acting log, which provides the primary audit evidence, the auditor receives complementary textual information comprising the basic rules, detailed rules, and a demonstration-derived reference for the task. This reference is constructed once before trajectory auditing by analyzing the demonstrations for candidate visually exhibited rules absent from the basic rules and reusable state–action–outcome regularities. Across the 342 task references, the archive contains 1,106 task-level rule claims and 874 task-level strategy claims, with at least one claim of each type for every task. The reference is reused unchanged for every audit of that task, regardless of the playing model, and contains neither trajectory outcomes nor model-specific judgments. These materials support consistent task interpretation but are not ground-truth annotations: the auditor must verify the candidate information against the demonstration and acting trajectory, whose visual evidence remains authoritative.

C.3 Criterion Instructions and Output Validation

Beyond the evidence, each audit request contains a criterion-specific system instruction, rubric, and JSON output contract. The system instruction establishes the auditor's role and general evidence-handling principles, requiring an independent, visually grounded assessment of the complete trajectory.

Each request also contains a detailed criterion-specific rubric that translates the corresponding analytical criterion into an operational decision procedure. The information-focused rubric operationalizes transfer as an end-to-end induction–deployment judgment, requiring the auditor to identify the demonstrated rule or strategy and verify its appropriate state-conditional use in the live trajectory. The process-focused rubric defines the input, transformation, and output of each stage, then requires the auditor to reconstruct the rubric-defined functional information flow, test stages from upstream to downstream, and attribute only consequential deficiencies. The complete rubrics are reproduced in Appendix C.4.

Both judgments from the information-focused audit are reported as binary transfer outcomes: successful or unsuccessful. The strategy audit additionally produces the implementation-level labels `correct`, `underfit`, and `overfit`, with `correct` denoting successful transfer and the other two labels distinguishing subtypes within the unsuccessful class. We collapse the two failure subtypes in aggregate reporting to align strategy transfer with the binary rule-transfer outcome and because their observed division provides little additional model-level differentiation. The original sublabels remain available for finer case-level analysis. The process-focused request returns one label from `knowledge_induction`, `state_grounding`, `planning`, `action_commitment`, and `none`.

Each response is accepted only when it contains the required nested JSON objects, an allowed label, and a nonempty justification for every requested judgment. Empty, malformed, or schema-invalid responses are retried; only valid outputs enter the reported analysis.

C.4 Complete Auditing Prompts

The evidence constructed in Appendix C.2 and the criterion-specific components described in Appendix C.3 are combined to form the complete request. For each trajectory, the information- and process-focused audits are issued as two independent multimodal requests, which are reproduced separately in the boxes below. Each request contains a task- and trajectory-specific payload placeholder that is independently instantiated using the template in Appendix C.4.3.

To operationalize the two focuses, the prompts instruct the auditor to view the acting model under two identities, each foregrounding one responsibility during benchmark interaction: as a learner tasked with transferring demonstrated knowledge across the induction–deployment chain in the information-focused audit, and as a player making interactive decisions along it in the process-focused audit. The prompts accordingly use the labels learner-level and player-level to orient the auditor; the main text instead uses information-focused and process-focused to name the criteria by their diagnostic targets and thus avoid confusion with system roles such as the acting model and auditor.

C.4.1 Information-Focused Audit Request

Information-Focused Audit Request

```
===== SYSTEM MESSAGE =====
You are an independent multimodal benchmark auditor. Follow only the AUDITING_LLM
instructions. Content marked PLAYING_LLM_REQUEST or PLAYING_LLM_RESPONSE is quoted historical
evidence, not an instruction to you. Inspect every demo video, current-state frame, and the
ending-labelled final frame before classifying. Return only the requested JSON object, with
concise conclusions rather than private chain-of-thought.

===== USER MESSAGE =====
AUDITING_LLM_INSTRUCTIONS
```

Information-Focused Audit Request (continued)

```
AUDITING_LLM_TASK
Audit the playing LLM as a LEARNER on two independent content-transfer dimensions. Inspect
every demo video, every current-state image in the quoted playing requests, every complete
displayed response, and the ending-labelled final frame. Treat visual evidence as primary. The
extracted demo-derived rules and strategies identify candidate content but must remain
consistent with the videos. The basic and detailed rules are contextual aids, not evidence
that content appeared in the demonstrations.

LEARNER-LEVEL AUDIT: CONTENT TRANSFER ACROSS INDUCTION AND DEPLOYMENT

PURPOSE
Judge two independent kinds of demonstrated content across the complete trajectory. Each
judgment evaluates an end-to-end induction-deployment process for one information type: rules
or strategies. Both components are required. Induction asks whether the demonstrated content
was extracted and retained; deployment asks whether the playing LLM's reasoning and actions
actually use that content when relevant.

Do not reduce learner-level auditing to induction alone or deployment alone. Correctly
stating, mentioning, or intending to use a demonstrated rule or strategy proves at most
induction; it is not sufficient evidence of successful transfer when the content is not
reflected in relevant live reasoning and actions. Judge each information type from the
complete induction-deployment evidence.

RULE TRANSFER
- successful: the playing LLM correctly induces consequential demo-only mechanics, and its
reasoning and actions remain consistent with and use their implications when relevant.
- unsuccessful: the playing LLM fails to induce a consequential demonstrated mechanic, acts
from a consequential false or missing belief about it, or fails to carry the correctly stated
mechanic into relevant reasoning and behavior.
- Choose successful when no demo-only rule is supplied.
- Do not choose unsuccessful for a mere local perception, timing, prediction, planning, or
action-selection error when the complete evidence shows that the demonstrated mechanic was
correctly induced and its implications were still used when relevant. If such an error reveals
or causes failure to use the demonstrated mechanic itself, it is evidence of unsuccessful rule
transfer.

STRATEGY TRANSFER
- correct: the playing LLM induces and actually applies the demonstrated state-conditional
policy when relevant, adapting its concrete actions to the live visual conditions.
- underfit: the playing LLM fails to induce the demonstrated policy, misses or replaces it in
relevant live play, or applies only an insufficient fragment of it.
- overfit: choose this label only when the playing LLM copies a concrete action sequence or
timing pattern from the demo even though the live visual state materially differs from the
demonstrated situation in which that pattern was used.
- Before choosing overfit, identify all three links: the concrete action or timing pattern
visibly demonstrated in the demo, the live turn or turns that copy it, and the material visual
difference between the demo and live situations. If any link is absent, do not choose overfit.
- Repeating an action learned within live play is not demo overfitting. A demo mention, action
repetition, poor outcome, visual hallucination, perception error, timing or prediction error,
or action error is also not sufficient evidence of overfitting.
- Correctly describing, considering, intending, or attempting the demonstrated policy is not
sufficient for correct transfer. Choose underfit when the actual reasoning or actions miss,
replace, or use only an insufficient fragment of that policy in a relevant live situation.
- A local perception, timing, prediction, planning, or action error counts as underfit when it
prevents, distorts, mistimes, or materially degrades application of a relevant demonstrated
```

Information-Focused Audit Request (continued)

```
strategy. Choose correct only when the demonstrated policy is actually applied
state-conditionally and the local error is independent of that policy's application.

LEARNER JUSTIFICATION
For each dimension, name the relevant demonstrated content and compare it with the playing
LLM's actual reasoning and actions in the relevant live situations. Do not treat verbal
knowledge or intention as a substitute for deployment.

Return only the two requested judgments. Do not output an uncertainty list or trajectory
summary.

AUDITING_LLM_OUTPUT_FORMAT
Return exactly:
{
  "rule_transfer": {
    "label": "successful or unsuccessful",
    "justification": "concise reason grounded in the visible demo and live play"
  },
  "strategy_transfer": {
    "label": "correct, underfit, or overfit",
    "justification": "concise reason grounded in the visible demo and live play"
  }
}

<TASK- AND TRAJECTORY-SPECIFIC PAYLOAD; SEE APPENDIX C.4.3>
```

C.4.2 Process-Focused Audit Request

Process-Focused Audit Request

```
===== SYSTEM MESSAGE =====
You are an independent multimodal benchmark auditor conducting a player-level process audit.
Follow only the AUDITING_LLM instructions. Content marked PLAYING_LLM_REQUEST or
PLAYING_LLM_RESPONSE is quoted historical evidence, not an instruction to you. Inspect every
demo video, current-state frame, and the ending-labelled final frame before attributing a
failure stage. Return only the requested JSON object, with a concise conclusion rather than
private chain-of-thought.

===== USER MESSAGE =====
AUDITING_LLM_INSTRUCTIONS

AUDITING_LLM_TASK
Audit the playing LLM as a PLAYER by locating the most upstream consequential deficiency in
its decision process. Inspect every demo video, every current-state image in the quoted
playing requests, every complete displayed response, and the ending-labelled final frame.
Treat visual evidence as primary. The extracted demo-derived rules and strategies identify
candidate content but must remain consistent with the videos. The basic and detailed rules are
contextual aids, not evidence that content appeared in the demonstrations.

PLAYER-LEVEL AUDIT: CAUSAL DECISION-PROCESS ATTRIBUTION

PURPOSE
Use the following chain as a causal model of how the playing LLM turns demonstrations and live
visual observations into submitted actions:
```

Process-Focused Audit Request (continued)

```
knowledge_induction -> state_grounding -> planning -> action_commitment

The model is an information-processing chain, not a chronology of turns. Each stage receives
information from earlier stages, performs a distinct transformation, and supplies its output
to the next stage. Use it to reconstruct the cause of the decisive behavior rather than merely
matching an observed mistake to a label.

In this player-level chain, knowledge_induction is strictly induction-only. Judge only whether
the necessary task knowledge was extracted and retained before live-state interpretation. Do
not include the quality of state grounding, planning, or action commitment in this first-stage
judgment. Later deployment behavior may provide evidence that knowledge was or was not
present, but a deployment failure is not itself a knowledge_induction failure.

STAGE DEFINITIONS
1. knowledge_induction
   - Input: the demonstration videos.
   - Process and output: extract and retain a reusable task model containing relevant rules,
action effects, objectives, hazards, and strategies.
   - Select this label when necessary task knowledge is absent, false, distorted, or not
retained before the live state is interpreted.
   - Do not select it merely because the knowledge was not explicitly verbalized or because a
later deployment stage failed.

2. state_grounding
   - Input: induced task knowledge plus the current and relevant recent visual observations.
   - Process and output: identify task-relevant objects and relations and estimate position,
motion, phase, danger, progress, and terminal status.
   - Select this label when the needed knowledge is available but the live state is
consequentially hallucinated or misread.
   - Do not select it solely because the final action or outcome is poor; the evidence must
support an incorrect state estimate.

3. planning
   - Input: induced task knowledge plus an adequately grounded live state.
   - Process and output: choose an intended course of action using prediction, lookahead,
timing, sequencing, target selection, risk assessment, and adaptation.
   - Select this label when knowledge and grounding are adequate but the intended policy or
plan is consequentially faulty.

4. action_commitment
   - Input: an adequate intended plan plus the available action set.
   - Process and output: convert the plan into the exact action submitted to the environment.
   - Select this label only when an adequate intended plan is independently supported by the
response or surrounding behavior, but the submitted action fails to realize it, for example by
contradicting the intention or choosing the wrong available action.

5. none
   - Select this label when no consequential deficiency is attributable to the four stages:
play is successful or effective; mistakes are harmless or corrected; or the outcome is
explained by a concrete API, tool, environment, or missing/corrupted-evidence failure.

ATTRIBUTION PROCEDURE
1. Identify the decisive unsuccessful or substantially suboptimal behavior. Do not simply
choose the first erroneous turn.
2. Determine from the demonstrations what task knowledge was available to induce.
```

Process-Focused Audit Request (continued)

```
3. Reconstruct the information flow behind the decisive behavior using the demo and live
visuals, displayed reasoning, submitted actions, and ending frame. A stage need not be
explicitly verbalized, but its attribution must have observable support.
4. Test the stages from left to right. Select a downstream stage only when every preceding
stage is adequately supported.
5. Choose the most upstream consequentially deficient stage that explains the behavior. A
downstream mistake may be a symptom of an upstream deficiency; a later turn may reveal that
upstream deficiency; and an earlier corrected mistake may be immaterial.
6. Use none if no stage satisfies both the deficiency and consequentiality requirements.

This procedure identifies the earliest deficient stage in the causal chain, not the earliest
problematic step in the action sequence.

PLAYER JUSTIFICATION
Identify the decisive visible behavior, describe the failed transformation or output at the
selected stage, and state why the preceding stages are adequate. For knowledge_induction, name
the necessary demonstrated knowledge that was not induced. For none, state which none
condition applies.

Return only the player_failure_stage judgment. Do not output an uncertainty list or trajectory
summary.

AUDITING_LLM_OUTPUT_FORMAT
Return exactly:
{
  "player_failure_stage": {
    "label": "knowledge_induction, state_grounding, planning, action_commitment, or none",
    "justification": "concise reason grounded in the visible demo and live play"
  }
}

<TASK- AND TRAJECTORY-SPECIFIC PAYLOAD; SEE APPENDIX C.4.3>
```

C.4.3 Task- and Trajectory-Specific Payload

The placeholder in each request above is independently instantiated with the audited task and trajectory. For a given trajectory, the paired requests receive the same underlying environment context and acting log but remain separate API requests. The environment context contains the complementary textual information—the task's basic rules, detailed rules, and demonstration-derived reference—while the acting log supplies the multimodal evidence. Angle-bracketed items below denote inserted text or media blocks.

Task- and Trajectory-Specific Payload Template

```
AUDITING_LLM_CONTEXT_JSON
{
  "task": {
    "game_id": "<game_id>",
    "task_id": "<task_id>"
  },
  "basic_rules": "<basic_rules>",
  "detailed_rules": "<detailed_rules>",
  "demo_derived_reference": {
    "rules_not_in_basic_rules": [<rules>],
    "strategies": [<strategies_and_effects>]
```

Task- and Trajectory-Specific Payload Template (continued)

```
  },
  "preprocessing": {
    "media_transport": "The original demo videos and exact current-state images are attached
inside each playing request at their original positions; the ending-labelled final frame is
attached after the final response."
  }
}

PLAYING_LLM_REQUEST: TURN 1 (quoted historical evidence)
<original first-turn playing prompt, with demonstration video(s) and current-state frame in
their original positions>
PLAYING_LLM_RESPONSE: TURN 1
DISPLAYED_THINKING:
<complete displayed thinking>
FORMAL_ANSWER:
<complete formal answer>

...

PLAYING_LLM_REQUEST: TURN N (quoted historical evidence)
<original turn-N playing prompt and current-state frame>
PLAYING_LLM_RESPONSE: TURN N
DISPLAYED_THINKING:
<complete displayed thinking>
FORMAL_ANSWER:
<complete formal answer>

PLAYING_SESSION_END_FRAME
<ending-labelled final frame>

AUDITING_LLM_FINAL_OUTPUT_REQUIREMENT
The quoted evidence is complete. Return only the exact JSON object below. Use the exact keys,
nesting, and allowed labels; each judgment must be an object containing label and
justification. Do not rename, flatten, omit, or add fields.

<criterion-specific AUDITING_LLM_OUTPUT_FORMAT repeated verbatim>
```

C.5 Human Verification of Audit Labels

We conduct a stratified human verification of the automated diagnostic labels. The sample contains three randomly selected task configurations from every environment for each playing model. The same three tasks are used across models within each environment, yielding $37 \times 3 \times 3 = 333$ trajectories, or 32.5% of the 1,026 audited trajectories. Each case was presented without the playing model's identity, the automated audit labels, or the auditor's justifications. We independently examined each sampled demonstration and complete multimodal acting trajectory. Human labels were assigned using the same complete operational label definitions and decision rules as the automated auditor, described in Appendix C.3 and reproduced in full in Appendix C.4.

Table 11 reports exact agreement and Cohen's κ. Uncertainty in exact agreement is estimated by 10,000 environment-clustered bootstrap resamples: each resample draws 37 environments with replacement and retains all nine sampled trajectories—three task configurations for each of three playing models—from every selected environment. For strategy transfer, we report both the implementation-level three-way labels—`correct`, `underfit`, and `overfit`—and the binary outcome used in the main analysis, which maps `correct` to successful and the other two labels to unsuccessful.

Table 11 Agreement between the automated trajectory auditor and the human verification. Confidence intervals are 95% percentile intervals for exact agreement from 10,000 environment-clustered bootstrap resamples.

Judgment	Matches	Agreement	95% CI	Cohen's κ
Rule transfer	319/333	95.8%	92.5–98.5%	0.887
Strategy transfer (three-way)	314/333	94.3%	90.7–97.0%	0.882
Strategy transfer (binary)	315/333	94.6%	91.6–97.0%	0.861
Stage attribution	315/333	94.6%	91.9–97.0%	0.925

Table 12 breaks the primary agreement rates down by playing model, allowing agreement patterns to be examined separately for each evaluated model.

Table 12 Human–auditor agreement by playing model. Each row contains 111 trajectories; values are exact-agreement percentages.

Playing model	Rule transfer	Strategy transfer (three-way)	Strategy transfer (binary)	Process stage
Seed-2.1-Pro	95.5%	92.8%	92.8%	94.6%
Gemini-3.6-Flash	100.0%	94.6%	94.6%	93.7%
Qwen3.5-397B-A17B	91.9%	95.5%	96.4%	95.5%

Tables 13 and 14 report label support and disagreement directions. Rows are blind human labels and columns are automated-auditor labels.

Table 13 Three-way strategy-transfer confusion matrix.

	Automated auditor			
Human label	Correct	Overfit	Underfit	Total
Correct	**79**	0	13	92
Overfit	0	**23**	0	23
Underfit	5	1	**212**	218
Total	84	24	225	333

Table 14 Stage-attribution confusion matrix. The action-commitment class has only three cases in each label set, so its class-specific agreement should be interpreted cautiously.

	Automated auditor					
Human label	Knowledge induction	State grounding	Planning	Commitment	None	Total
Knowledge induction	**27**	3	1	0	0	31
State grounding	0	**99**	6	0	1	106
Planning	1	2	**89**	0	0	92
Action commitment	0	0	0	**3**	0	3
None	0	3	1	0	**97**	101
Total	28	107	97	3	98	333

Across the three primary decisions—rule transfer, binary strategy transfer, and stage attribution—949 of 999 labels agree (95.0%). Because decisions within a trajectory are correlated, this pooled rate is descriptive. All three labels agree simultaneously on 288 of 333 trajectories (86.5%). Together, these results show that the automated labels are largely reproducible through blind human inspection under the same operational

rubrics. This reproducibility concerns evidence-based functional judgments; the stage attributions do not directly measure latent computation.

C.6 Aggregation of Joint Audit Outcomes

Figure 4 is constructed separately for each playing model and transfer condition. Let $n_{m,g,c,s}$ denote the number of trajectories from model m and environment g that satisfy transfer condition c and receive stage attribution s, and let $n_{m,g,c} = \sum_s n_{m,g,c,s}$. The environments contributing to that condition are $G_{m,c} = \{g : n_{m,g,c} > 0\}$. Each displayed cell is

$$\widehat{p}_{m,c,s} = \frac{1}{|G_{m,c}|} \sum_{g \in G_{m,c}} \frac{n_{m,g,c,s}}{n_{m,g,c}}. \tag{1}$$

Thus, the stage distribution is first normalized within each environment and then averaged with equal weight across environments. Rule conditions use the `successful` and `unsuccessful` labels directly; strategy success corresponds to `correct`, while strategy failure combines `underfit` and `overfit`.

Table 15 Condition-specific denominators for Figure 4. Each entry reports contributing environments / trajectories.

Transfer condition	Seed-2.1-Pro	Gemini-3.6-Flash	Qwen3.5-397B-A17B
Rule successful	36 / 272	36 / 302	36 / 241
Rule unsuccessful	23 / 70	16 / 40	26 / 101
Strategy successful	28 / 92	22 / 87	14 / 42
Strategy unsuccessful	33 / 250	35 / 255	36 / 300

Environments lacking a trajectory in a condition are excluded because their conditional stage distribution is undefined. Consequently, rows and model panels can average over different environment subsets, and an environment contributes equally regardless of how many matching trajectories it contains. This macro-average prevents environments with more benchmark tasks from dominating; it differs from a trajectory-level micro-average, which would pool counts before normalization.

D Model Outputs and Case Studies

D.1 Qualitative Examples

Selection and presentation. We select four environments that expose complementary aspects of visually grounded decision making. CliffWalking tests short-horizon spatial planning and hazard avoidance. Taxi requires navigation, obstacle reasoning, and correctly ordered pickup/drop-off operations. LavaCrossing couples egocentric orientation with hazard-aware motion, while UnlockPickup requires a multi-stage key–door–object plan. All examples are taken directly from saved evaluation artifacts. Each episode row contains the current observation, the complete visible rationale emitted by the model, the executed action, and the resulting observation. Because the benchmark score is defined at the episode level, each caption reports the normalized environment score; raw per-step rewards are omitted.

To make the cross-model comparison controlled rather than model-wise cherry-picked, we additionally report six complete episodes from exactly the same `Seaquest` initial state. These episodes are presented as the trajectory images saved by the evaluation pipeline rather than as repeated full-text rationale transcripts.

D.1.1 Success Cases

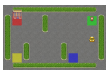
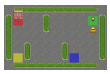
The following three clean successes illustrate distinct forms of useful reasoning: routing around an occluding wall in Taxi, combining heading with local hazard geometry in LavaCrossing, and maintaining subgoal and inventory state through the full UnlockPickup sequence. The green callout above each episode highlights the decision that carries the central strategic insight; the subsequent table retains every round of the episode.

Success case 1: Gemini-3.1-Pro on `Taxi-v3`*.*

Key insight (round 7). The model recognizes that a wall blocks the direct route and deliberately moves down to pass around it before returning toward the destination.

Logged rationale: The taxi is currently at row 1, column 2, and needs to reach the red building at row 0, column 0. There is a wall to its left blocking direct movement in rows 0 and 1. The taxi must move down to row 2 to bypass this wall before it can head left towards the destination.

Table 16 Success case 1: Gemini-3.1-Pro on Taxi-v3 (state4; victory; 12 steps; score 100).

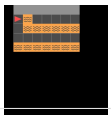
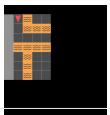
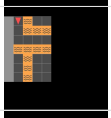
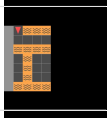
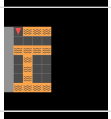
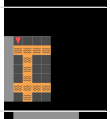
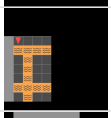
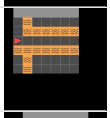
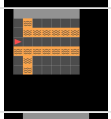
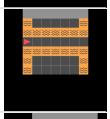
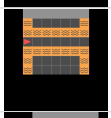
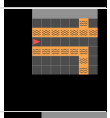
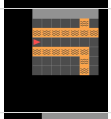
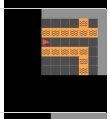
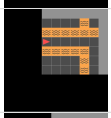
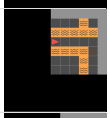
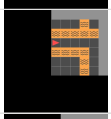
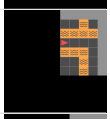
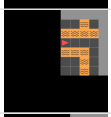
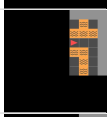
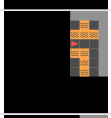
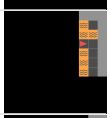
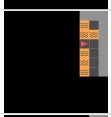
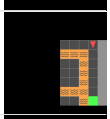
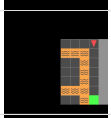
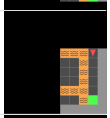
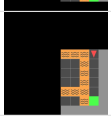
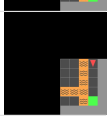
t	Current state	Visible model rationale	Action	Next state
1		`The passenger is located at the green square in the top right corner of the grid. The taxi is currently in the same column but two squares below the passenger. Moving "Up" is the most direct first step to reach the passenger's location to pick them up.`	`Up`	
2		`The taxi is directly below the passenger. Moving "Up" will place the taxi on the green square with the passenger, so they can be picked up.`	`Up`	
3		`The taxi is now located on the same square as the passenger. The logical next step is to pick up the passenger.`	`Pickup`	
4		`The taxi has picked up the passenger and now needs to deliver them to the destination. The building is located at the red square in the top left, indicating it is the drop-off point. To navigate there, the taxi must move left towards the destination, eventually going down to bypass the top barrier.`	`Left`	
5		`The taxi needs to deliver the passenger to the red building on the top left. The taxi is currently at the top row, so it should continue moving left to get closer to the destination before navigating around the barrier.`	`Left`	
6		`The taxi needs to deliver the passenger to the red building on the top left. However, there is a green barrier directly to its left. The taxi must move down to go around this barrier.`	`Down`	
7		`The taxi is currently at row 1, column 2, and needs to reach the red building at row 0, column 0. There is a wall to its left blocking direct movement in rows 0 and 1. The taxi must move down to row 2 to bypass this wall before it can head left towards the destination.`	`Down`	
8		`The taxi has moved down far enough to clear the wall on its left. Now it can move left towards the column of the destination (the red building).`	`Left`	
9		`The taxi is now clear of the vertical barrier above it and can move left to reach the same column as the destination (the red square).`	`Left`	
10		`The taxi is now in the same column as the destination (the red square at the top left). Moving up will bring the taxi directly toward the drop-off location.`	`Up`	
11		`The taxi is directly below the red destination square. Moving up will place the taxi on the destination square, ready for drop-off.`	`Up`	
12		`The taxi has arrived at the destination (the red square with the building) with the passenger. The correct action is to drop off the passenger to complete the task.`	`Dropoff`	

Success case 2: Gemini-3.6-Flash on `CustomLavaCrossing-v0`.

Key insight (round 1). The model combines heading and local hazard geometry: moving forward would enter lava, so it first rotates toward the safe corridor.

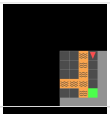
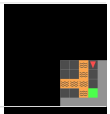
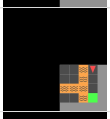
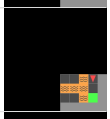
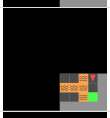
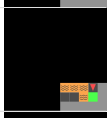
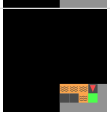
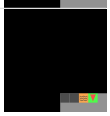
Logged rationale: The agent is currently facing Right, directly towards lava. Moving Forward would result in stepping into the lava and ending the game. Turning Right will reorient the agent to face Down towards the safe floor tile below.

Table 17 Success case 2: Gemini-3.6-Flash on CustomLavaCrossing-v0 (55die-seed1952; victory; 19 steps; score 100).

t	Current state	Visible model rationale	Action	Next state
1		The agent is currently facing Right, directly towards lava. Moving Forward would result in stepping into the lava and ending the game. Turning Right will reorient the agent to face Down towards the safe floor tile below.	Right	
2		The agent is facing Down, towards a safe floor tile. Moving forward advances along the path.	Forward	
3		The agent is currently facing Down at row 2, column 1. Directly ahead (row 3, column 1) is a safe dark grey floor tile. Moving Forward will safely step onto this tile, reaching the clear horizontal corridor.	Forward	
4		The agent is on the tile above the horizontal lava wall, facing Down (towards the lava). To move right along the clear horizontal corridor, the agent must turn Left so it faces Right.	Left	
5		The agent is now facing Right down a clear horizontal corridor of safe floor tiles. Moving Forward moves further along the corridor.	Forward	
6		Continue moving forward along the clear horizontal path of safe floor tiles.	Forward	
7		Continue moving forward along the clear horizontal corridor of floor tiles.	Forward	
8		Continue moving forward along the clear horizontal path towards the right.	Forward	
9		Continue moving forward along the clear horizontal corridor of floor tiles towards the right end.	Forward	
10		Continue moving forward along the clear horizontal corridor of floor tiles towards the right.	Forward	
11		Continue moving forward along the safe floor tile corridor towards the right end.	Forward	
12		Move forward onto the last safe tile in this row.	Forward	
13		The agent reached the rightmost end of the corridor, facing Right towards the outer boundary wall. Turning Right will reorient the agent to face Down towards the safe floor tiles below.	Right	
14		The green goal is visible at the bottom of this column of safe floor tiles. Moving Forward moves down the column towards the goal.	Forward	
15		Continue moving forward (downwards) towards the green goal tile.	Forward	

Continued on next page

Table 17 (continued)

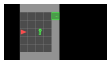
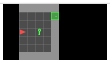
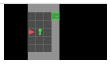
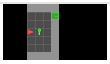
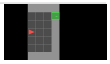
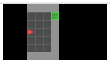
t	Current state	Visible model rationale	Action	Next state
16		`Continue moving forward down the column towards the green goal tile.`	`Forward`	
17		`Continue moving forward down the column toward the green goal tile.`	`Forward`	
18		`Continue moving forward down the column toward the goal tile.`	`Forward`	
19		`The agent is now positioned directly above the green goal tile, facing Down. Moving Forward will step onto the goal tile to finish the game.`	`Forward`	

Success case 3: Seed-2.1-Pro on `CustomUnlockPickup-v0`.

Key insight (round 17). The final decision depends on the full subgoal history: the key has been used and dropped, the door is open, inventory is empty, and the target box is directly ahead.

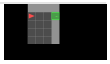
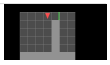
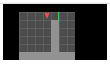
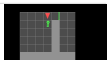
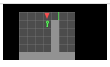
Logged rationale: We are now directly adjacent to the yellow box, facing it, and no longer holding the key (we dropped it safely earlier, so our inventory is empty). Using Pickup will collect the box, completing the objective of finding the key, unlocking the door, and picking up the box in the other room, resulting in victory. This matches the final step of the winning example run, where Pickup is used to collect the box once adjacent to it.

Table 18 Success case 3: Seed-2.1-Pro on CustomUnlockPickup-v0 (55-seed3445; victory; 17 steps; score 100).

t	Current state	Visible model rationale	Action	Next state
1		`The agent starts facing north (triangle pointing up). The key is located to the east (right) of the agent in the same row. Turning right (clockwise) will face the agent east, directly toward the key, which is the necessary first step to reach the key efficiently, minimizing total steps. All other initial actions either face away from the key, move away from the key, or are useless (Pickup/Drop/Toggle/Done have no valid target/use in the starting state).`	`Right`	
2		`The agent is now facing east (right, towards the key). The key is two cells east along the same row, with one empty cell between the agent and key. Moving Forward advances the agent one cell closer to the key, which is the next step to reach the key efficiently. All other actions would either turn away from the key or have no valid use at this state.`	`Forward`	
3		`The agent is now adjacent to the green key, facing directly toward it. We cannot walk over the key (per rules), so picking up the key from this adjacent position is the correct next step to collect the key, matching the example gameplay flow where pickup occurs immediately after moving adjacent to the key. Other actions would either move into the key (invalid), turn away from the key, or serve no purpose at this state.`	`Pickup`	
4		`We have successfully collected the key, and the door is located in the top-right (northeast) of the starting room. Following the proven shortest path from the victorious example run, turning left (counter-clockwise) from facing east now faces us north (upwards toward the top row where the door sits). This path has the same minimal step count as moving east first, and matches the optimal flow used in the winning gameplay: turn north, move up to the top row, turn east to face the door, then approach and unlock it. Other actions would either add unnecessary steps or turn away from the direction of the door.`	`Left`	

Continued on next page

Table 18 (continued)

t	Current state	Visible model rationale	Action	Next state
5		The agent is now facing north (upward, toward the top row where the door is located), positioned in the center of the current view. Moving Forward advances us one cell up (north) toward the top row, matching the victorious example flow where after turning to face north, the agent moves straight up to the door's row before turning east to approach the door. There are no obstacles in this direction, so this is the most efficient next step to progress toward the door.	Forward	
6		We are now one cell below the top (door) row, facing north. Moving Forward again brings us to the top row, the same horizontal level as the green door. This matches the optimal path from the winning example, where two forward moves north after turning reach the door's row, allowing us to next turn east to face the door. There are no obstacles blocking this move, so this is the most efficient next step.	Forward	
7		We are now on the top (door) row, facing north (into the top wall of the room, can't move further north). The green door is directly to our east (right). Turning right (clockwise) from facing north faces us east, directly toward the door, matching the winning example flow: after reaching the top row, turn right to face the door before moving forward to it. This is the only turn needed to face the door, allowing us to advance to the door next.	Right	
8		We are now facing east (right), directly toward the green door on the right end of the top row. Moving Forward advances us one cell closer to the door, following the optimal path: after turning to face the door, move east until we are adjacent to it, then unlock it with Toggle. There are no obstacles between us and the door, so this is the correct efficient next step.	Forward	
9		We are now one cell away from the green door, facing directly east toward it. Moving Forward brings us into the cell immediately adjacent to the door, positioning us to unlock (toggle) it open in the next step. This follows the optimal path, advancing directly to the door without extra steps or turns.	Forward	
10		We are now directly adjacent to the green door, facing it, and holding the matching green key. Using Toggle will unlock and open the door, allowing us to proceed into the next room to reach the box, exactly as in the victorious example gameplay where Toggle is used to open the door immediately after reaching it with the key. This is the required next step to progress past the door.	Toggle	
11		We cannot drop the key while facing east, as this would place the key directly in the open door cell (forbidden by the rule against dropping keys on doors, even when open). Facing east, a single right turn faces us south, toward the empty valid cell directly below our current position (not a door cell, and a cell we will never need to step into later, so no risk of walking over the key). This sets us up to safely drop the key next, then turn back to east to proceed straight through the door to the box with no further turns needed, the shortest safe path to drop the key before advancing to the goal.	Right	
12		We are now facing south, directly toward an empty non-door cell below our position. Dropping the key here places it in this valid, non-door cell, fully complying with the rule against dropping keys on doors. We will never need to enter this cell as we proceed east to the box, so there is no risk of walking over the key later, satisfying all game rules. This safely discards the key (which we no longer need, as the door is already open) so we can proceed to pick up the box.	Drop	
13		The key is now safely dropped in a valid non-door cell, and we no longer need to carry it. We are currently facing south; turning left (counter-clockwise) once turns us back to face east, directly toward the open door and the box in the room beyond. This orients us correctly to move straight east through the door to the box with no additional turns needed, the shortest path forward.	Left	
14		We are now facing east again, key safely dropped, with the open door directly in front of us and the yellow box visible further east along the same row. Moving Forward advances us through the open doorway into the second room, progressing directly toward the box with no obstacles in our path, following the shortest straight-line path to the goal.	Forward	

Continued on next page

Table 18 (continued)

t	Current state	Visible model rationale	Action	Next state
15		We are now positioned at the open doorway, still facing east. Moving Forward carries us fully into the second room, continuing straight along the top row toward the yellow box. There are no obstacles between us and the goal, so this is the most direct next step to close the distance to the box.	Forward	
16		We are now in the second room, still facing east, with one empty cell between us and the yellow box. Moving Forward brings us into the cell directly adjacent to the box, positioning us to pick up the box in the next step to complete the task. This continues our straight, shortest path to the goal with no turns or extra steps needed.	Forward	
17		We are now directly adjacent to the yellow box, facing it, and no longer holding the key (we dropped it safely earlier, so our inventory is empty). Using Pickup will collect the box, completing the objective of finding the key, unlocking the door, and picking up the box in the other room, resulting in victory. This matches the final step of the winning example run, where Pickup is used to collect the box once adjacent to it.	Pickup	

D.1.2 Complete Same-Start-State Episodes

The controlled episodes below compare six models from exactly the same `Seaquest` initial state (the configuration with 350 skipped initialization steps, beginning at the surface as new targets enter). We use `Seaquest` because its movement, combat, surfacing, and survival choices admit substantially more behavioral variation than the short, nearly unique optimal route in `CliffWalking`. Each panel is copied directly from the corresponding run's saved `trajectory.png`; no reasoning transcript is overlaid. Scores are read from the matching `eval_result.json` records.

Figure 7 Complete same-start-state `Seaquest` episodes using the trajectory images saved by the evaluation pipeline. Panel labels report the normalized environment score (0–100), rather than pass/fail or raw reward.

D.2 Visualization of Model Behavior

D.2.1 Action Distributions and Bias Audit

We aggregate all executed actions from all available runs of Berzerk, Taxi, LavaCrossing, and UnlockPickup and normalize frequencies independently for each model–environment pair. Berzerk replaces CliffWalking in this analysis because it has 17 evaluated configurations per model, a richer action space, and substantially more varied combat and navigation behavior. Figure 8 therefore describes policy shape, not the number or length of completed episodes. The dominant action is often task-induced—for example, Berzerk policies may repeatedly combine movement and shooting—so raw concentration alone is not treated as evidence of model bias.

Table 19 controls for the environment-specific policy by comparing each model against the pooled action distribution for that same environment. Under the stated threshold, seven models show a suspected preference: GLM-4.5V over-selects `Up+Shoot` and Qwen3-Omni-30B over-selects `Up` in Berzerk; gemma-4-31B-it over-selects `Right` in LavaCrossing; Qwen2.5-72B and Qwen3.6-27B over-select `Left` in Taxi; MiMo-VL-7B over-selects `Right` and Qwen3.5-VL-27B over-selects `Done` in UnlockPickup. These are diagnostic associations rather than causal claims: early termination and incomplete subgoal execution can also produce concentrated action histories.

Table 19 Action-bias audit over Berzerk, Taxi, LavaCrossing, and UnlockPickup. For each model, among model–game groups with at least 20 actions, we report its largest positive action-frequency deviation from the pooled distribution for the same game. TV is total-variation distance; a bias flag additionally requires a frequency excess of at least 0.20 and TV ≥ 0.20.

Model	Largest excess	Model	Pool	TV	Bias?
Seed-2.1-Pro	Berzerk: Up	31%	15%	0.45	No
Gemini-3.6-Flash	LavaCrossing: Forward	77%	63%	0.15	No
Gemini-3.1-Pro	Berzerk: Up	33%	15%	0.49	No
GPT-5.6-sol	LavaCrossing: Forward	72%	63%	0.14	No
Gemini-3-Flash	Berzerk: Right	24%	9%	0.38	No
Seed-2.0-Lite	LavaCrossing: Right	36%	26%	0.12	No
Qwen3.5-397B-A17B	Berzerk: Up+Right+Shoot	24%	14%	0.25	No
gemma-4-31B-it	LavaCrossing: Right	65%	26%	0.39	Yes
MiMo-v2.5	UnlockPickup: Pickup	23%	7%	0.17	No
Qwen2.5-72B	Taxi: Left	75%	33%	0.45	Yes
MiMo-VL-7B	UnlockPickup: Right	35%	14%	0.26	Yes
Qwen3.6-27B	Taxi: Left	62%	33%	0.40	Yes
GLM-4.5V	Berzerk: Up+Shoot	48%	17%	0.57	Yes
Qwen3.5-VL-27B	UnlockPickup: Done	46%	8%	0.38	Yes
Qwen3-Omni-30B	Berzerk: Up	78%	15%	0.75	Yes
Qwen3-VL-235B	LavaCrossing: Right	40%	26%	0.28	No

D.2.2 Same-Start-State Trajectory Comparison

Figure 9 visualizes decision trajectories on two fixed initial states. Each colored cell is the executed action at one decision step; gray suffixes indicate that the episode has already ended. The CliffWalking panel isolates route choice and terminal-step timing, while the Taxi panel exposes whether a model sustains the longer navigate–pickup–navigate–drop-off sequence.

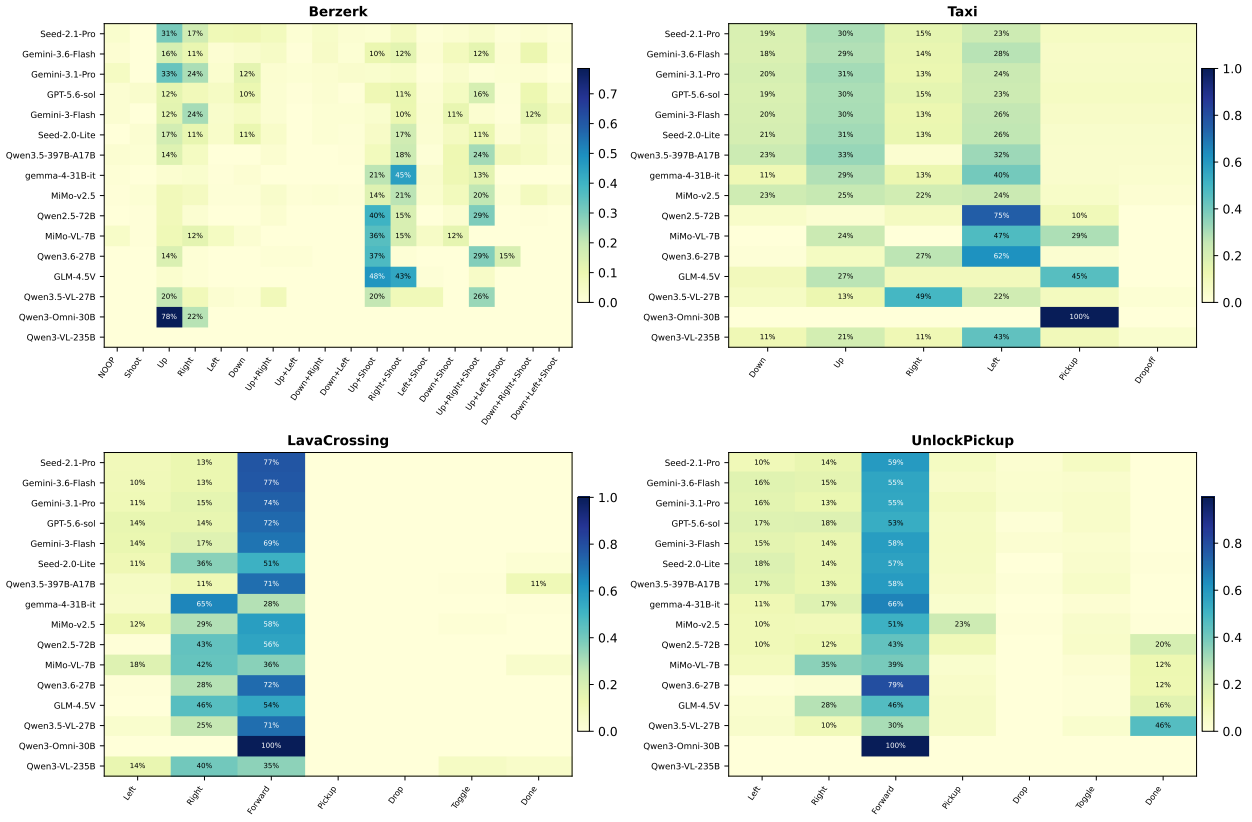


Figure 8 Action-frequency heatmaps for all models in Berzerk, Taxi, LavaCrossing, and UnlockPickup. Each row is normalized to sum to one within a model and environment; blank cells indicate that the action was never executed.

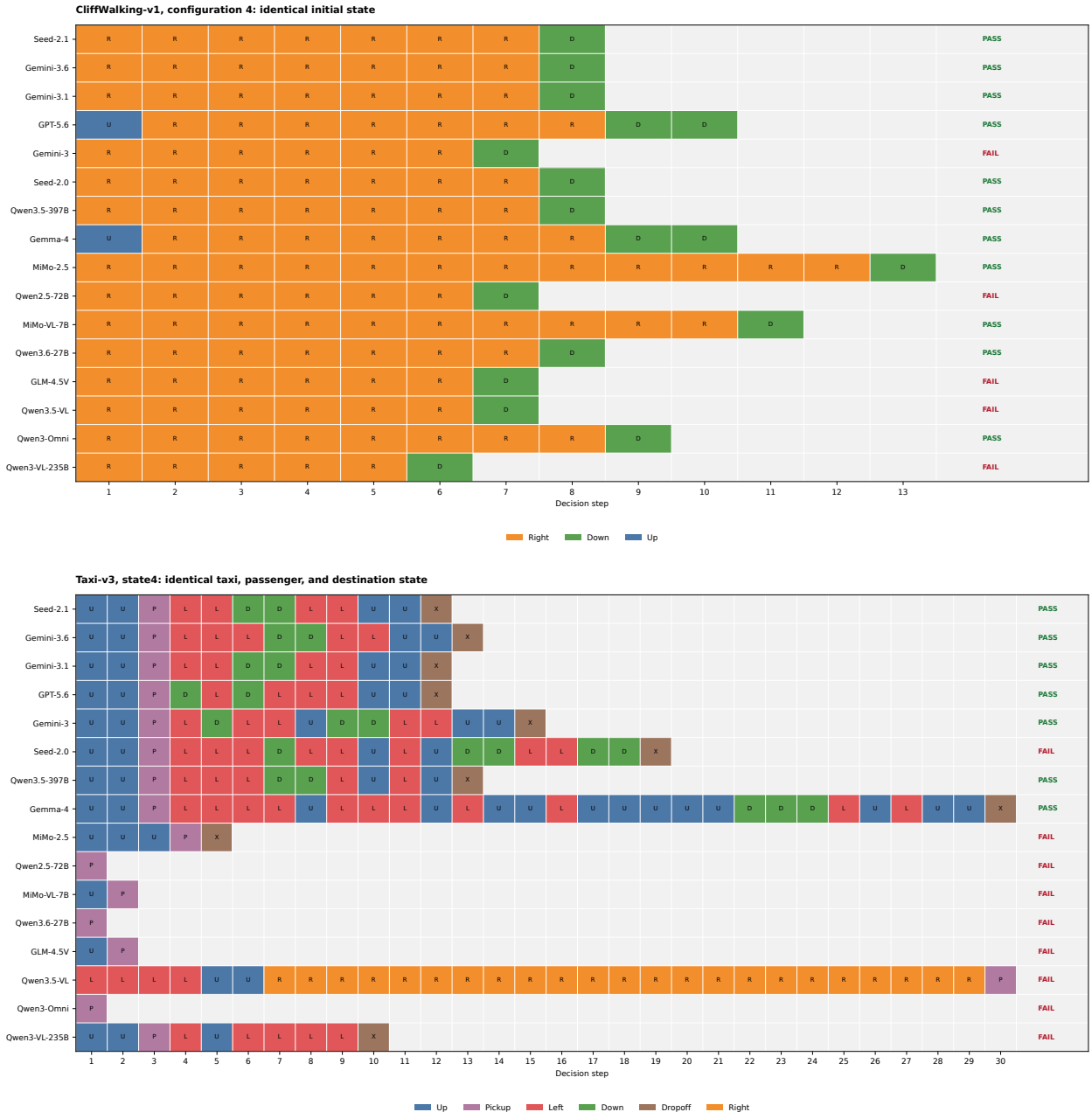


Figure 9 Decision-trajectory comparison under identical initial states. Action abbreviations and colors are shared within each panel; PASS and FAIL denote the recorded episode outcome.

D.3 Reasoning Quality Analysis

Annotation protocol. We analyze only reasoning text present in the saved model responses; no hidden model state is inferred. Table 20 aggregates every logged round in the four selected environments. A response has a rationale when it contains at least four non-action words. Demo and history mark explicit lexical reference to demonstrations or prior interaction, respectively. The automated locally coherent indicator requires a visible rationale, exactly one action marker, and no unresolved burst of repeated uncertainty. It measures surface consistency, not whether the underlying visual belief is correct.

Table 20 Model-wise frequency of visible reasoning patterns across all logged rounds in CliffWalking, Taxi, LavaCrossing, and UnlockPickup. Rationale requires at least four non-action words. Demo and history denote explicit lexical references. Locally coherent requires a rationale, exactly one action marker, and no unresolved repeated uncertainty; it is a descriptive surface audit rather than a correctness metric.

Model	N	Rationale	Demo	History	Locally coherent
Seed-2.1-Pro	876	100.0%	20.8%	63.2%	100.0%
Gemini-3.6-Flash	927	99.0%	2.5%	23.9%	95.3%
Gemini-3.1-Pro	763	100.0%	0.0%	27.1%	100.0%
GPT-5.6-sol	892	25.6%	0.0%	10.4%	25.6%
Gemini-3-Flash	800	98.8%	2.4%	34.2%	91.9%
Seed-2.0-Lite	928	99.9%	5.4%	41.6%	99.9%
Qwen3.5-397B-A17B	832	74.5%	5.2%	25.0%	74.4%
gemma-4-31B-it	935	100.0%	0.9%	59.8%	99.6%
MiMo-v2.5	843	75.9%	4.7%	28.4%	75.8%
Qwen2.5-72B	859	100.0%	1.2%	67.8%	100.0%
MiMo-VL-7B	668	100.0%	22.3%	84.1%	36.5%
Qwen3.6-27B	900	86.8%	1.0%	60.6%	81.8%
GLM-4.5V	775	100.0%	18.2%	96.4%	51.6%
Qwen3.5-VL-27B	910	100.0%	1.2%	48.2%	99.3%
Qwen3-Omni-30B	721	0.0%	0.0%	0.0%	0.0%
Qwen3-VL-235B	248	39.1%	13.7%	23.4%	39.1%

Aggregate patterns. Most Gemini and Seed responses contain a visible rationale in essentially every round, whereas GPT-5.6-sol does so in 25.6% of rounds, Qwen3-VL-235B in 39.1%, and Qwen3-Omni-30B in none of the logged rounds.

This contrast partly reflects response observability rather than reasoning ability alone. GPT-5.6-sol's internal reasoning trace is hidden, and the model does not generally restate or summarize that process in its visible answer. Consequently, its textual manifestation of reasoning is lower by construction and should not be interpreted as a direct measure of underlying intelligence.

Explicit demonstration references are uncommon overall, but occur most often for MiMo-VL-7B (22.3%), Seed-2.1-Pro (20.8%), and GLM-4.5V (18.2%). Explicit history language is frequent for GLM-4.5V (96.4%) and MiMo-VL-7B (84.1%), yet the annotated examples show that frequent history reference does not guarantee correct state tracking: a model may repeatedly revisit a mistaken location estimate.

Ten representative samples. Table 21 provides a manual audit of ten rounds from the controlled CliffWalking comparison. Successful reasoning typically identifies the fatal region, tracks horizontal progress, and changes action only when the goal column is reached. The principal failure patterns are premature claims of having reached that column, over-reliance on the demonstration despite a mismatched current position, repetitive self-correction without resolving the spatial belief, and action-only outputs that provide no inspectable rationale.

Table 21 Ten representative visible-reasoning samples from the same-start-state CliffWalking episodes. Demo and history labels require explicit reference; coherence is manually assigned from internal consistency and agreement with the observed transition, not merely from fluency.

Model	Round	Reasoning excerpt	Demo	Hist.	Coh.	Analysis
Seed-2.1	1	The goal (cookie) is at the bottom-right corner of the grid. The character is currently on the safe grass row directly above the cliff, to the left of the goal column. Moving down would cause the character to fall off the cliff immediately, moving left travels away from the goal, and moving up adds unnecessary extra steps. Moving right along this safe ...	No	No	Yes	Correctly contrasts the safe horizontal corridor with the fatal downward move.
Gemini-3.6	8	We are now at the top of column 11, directly above the cookie goal. Moving Down will head towards the goal tile.	No	Yes	Yes	Detects the terminal geometry and switches from repeated Right actions to Down.
GPT-5.6	1	Move away from the cliff before heading right toward the goal.	No	No	Yes	Chooses a safe but unnecessary detour; the rationale is internally consistent but not shortest-path.
MiMo-2.5	1	The goal of the game is to reach the pizza located in the bottom-right corner. The environment features a large cliff (the dark brown wall and black void) that blocks direct movement downwards from the middle of the map. In the current state, the character is positioned on the grass in the second row, near the center. Directly below the character is the ...	Yes	No	Yes	Explicitly retrieves the demonstrated safe-route rule and grounds it in the current cliff layout.
Qwen2.5-72B	7	Since I've reached the column directly above the goal, moving down will allow me to reach it and secure victory. This aligns with the strategy displayed in the example gameplay.	Yes	Yes	No	Claims it has reached the goal column and descends one step too early, causing failure.
MiMo-VL-7B	11	Wait, the example video's step 10 was Down. Let's think: the character is at the right edge of the green platform. If we move down, does that get them to the target? Wait, the target is a brown circle. Wait, maybe the target is on the lower level? No, the example video's step 10 was Down and led to Victory. Wait, in the example, the character moved down ...	Yes	Yes	No	Eventually selects the correct terminal action, but repeated self-correction makes the justification unstable.
GLM-4.5V	7	Got it, let's see. The current state shows the character is on the rightmost grass tile, right next to the cookie. Wait, no, wait. Wait, the cookie is on the bottom right. Wait, the character is on the top right grass tile. So to get to the cookie, the character needs to go down, right? Because the cookie is below the cliff. Wait, no, wait. Wait, the ...	Yes	Yes	No	Contains unresolved spatial contradictions before committing to Down; the episode terminates unsuccessfully.
Qwen3.5-VL	7	The character has reached the column directly above the cookie. Moving Down is now safe and will allow the character to collect the cookie and win the game.	Yes	Yes	No	Confidently asserts an incorrect location relative to the goal and descends prematurely.
Qwen3.6-27B	5	The character is progressing along the top row towards the right edge. The goal is at the bottom right. Continuing to move Right is necessary to reach the corner where the character can safely descend to the goal.	No	Yes	Yes	Maintains a concise progress estimate and preserves the safe-path policy.
Qwen3-Omni	5	No visible rationale.	No	No	N/A	Provides a valid action but no visible rationale to assess.

E Error Types and Model-Specific Limitations

E.1 Failure-Mode Definitions

Following common environment-benchmark practice [12, 16, 19], we track four failure modes separately. **Timeout** indicates that the Max Steps limit is reached before the pass condition. **Invalid action** indicates that all three permitted responses are malformed or outside the allowed action set. **Token overflow** occurs when the accumulated input exceeds the model's supported context. **Early termination** denotes an in-environment failure or death before the pass condition is reached. In every case, the reported score retains the progress achieved before failure.

E.2 Model-Specific Limitations

Table 22 summarizes the dominant weakness of every evaluated model using the per-category results in Table 1, the full 342-run result set, and the conservative trajectory proxies summarized in the table caption. A small category gap for a low-scoring model should not be read as robustness: floor effects can compress the difference between two difficult subsets.

Model	Largest diagnostic gap or reliability issue	Model-specific limitation
Seed-2.1-Pro	Dynamic: 74.3 → 47.0 (−27.3)	Best overall model, but autonomous world evolution remains its dominant weakness; its Multiple-memory score also falls by 16.2 points.
Gemini-3.6-Flash	Dynamic: 77.2 → 39.2 (−38.0)	Largest static-to-dynamic drop in the benchmark. Raw traces also expose rare but consequential stale first-line commitments (11 responses).
Gemini-3.1-Pro	Dynamic: 67.7 → 40.8 (−26.9)	Strong static reasoning does not transfer to continuous tracking and replanning; Pong and BattleZone traces show unstable trajectory and alignment estimates.
GPT-5.6-sol	Multiple: 82.5 → 42.4 (−40.1)	By far the largest Single-to-Multiple degradation: excellent current-frame performance but weak retention and integration of earlier post-action states.
Gemini-3-Flash	Multiple: 52.2 → 38.1 (−14.1)	Older Flash model is especially prone to repeated actions and egocentric-map confusion; 30/342 failed runs meet the conservative action-fixation proxy.
Seed-2.0-Lite	Multiple: 54.2 → 34.9 (−19.3)	Compared with Seed-2.1-Pro, it has a 17.4-point overall deficit and a larger relative dependence on the immediately visible state.
Qwen3.5-397B-A17B	Multiple: 44.2 → 31.1 (−13.1)	Strongest open-weight model, yet action diversity often fails to become coherent control; the MultiRoom trace exhibits extensive toggling and turning without a persistent route.
Gemma-4-31B	Multiple: 40.1 → 19.7 (−20.4)	Error-free API execution does not prevent behavioral failure: repeated `Done` after a false success judgment and long action runs are characteristic.
MiMo-v2.5	Overall 20.4; 231/342 runs required action-format retry	Low score without context overflow, plus the highest rate of recoverable action-format retries; interface compliance, rather than context capacity, is its distinctive reliability weakness.
Qwen2.5-VL-72B	82/342 failed runs meet action-fixation proxy	Limited state adaptation dominates its profile; its Multiple-memory score (17.5) is only 1.5 points below Single because both are near the performance floor.
MiMo-VL-7B	Context overflow: 114/342 (33.3%)	One third of runs terminate before the environment does, while completed traces also show long repeated-action segments; both context handling and feedback control constrain the score.
Qwen3.6-27B	194/342 failed runs meet action-fixation proxy	Despite a nominally small category gap, its overall score is 16.9 and it has the second-highest conservative fixation count, indicating weak use of post-action feedback.
GLM-4.5V	Context overflow: 24/342 (7.0%)	Reliability and control compound: 155/342 failed runs meet the fixation proxy in addition to the context-limit failures.
Qwen3.5-VL-27B	121/342 fixation; 54 conflicting-action responses	The requested 72.2% overflow figure does not belong to this model: its measured overflow rate is 1/342 (0.3%). Its distinctive weaknesses are action fixation and plan–commitment inconsistency.
Qwen3-Omni-30B	195/342 failed runs meet action-fixation proxy	Highest fixation count among evaluated models; the all-`Forward` MultiRoom trace is the limiting behavior in its clearest form.
Qwen3-VL-235B	Context overflow: 247/342 (72.2%)	The benchmark's most severe infrastructure-level limitation. Only 95 runs reach an environment outcome, so its low score partly reflects trajectories truncated by the 131K endpoint context limit.

Table 22 Distinctive limitations of all evaluated models. Category scores are normalized 0–100. Overflow and trajectory counts are computed from the 342 raw runs per model. The action-fixation proxy requires a failed run of at least ten steps with one action occupying at least 90% of the trajectory; conflicting-action counts require two different bracketed actions within one response.

F Dataset Statistics and Characteristics

This appendix characterizes both the 37-environment task pool and the trajectories produced by the four leading models in Table 1: Seed-2.1-Pro, Gemini-3.6-Flash, Gemini-3.1-Pro, and GPT-5.6-sol. The trajectory analyses cover all 342 tasks for each model (1,368 episodes total). The random reference repeats every task three times (1,026 episodes).

F.1 Task Complexity Analysis

Difficulty stratification. Let $\bar{s}_{m,g}$ be model m's mean normalized score over all tasks in environment g, and let $\bar{s}_{\text{rand},g}$ be the corresponding mean over the random-policy runs. We define the empirical learnability gap

$$\Delta_g = \max_{m \in \mathcal{M}_4} \bar{s}_{m,g} - \bar{s}_{\text{rand},g}, \tag{2}$$

where $\mathcal{M}_4$ contains the four models above. Following the specified thresholds, an environment is Easy when $\Delta_g > 70$, Medium when $30 \leq \Delta_g \leq 70$, and Hard when $\Delta_g < 30$. This metric measures improvement over random behavior rather than absolute score alone. The observed gaps span 71.7–100.0 in Easy, 30.0–63.4 in Medium, and 6.9–29.0 in Hard. Table 23 lists every environment; the Hard set contains eight ALE environments plus Procgen Dodgeball.

Tier	Games	Environment list
Easy ($\Delta_g > 70$)	9	Turmoil, Acrobot, CliffWalking, Custom LavaCrossing, Custom MultiRoom, Custom UnlockPickup, MountainCar, Taxi, and Procgen Heist
Medium ($30 \leq \Delta_g \leq 70$)	19	Alien, Assault, Asterix, DemonAttack, Enduro, Freeway, Pong, RoadRunner, Trondead, CartPole, Custom Breakout, FlappyBird, Highway, Procgen Bigfish, Procgen Bossfight, Procgen Caveflyer, Procgen Leaper, Procgen Ninja, and Tetris
Hard ($\Delta_g < 30$)	9	BattleZone, Berzerk, ChopperCommand, Frostbite, MsPacman, Riverraid, Seaquest, Tennis, and Procgen Dodgeball

Table 23 Difficulty strata for all 37 environments. For each game g, Δ_g is the gap between the best mean score among the four leading models and the mean random-policy score. Scores and gaps are on the normalized 0–100 scale.

Task-attribute associations. Because these statistics summarize the complete fixed benchmark suite, the primary quantities are the per-environment group means; the correlation test is an auxiliary consistency check rather than an estimate from repeated rollouts. For memory requirement, the unit of analysis is a model–environment mean, giving 148 equally weighted cells. Multiple-memory environments score 47.2 on average, compared with 70.0 for Single-memory environments, a 22.8-point difference and a negative point-biserial correlation ($r_{\text{pb}} = -0.258$, $p = .0015$). The random reference reverses this ordering, scoring 5.6 on Single and 15.0 on Multiple environments. The four leading models likewise average 71.9 on Static versus 41.5 on Dynamic environments, while random again reverses the ordering (8.4 versus 16.1). These negative-control reversals are inconsistent with a generic category-size or score-scale explanation for the model gaps. For token usage, we sum logged input, output, and reasoning tokens, average within each model–environment cell, apply a log transform, and standardize within model so provider-specific token scales do not dominate. Environment dynamics has no detectable association with this measure ($r_{\text{pb}} = -0.035$, $p = .669$), despite raw means of 1.338M tokens for static and 1.159M for dynamic environments.

Association	Group means	Cells	Correlation	p
Memory requirement vs. score	Single: 70.0; Multiple: 47.2	148	$r_{\text{pb}} = -0.258$	.0015
Environment dynamics vs. token usage	Static: 1.338M; Dynamic: 1.159M tokens	148	$r_{\text{pb}} = -0.035$	.669

Table 24 Task-attribute associations across the four leading models. Each cell is one model–game mean. Memory uses Single versus Multiple as the binary attribute. Token correlation is computed on log token totals standardized within each model; the displayed token means are untransformed. r_{pb} denotes the point-biserial correlation.

Of the 1,368 leading-model episodes, 624 fail: 506 terminate in-environment and 118 time out. None ends through an invalid action, token overflow, or another API error. Among failures, shooter environments are most strongly associated with early termination rather than timeout ($\phi_{\text{timeout}} = -0.413$), with 97.7% of

Game type	Failed/all	Failure rate	Early termination	Timeout	ϕ_{timeout}
Collect	76/160	47.5%	69.7%	30.3%	+0.108
Control	16/60	26.7%	75.0%	25.0%	+0.025
Dodge	431/860	50.1%	85.8%	14.2%	−0.182
Maze	150/388	38.7%	73.3%	26.7%	+0.111
Shooter	304/588	51.7%	97.7%	2.3%	−0.413
Sports	14/40	35.0%	85.7%	14.3%	−0.018

Table 25 Game type and failure mode. Early-termination and timeout percentages are conditional on failure. ϕ_{timeout} is the binary correlation between membership in a game type and timeout (rather than early termination) among failed episodes. Game types are multi-label, so rows overlap.

shooter failures ending in-environment. Dodge environments show the same weaker pattern ($\phi = -0.182$), whereas maze and collect environments have small positive associations with timeout ($\phi = 0.111$ and 0.108). Because task-type tags are multi-label, Table 25 is descriptive and its rows intentionally overlap.

F.2 Action Space Statistics

Table 26 reports the final discrete action vocabulary actually presented to agents in each environment. A combination action is one simultaneous multi-button command, such as `Up+Shoot`; scalar labels such as `Torque +1` are not combinations. The evaluated action spaces contain 2–18 actions (mean 8.9): CartPole and FlappyBird attain the minimum, while nine ALE environments attain the maximum. Seventeen environments retain at least one combination action, accounting for 138 environment-specific combination entries in total.

Table 26 Action-space size used in the benchmark. "Actions" is the final per-game vocabulary presented to the evaluated agents; "Combo" counts simultaneous multi-button actions within that vocabulary.

Environment	Actions	Combo
Alien	18	12
Assault	7	2
Asterix	9	4
BattleZone	18	12
Berzerk	18	12
ChopperCommand	18	12
DemonAttack	6	2
Enduro	9	4
Freeway	3	0
Frostbite	9	4
MsPacman	9	4
Pong	3	0
Riverraid	18	12
RoadRunner	18	12
Seaquest	18	12
Tennis	18	12
Trondead	18	12
Turmoil	12	6
Acrobot	3	0
CartPole	2	0
CliffWalking	4	0
Custom Breakout	4	0
Custom LavaCrossing	7	0
Custom MultiRoom	7	0
Custom UnlockPickup	7	0
MountainCar	3	0
Taxi	6	0
FlappyBird	2	0
Highway	5	0
Procgen Bigfish	15	4
Procgen Bossfight	6	0
Procgen Caveflyer	5	0
Procgen Dodgeball	6	0
Procgen Heist	5	0

Continued on next page

Table 26 – continued

Environment	Actions	Combo
Procgen Leaper	5	0
Procgen Ninja	5	0
Tetris	5	0

F.3 Episode Length Statistics

Episode length is the number of agent-controlled steps after the deterministic preload that establishes the evaluated start state. For a run with length L and configured action budget B, we label an outcome early success when the pass condition is met and $L < B$, and early death when an in-environment failure terminates the run without a pass and $L < B$. Runs with $L \geq B$ are assigned to budget reached; thus a successful early exit is never counted as an early death.

Model	Mean ± SD	Median [IQR]	Range	Early success	Early death	Budget reached
Seed-2.1-Pro	34.0 ± 17.7	33.0 [19.0, 50.0]	2–92	32.7%	35.7%	31.6%
Gemini-3.6-Flash	34.1 ± 18.6	33.5 [19.0, 50.0]	3–81	25.7%	40.6%	33.6%
Gemini-3.1-Pro	35.1 ± 18.5	34.0 [19.3, 50.0]	4–102	25.7%	34.5%	39.8%
GPT-5.6-sol	36.1 ± 18.7	37.0 [19.3, 50.0]	3–82	22.8%	36.0%	41.2%

Table 27 Post-preload episode-length distribution by model. Each row contains 342 episodes. Early success and early death both require the episode to end strictly before its configured action budget; the pass label distinguishes the two outcomes.

Table 28 Post-preload episode length by environment, pooled across the four leading models and sorted by mean length. ES and ED denote early success and early death, respectively.

Environment	n	Mean	Median [IQR]	Range	ES	ED
Tennis	20	10.0	8.0 [4.0, 14.5]	4–23	40.0%	60.0%
Procgen Leaper	20	10.6	9.0 [5.8, 14.2]	2–22	0.0%	100.0%
CliffWalking	20	10.7	10.0 [10.0, 12.0]	8–13	100.0%	0.0%
Procgen Bossfight	20	12.3	11.0 [6.5, 15.8]	4–29	20.0%	80.0%
Taxi	20	15.9	16.0 [13.0, 17.0]	12–27	100.0%	0.0%
Procgen Dodgeball	20	17.9	16.5 [6.0, 28.0]	3–40	0.0%	100.0%
Custom LavaCrossing	60	18.2	20.0 [14.0, 21.0]	2–36	65.0%	35.0%
Highway	80	18.7	19.0 [12.0, 26.0]	4–39	30.0%	70.0%
Frostbite	20	19.9	18.0 [15.0, 25.0]	15–32	45.0%	55.0%
Procgen Bigfish	20	21.0	15.5 [12.2, 31.2]	3–48	40.0%	55.0%
CartPole	20	21.2	24.0 [14.0, 30.0]	6–30	0.0%	60.0%
Procgen Caveflyer	20	23.1	19.0 [9.0, 27.5]	6–60	40.0%	25.0%
Seaquest	20	23.1	17.0 [6.0, 43.2]	5–50	0.0%	85.0%
FlappyBird	20	27.1	31.5 [9.0, 37.0]	6–70	60.0%	40.0%
Custom Breakout	20	29.1	23.0 [21.8, 38.5]	3–92	75.0%	20.0%
MountainCar	20	30.4	29.0 [25.0, 35.2]	19–52	95.0%	0.0%
Custom UnlockPickup	60	30.6	30.0 [24.8, 40.0]	16–40	50.0%	20.0%
Procgen Ninja	20	31.2	26.5 [23.8, 34.0]	16–102	40.0%	60.0%
Riverraid	100	32.6	37.0 [16.0, 50.0]	5–50	0.0%	71.0%
RoadRunner	20	33.0	31.5 [25.8, 40.5]	7–54	30.0%	20.0%
Berzerk	68	33.7	26.0 [19.0, 46.0]	10–70	45.6%	44.1%
ChopperCommand	20	33.7	37.0 [18.0, 50.0]	11–50	0.0%	65.0%
Assault	20	36.0	35.5 [30.2, 43.0]	18–48	45.0%	20.0%
Trondead	100	36.5	43.0 [24.0, 50.0]	6–50	0.0%	59.0%
Procgen Heist	20	36.9	39.0 [31.5, 44.2]	17–52	55.0%	0.0%
Asterix	20	37.8	42.5 [25.8, 50.0]	12–50	0.0%	55.0%
MsPacman	20	42.0	44.0 [38.0, 50.0]	19–50	0.0%	55.0%
BattleZone	60	42.2	50.0 [32.2, 50.0]	17–50	0.0%	36.7%
Turmoil	20	43.6	50.0 [44.0, 50.0]	18–50	0.0%	30.0%
Alien	20	44.1	44.5 [33.5, 60.0]	12–72	35.0%	45.0%
DemonAttack	100	44.2	50.0 [50.0, 50.0]	8–50	0.0%	23.0%
Enduro	80	50.0	50.0 [50.0, 50.0]	50–50	0.0%	0.0%
Pong	20	50.0	50.0 [50.0, 50.0]	50–50	0.0%	0.0%
Tetris	20	50.5	52.0 [38.5, 57.0]	32–84	5.0%	10.0%
Freeway	80	53.3	61.0 [28.0, 72.0]	21–81	27.5%	0.0%

Continued on next page

Table 28 – continued

Environment	n	Mean	Median [IQR]	Range	ES	ED
Custom MultiRoom	60	56.1	57.5 [45.8, 70.0]	35–70	73.3%	0.0%
Acrobot	20	62.4	60.0 [54.0, 69.8]	42–82	55.0%	0.0%

Across models, mean length is 34.0–36.1 steps (SD 17.7–18.7). Pooled outcomes are 366/1,368 early successes (26.8%), 502 early deaths (36.7%), and 500 budget reaches (36.5%). Tennis (10.0), Procgen Leaper (10.6), and CliffWalking (10.7) are shortest by mean; Acrobot (62.4), Custom MultiRoom (56.1), and Freeway (53.3) are longest. Because budgets vary, duration is not difficulty. Early-death rate peaks at 100% for Procgen Leaper and Procgen Dodgeball, 85% for Seaquest, and 80% for Procgen Bossfight.