S^3GYM: CAN LLMS TURN SELF-TESTING AND SELF-JUDGING INTO SELF-IMPROVEMENT?

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) increasingly interact with external environments and accumulate substantial behavioral experience, yet existing agent benchmarks largely evaluate them as fixed policies. It therefore remains unclear whether an agent can actively test its behavior, judge the resulting experience, and use that experience to improve future decisions. We introduce **S^3Gym**, an interactive benchmark for evaluating LLM self-improvement through three coupled capabilities: **Self-Testing**, **Self-Judging**, and **Self-Improvement**. Across seven text-based games, the protocol separates permissive exploration from strict held-out evaluation and uses executable environment verifiers to audit self-judgments and performance. We compare History ICL and score-conditioned Summary Memory across seven proprietary models, and separately study parameter training with Qwen3-8B. Self-judgments exhibit different levels of agreement with reward occurrence and normalized reward magnitude, while their association with subsequent performance gains is weak. The relative benefits of Summary Memory and History ICL vary across models and games. Selected illustrative cases suggest that explicit, state-dependent action rules grounded in observed outcomes provide more useful iteration directions than generic reminders. In the separate Qwen3-8B study, parameter training yields gains on Trust Evolution and persistent degradation on Plants-vs-Zombies (PvZ). These findings motivate evaluating judgment quality, actionable experience reuse, and retained performance separately. S^3Gym supports this diagnosis by distinguishing accumulated experience from measurable improvement in held-out behavior.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed as autonomous agents that invoke tools, execute code, and interact with external environments through repeated observation–reasoning–action–feedback loops. Such interaction is central to applications including automated research, software engineering, and open-ended decision-making, where agents continually accumulate behavioral experience. Existing benchmarks evaluate multi-turn decision-making, environment exploration, and task completion (Liu et al., 2024; Ma et al., 2024), but they typically treat the evaluated model as a fixed policy. As a result, they primarily answer a static question—how capable is the model at the time of evaluation?—while offering limited insight into whether the model can use its own past interactions to improve future behavior. We refer to this experience-driven capability as **Self-Improvement**.

Theories of experiential learning and reflective inquiry suggest that experience becomes useful only when it is actively examined, abstracted, and tested again. Dewey and Kolb emphasize the transformation of experience into knowledge through reflection and reapplication (Dewey, 1933; Kolb, 1984), while Popper characterizes progress as the iterative exposure of conjectures to tests that can reveal error (Popper, 1959). The same principle applies to LLM agents: collecting trajectories alone does not guarantee learning. An agent must generate informative evidence, interpret the causes of success and failure, and convert those interpretations into improved behavior. We therefore decompose experience-driven learning into three interdependent capabilities: **Self-Testing**, in which the agent explores strategies and gathers diagnostic evidence; **Self-Judging**, in which it evaluates actions, outcomes, and their potential reusability; and **Self-Improvement**, in which the resulting experience alters future decisions (Schön, 1983; Argyris & Schön, 1978).

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

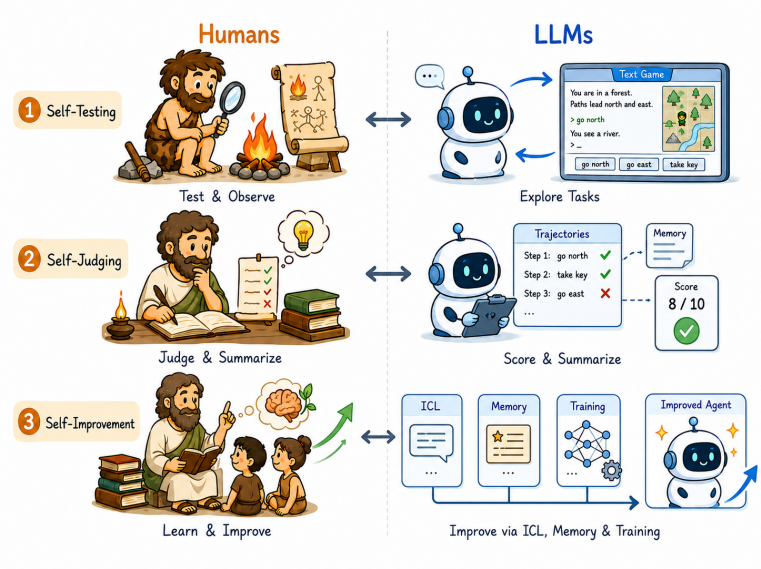


Figure 1: Conceptual overview of experience-driven improvement in humans and LLM agents. Both processes involve generating experience through Self-Testing, interpreting it through Self-Judging, and incorporating it through reusable memory or learning mechanisms.

Experience that has been tested and judged can be incorporated at different levels of persistence. At the short-term **in-context learning** level, an agent directly conditions future decisions on previous observations, actions, and scores. At the intermediate **external-memory** level, long trajectories are compressed into natural-language rules, failure patterns, or reusable skills, as in Reflexion and Voyager (Shinn et al., 2023; Wang et al., 2023). At the long-term **training** level, selected or corrected trajectories are internalized through parameter updates, following the broader direction explored by STaR, ReST, and Re-ReST (Zelikman et al., 2022; Gulcehre et al., 2023; Dou et al., 2024). These mechanisms make different trade-offs: raw histories preserve detailed evidence but consume context, summaries provide compact abstractions but depend on judgment quality, and training offers persistent internalization but may also amplify incorrectly judged experience. A common exploration and evaluation protocol can support comparisons of context-level pathways and separate studies of parameter training.

To address this need, we introduce **S^{3}Gym**, short for **Self-Testing, Self-Judging, and Self-Improvement Gym**, an interactive benchmark for evaluating whether LLMs can become more capable through their own environmental experience. Each evaluation consists of an **Exploration Phase** and an **Evaluation Phase**, reflecting the distinction between searching for new strategies and exploiting acquired knowledge (March, 1991). During exploration, the agent interacts with relatively permissive environment configurations, generating successful, unsuccessful, and partially successful trajectories. S^{3}Gym separates model-generated self-judgments from environment-verifiable outcomes: self-judgments determine how interaction experience is organized and reused, whereas environment scores measure the resulting behavioral performance. The updated agent is then evaluated on stricter and held-out configurations, testing whether experience transfers to new conditions. Recording judgments and outcomes also allows us to examine associations between **Self-Judging** and subsequent **Self-Improvement**.

We instantiate S^{3}Gym (Figure 1) with text-based games, following the game-based evaluation paradigm of KORGym (Shi et al., 2025). Whereas KORGym evaluates LLM reasoning through diverse interactive games, S^{3}Gym uses such environments as controlled testbeds for **experience-driven self-improvement**. Their multi-turn interaction, partial observability, delayed consequences, and long-horizon decision-making provide rich behavioral trajectories, while programmatically verifiable rules enable reproducible evaluation with objective step-level or terminal scores. Related environments, including TextWorld, ALFWorld, and ScienceWorld, have similarly demonstrated the utility of textual interaction for evaluating planning, embodied reasoning, and scientific problem

solving (Côté et al., 2018; Shridhar et al., 2021; Wang et al., 2022). S^{3}Gym includes seven games—CHESS, MINESWEEPER, NULLIFY, TETRIS, SNAKE, PVZ, and TRUST EVOLUTION—covering latent-rule induction, constraint satisfaction, numerical transformation, spatial planning, resource allocation, survival, and multi-agent strategy. Each game provides related but distinct exploration and evaluation configurations, testing whether agents can reuse experience beyond individual actions or seeds.

Our contributions are fourfold: (1) a **formulation connecting Self-Testing, Self-Judging, and Self-Improvement** with separate exploration and held-out evaluation; (2) a comparison of **History ICL and Summary Memory**, alongside a **parameter-training study**; (3) an independent audit of **self-judgments against executable outcomes**; and (4) **empirical analyses of improvement dynamics, summary effectiveness**, and failure modes across seven games.

2 RELATED WORK

Interactive evaluation. TextWorld, ALFWorld, and ScienceWorld provide controlled environments for language-grounded planning and interaction (Côté et al., 2018; Shridhar et al., 2021; Wang et al., 2022). AgentBench, AgentBoard, and KORGym broaden the evaluation of language agents and game-based reasoning (Liu et al., 2024; Ma et al., 2024; Shi et al., 2025). S^{3}Gym uses interactive environments to measure changes in behavior after experience reuse, alongside initial task competence.

Experience reuse and model-generated supervision. Reflexion and Voyager retain textual feedback or reusable skills (Shinn et al., 2023; Wang et al., 2023), while ReasoningBank organizes self-judged experience into reasoning memory (Ouyang et al., 2025). STaR, ReST, Re-ReST, and Self-Rewarding Language Models use self-generated or model-evaluated data for optimization (Zelikman et al., 2022; Gulcehre et al., 2023; Dou et al., 2024; Yuan et al., 2024). Such approaches motivate comparing experience representations while independently checking the judgment signal. Studies of LLM judges document limitations in evaluation reliability (Zheng et al., 2023; Tan et al., 2025). Here, we compare action-level self-scores with executable rewards and examine their association with later improvement.

Adaptive-agent benchmarks. PostTrainBench, SEA-Eval, and SEAGym study post-training, sequential adaptation, and persistent harness evolution, respectively (Rank et al., 2026; Jiang et al., 2026; Zheng et al., 2026). S^{3}Gym focuses on the combination of self-scored interaction, alternative experience representations, and strict held-out evaluation. Appendix F provides an extended discussion and benchmark comparison.

3 THE S^{3}GYM PROTOCOL

3.1 EXPLORATION AND EVALUATION

Each game has an exploration configuration and an evaluation configuration. Exploration permits additional trials or uses an easier state space; evaluation imposes stricter termination rules or more demanding configurations. Agents alternate experience collection with evaluation, following the cycle

$$\text{Explore} \rightarrow \text{Judge} \rightarrow \text{Consolidate} \rightarrow \text{Update} \rightarrow \text{Evaluate}.$$

Exploration and evaluation use disjoint random seeds. Evaluation trajectories are excluded from history, summary memory, and training data. The held-out score therefore tests transfer from exploration rather than reuse of evaluation episodes. Appendix A lists the exact configuration differences.

Figure 2 summarizes the protocol.

3.2 INTERACTION AND SELF-JUDGING

At step i of episode x, the agent receives observation $O_{x,i}$, within-episode history $H_{x,i}$, and cross-episode experience $Z_t^{(p)}$. Pathway p determines the representation of that experience. The policy

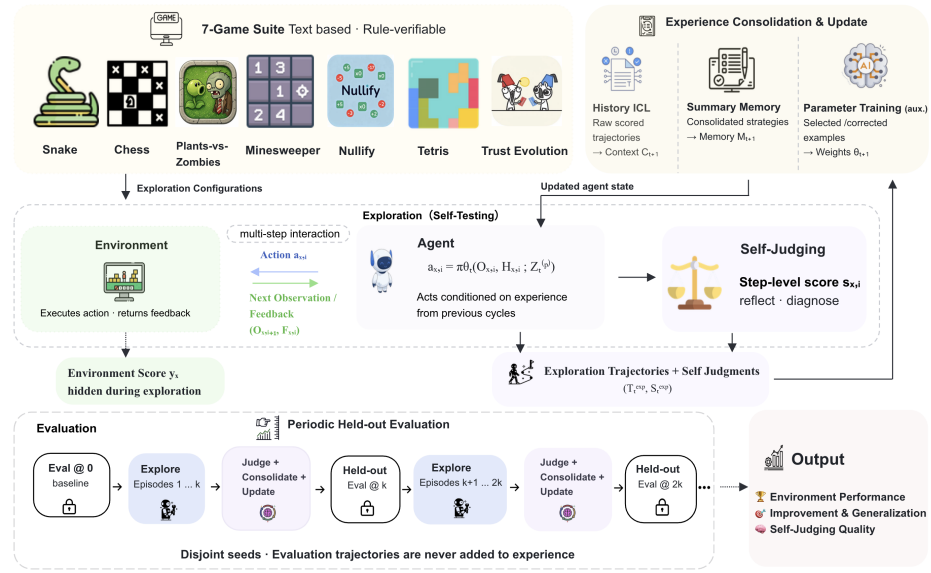


Figure 2: S^{3}Gym separates experience collection and reuse from strict held-out evaluation. History ICL and Summary Memory retain experience in context; parameter training is examined separately with Qwen3-8B.

jointly proposes an action and an associated self-score:

$$(a_{x,i}, s_{x,i}) = \pi_{\theta_t}(O_{x,i}, H_{x,i}; Z_t^{(p)}). \tag{1}$$

The self-score $s_{x,i}$ is intended to estimate the reward associated with the proposed action, rather than generic confidence. The environment executes the action and returns

$$(O_{x,i+1}, F_{x,i}, r_{x,i}, d_{x,i}) = \mathrm{Env}(O_{x,i}, a_{x,i}), \tag{2}$$

where $F_{x,i}$ is observable feedback, $r_{x,i}$ is the verifier reward, and $d_{x,i}$ indicates termination. The agent-visible history records observations, actions, self-scores, and feedback. Verifier rewards and terminal scores are withheld during exploration, although visible state changes can still convey information about outcomes. The benchmark compares $s_{x,i}$ with $r_{x,i}$ to assess judgment reliability and uses terminal scores to evaluate behavior. Appendix G documents the prompts, including ambiguities between action-level rewards and cumulative scores.

3.3 Experience Representations

History ICL. The agent receives serialized cross-episode observations, actions, and self-scores together with the current episode. This representation retains detailed evidence. When the context limit is reached, the implementation truncates the tail of the previous-episode history while preserving the current episode and current prompt.

Summary Memory. At the start of an episode, the model compresses previous exploration history into practical recommendations, using its own action scores as part of the evidence. The summary records strategies to retain, mistakes to avoid, and directions for subsequent interaction. These tips are cached within the episode; current-episode history remains uncompressed. Thus, the comparison changes cross-episode experience representation while retaining the game rules and action interface.

Parameter training. A separate study fine-tunes Qwen3-8B on exploration-derived supervised examples. The procedure retains high-scoring actions and filters or corrects low-scoring actions. Evaluation uses the updated parameters without History ICL or Summary Memory. This study tests parameter-based reuse within the same game protocol. The extended formalization appears in Appendix E.

4 CONTEXT-LEVEL EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Games and models. The seven games are Chess, Minesweeper, Nullify, Tetris, Snake, Plants-vs-Zombies (PvZ), and Trust Evolution. We report the seven proprietary models with completed results across all games: GPT-4o, GPT-4.1, o3-mini, Gemini-2.5-Flash, Gemini-2.5-Pro, GPT-5.5, and Gemini-3.5-Flash. Parameter training is examined in a separate Qwen3-8B study; it is not a controlled three-pathway comparison within the same model.

Schedule and budgets. History ICL and Summary Memory are compared within each model using matched seeds, interaction budgets, and decoding settings. The context-level comparison uses 30 exploration episodes per run, with evaluation checkpoints at $x \in \{0, 3, \ldots, 30\}$. Each checkpoint averages three strict-mode evaluation episodes. History ICL has a 100,000-token input limit; each response can generate at most 14,000 new tokens. The controller allows 64 interaction steps per episode, subject to smaller game-specific horizons. Summary Memory additionally calls the model to compress experience, so matched episode budgets do not imply equal token or compute costs. Full settings appear in Tables 5 and 6.

Metrics. For one model–pathway–game run, let y_k be the score at checkpoint x_k, with $k = 0, \ldots, K$ and $K = 10$. Let $\mathcal{V}$ index valid checkpoints. We report average score, maximum score, and positive area above the initial baseline:

$$\mathrm{Avg} = \frac{1}{|\mathcal{V}|} \sum_{k \in \mathcal{V}} y_k, \qquad \mathrm{Max} = \max_{k \in \mathcal{V}} y_k, \tag{3}$$

$$\mathrm{AUC}^{+} = \int_0^{30} \max\{0, y(x) - y_0\}\, dx. \tag{4}$$

Here $y(x)$ linearly interpolates adjacent checkpoints. Explicitly marked invalid concurrency points are excluded from averages and maxima; area calculations connect adjacent valid checkpoints. Average score describes overall capability, whereas maximum score captures the best observed checkpoint. AUC^{+} records positive excursions above initialization but does not penalize below-baseline performance, so it is not a net improvement measure. Its raw values are comparable within a game, not across games with different score scales.

4.2 MAIN RESULTS

Table 1 reports all three metrics: (a) Avg., (b) Max., and (c) AUC^{+}. Each panel compares History ICL and Summary Memory across the seven games; bold values mark the best result within each pathway and game. All per-game curves appear in Appendix H.

Capability and improvement differ. Under History ICL, Gemini-3.5-Flash has the highest average Chess score and the largest AUC^{+} on Chess, Snake, and Trust Evolution. GPT-5.5 has the highest average scores on Minesweeper, Nullify, Tetris, and Snake, but its AUC^{+} does not lead all four games. For example, Gemini-2.5-Pro has the largest Minesweeper AUC^{+} despite a lower average score. Under Summary Memory, GPT-5.5 leads average performance on Chess, Minesweeper, Nullify, Tetris, and Snake. These differences show why absolute performance and above-baseline changes must be assessed separately.

Summarization helps selectively. Gemini-2.5-Flash's Minesweeper AUC^{+} rises from 0.000 with History ICL to 7.794 with Summary Memory; its PvZ value rises from 24.402 to 238.501. GPT-5.5's Chess AUC^{+} rises from 0.474 to 16.840. Conversely, GPT-5.5's PvZ value falls from 548.499 to 33.219. Gemini-3.5-Flash falls from 12.280 to 0.000 on Chess and from 200.000 to 55.095 on Trust Evolution. Thus, Summary Memory can outperform or underperform History ICL, depending on the model and game.

Task dependence. The games differ in reward sparsity, granularity, and scale. Isolated successes can therefore be especially prominent in maximum scores. Average scores and trajectories complement these peaks by showing whether successes recur. The pathway reversals above also show that a

Table 1: Main results on seven S^3Gym games. For each model–game pair, Avg. is the mean score over all available evaluation checkpoints, Max. is the maximum score, and AUC^+ is the positive area above the initial-score baseline. Higher is better for all metrics. Explicitly marked concurrency failures are excluded.

(a) Average score (Avg.)

Model	Chess	Minesweeper	Nullify	Tetris	Snake	PvZ	TrustEvo
ICL (History Context)							
GPT-4o	0.007	0.044	0.030	0.030	0.212	11.636	8.242
GPT-4.1	0.010	0.042	0.030	0.152	0.515	16.909	7.371
o3-mini	0.012	0.222	0.030	1.273	4.000	26.333	9.242
Gemini-2.5-Flash	0.009	0.063	0.030	0.121	1.424	20.697	9.008
Gemini-2.5-Pro	0.025	0.421	0.212	0.273	1.576	28.242	14.500
GPT-5.5	0.017	**0.604**	**0.545**	**4.242**	**9.061**	44.033	7.848
Gemini-3.5-Flash	**0.547**	0.482	0.394	1.970	8.455	**44.697**	**17.242**
Summary Memory							
GPT-4o	0.005	0.013	0.000	0.030	0.303	11.818	13.947
GPT-4.1	0.010	0.012	0.030	0.182	0.545	14.727	13.371
o3-mini	0.006	0.425	0.061	1.788	6.727	26.424	7.174
Gemini-2.5-Flash	0.074	0.438	0.091	0.152	1.364	21.939	8.720
Gemini-2.5-Pro	0.110	0.466	0.182	0.545	1.273	19.667	9.568
GPT-5.5	**0.533**	**0.742**	**0.576**	**3.515**	**10.697**	33.500	8.697
Gemini-3.5-Flash	0.208	0.603	0.455	1.091	10.242	**43.879**	**14.394**

(b) Maximum score (Max.)

Model	Chess	Minesweeper	Nullify	Tetris	Snake	PvZ	TrustEvo
ICL (History Context)							
GPT-4o	0.018	0.131	0.333	0.333	0.667	17.667	11.500
GPT-4.1	0.018	0.189	0.333	0.667	1.667	21.667	10.250
o3-mini	0.022	0.500	0.333	2.667	9.667	29.333	13.250
Gemini-2.5-Flash	0.022	0.670	0.333	0.333	2.333	29.000	13.583
Gemini-2.5-Pro	0.089	0.659	0.667	1.000	4.333	33.000	19.000
GPT-5.5	0.067	**0.974**	**1.000**	**11.333**	11.333	48.000	**23.000**
Gemini-3.5-Flash	**0.742**	0.705	**1.000**	3.000	**13.000**	**62.667**	21.000
Summary Memory							
GPT-4o	0.018	0.033	0.000	0.333	1.000	20.333	19.333
GPT-4.1	0.031	0.083	0.333	0.667	1.000	19.333	15.917
o3-mini	0.018	0.608	0.333	4.000	10.333	29.000	10.083
Gemini-2.5-Flash	0.147	0.665	0.333	0.667	2.333	26.333	13.750
Gemini-2.5-Pro	0.258	0.974	0.333	2.000	3.000	26.667	14.333
GPT-5.5	**0.773**	**1.000**	**1.000**	**6.667**	**14.000**	41.333	17.000
Gemini-3.5-Flash	0.524	0.961	**1.000**	2.333	12.000	**51.667**	**20.333**

(c) Positive area above baseline (AUC^+)

Model	Chess	Minesweeper	Nullify	Tetris	Snake	PvZ	TrustEvo
ICL (History Context)							
GPT-4o	0.233	1.437	0.500	1.000	0.000	0.052	3.443
GPT-4.1	0.186	1.329	1.000	0.000	8.501	129.000	5.966
o3-mini	0.004	0.000	1.000	0.000	0.000	5.383	62.638
Gemini-2.5-Flash	0.017	0.000	1.000	0.000	7.317	24.402	6.251
Gemini-2.5-Pro	0.000	**3.539**	**1.500**	0.000	0.000	60.416	161.876
GPT-5.5	0.474	0.959	1.000	**96.251**	0.163	**548.499**	164.411
Gemini-3.5-Flash	**12.280**	1.217	0.417	34.000	**37.387**	161.945	**200.000**
Summary Memory							
GPT-4o	0.007	0.046	0.000	0.500	1.917	12.778	10.350
GPT-4.1	0.177	0.046	1.000	0.000	0.000	75.462	154.876
o3-mini	0.079	0.001	2.000	28.598	0.300	40.425	30.521
Gemini-2.5-Flash	2.147	**7.794**	3.000	0.500	0.000	**238.501**	18.443
Gemini-2.5-Pro	0.573	0.721	0.000	10.084	**11.321**	11.872	10.624
GPT-5.5	**16.840**	2.951	**7.418**	**50.801**	0.000	33.219	**240.500**
Gemini-3.5-Flash	0.000	0.793	5.584	15.808	0.750	123.535	55.095

representation effective for one model–game pair need not remain effective elsewhere. These results support evaluating experience reuse at the model–game level, rather than treating summarization as a universal upgrade.

5 Analysis

5.1 RQ1: Can Parameter Training Enable Self-Improvement?

The separate Qwen3-8B study evaluates 20 consecutive checkpoints: the original model at episode 0 and updates at episodes 1–19. All seven games use strict evaluation without history or summary memory at inference. Training AUC^+ integrates positive improvement over episodes 0–19, whereas the context-level experiments cover episodes 0–30.

Trust Evolution improves from 0 to a maximum of 30, exceeds baseline at 18 of 19 updated checkpoints, with a post-training average of 8.684 and AUC^+ of 163.5. Chess reaches 0.0667 at the final checkpoint after mostly zero scores. Snake reaches 1 at episodes 8, 12, and 15, but returns to zero. Minesweeper, Nullify, and Tetris remain at zero. PvZ degrades from an initial score of 23 to 6 at every updated checkpoint. Figure 3 shows the complete training curve; Appendix D provides additional per-game observations.

One hypothesis for the loss of gains across updates is overfitting to exploration-specific patterns. Parameter updates may increasingly favor behaviors suited to permissive exploration conditions, reducing their transfer to stricter held-out evaluation.

The separate Qwen3-8B study captures gains on Trust Evolution, intermittent successes on Snake, and persistent degradation on PvZ. These trajectories distinguish improvement without inference-time memory from the retention of gains across subsequent updates.

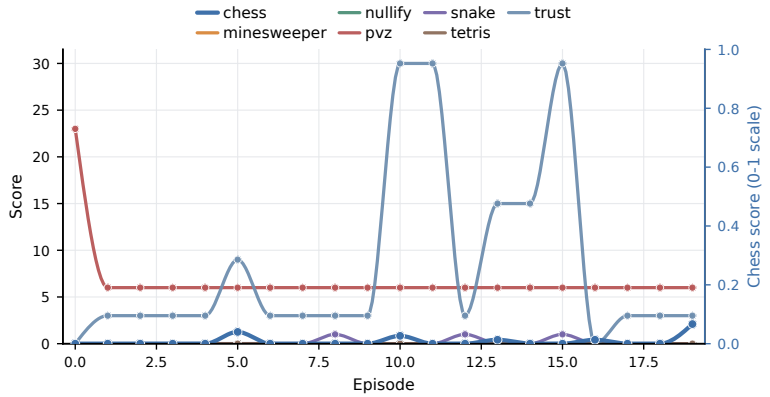


Figure 3: Strict-mode evaluation performance of Qwen3-8B over 20 checkpoints spanning episodes 0–19. Episode 0 denotes the original model. Because the games use different score scales, each trajectory should be interpreted relative to its own initial performance.

5.2 RQ2: Is Agent Self-Judging Reliable?

Step-level reliability. We audit 116,117 logged exploration and evaluation transitions across 98 runs, including records beyond episode 30 (Appendix B). Event agreement records whether $s_i > 0$ and $r_i > 0$ coincide. Normalized mean absolute error (NMAE) measures magnitude discrepancy after game-level min–max normalization. Here, over-confidence denotes $\Pr(s_i > 0, r_i \leq 0)$; under-confidence denotes $\Pr(s_i \leq 0, r_i > 0)$. Table 2 reports per-game macro-averages; Appendix B specifies normalization bounds and equations.

Agreement ranged from 0.496 on Chess to 0.881 on PvZ. However, PvZ had the largest NMAE (0.882), compared with 0.041 on Tetris and 0.056 on Nullify. Chess and Trust exhibited substantial over-confidence (0.365 and 0.291). Thus, agreement about positive outcomes did not imply agreement in normalized magnitudes, particularly where zero-reward transitions were common.

Table 2: Step-level reliability of model Self-Judging. Agreement measures binary agreement on whether a transition receives a positive score. Normalized mean absolute error (NMAE) measures magnitude discrepancy after game-level min–max normalization.

Game	Agreement	NMAE	Over-conf.	Under-conf.
Chess	0.496	0.141	0.365	0.139
Minesweeper	0.820	0.524	0.062	0.118
Nullify	0.827	0.056	0.170	0.004
PvZ	0.881	0.882	0.032	0.087
Snake	0.879	0.121	0.110	0.011
Tetris	0.846	0.041	0.151	0.002
Trust	0.665	0.392	0.291	0.044

Judgment–improvement coupling. The following blockwise and run-level analyses describe associations; they do not estimate the effects of intervening on judgment quality. Across 985 logged exploration blocks, we relate event agreement A and magnitude error E to the next normalized evaluation-score change g. Table 3 reports both correlations and the gain gaps between the highest and lowest judgment-quality tertiles. Across games, $\rho(A, g) = -0.010$ and $\rho(-E, g) = -0.018$; the corresponding gaps were -0.004 and -0.016. Minesweeper showed weak positive agreement coupling (0.071), but negligible magnitude-error coupling (0.006). These descriptive associations do not support local scoring accuracy as a reliable predictor of subsequent improvement.

Table 3: Blockwise judgment–improvement coupling. A is event agreement, E is normalized magnitude error, and g is the normalized strict-mode score change at the subsequent evaluation. The gap metrics compare the highest and lowest judgment- quality tertiles.

Game	$\rho(A, g)$	$\rho(-E, g)$	Gap_A	Gap_E
All	-0.010	-0.018	-0.004	-0.016
Chess	-0.014	-0.058	0.028	-0.039
Minesweeper	0.071	0.006	0.069	0.005
Nullify	-0.032	-0.082	-0.065	-0.058
PvZ	-0.038	-0.023	0.014	0.010
Snake	0.022	0.022	0.009	0.009
Tetris	-0.048	-0.021	0.010	0.004
Trust	-0.014	-0.003	-0.001	0.006

For run-level comparisons, normalized above-baseline area is

$$\mathrm{NABA} = \frac{\mathrm{AUC}^+}{\max(y_0, \epsilon)}, \tag{5}$$

where ϵ is the smallest strictly positive checkpoint score in the run; an all-zero run has NABA zero. AUC^+ integrates the positive part of the linearly interpolated curve using actual episode widths. Small baselines can inflate NABA, and below-baseline regressions are not penalized. Event agreement correlated weakly and negatively with NABA (Pearson $r = -0.24$; Spearman $\rho = -0.17$). **Reward occurrence agreement, normalized magnitude agreement, and subsequent performance gains provide distinct assessments of Self-Judging.** Their weak observed coupling motivates measuring local judgment quality and downstream improvement separately.

5.3 RQ3: Do Score-Conditioned Summaries Produce Effective Iteration Directions?

Summary Memory versus direct history. We compare the two pathways using $\Delta\mathrm{NABA}$, defined as Summary Memory minus History ICL with pathway-specific baselines (Equation 5). Table 4 summarizes all 49 model–game pairs. Mean differences favored Summary Memory on Chess, Nullify, and Trust; Tetris was nearly balanced, with three wins per pathway and one tie. History ICL had higher mean NABA on Minesweeper, PvZ, and Snake.

Table 4: Summary Memory compared with direct-history ICL. Each pair evaluates the same model and game under the two history-organization strategies.

Game	Pairs	Summary wins	History wins	Ties	Mean ΔNABA
Chess	7	4	3	0	23.19
Minesweeper	7	3	4	0	-9.92
Nullify	7	4	2	1	5.70
PvZ	7	3	4	0	-1.22
Snake	7	3	4	0	-2.70
Tetris	7	3	3	1	0.25
Trust	7	4	3	0	24.49

Cases with positive overall differences. The selected illustrative cases below link summary content to observed actions and outcomes (Appendix C). Four cases pair explicit rules with positive overall differences. Parenthetical values below give ΔNABA. GPT-5.5/Trust selected `cheat`, received reward 3.0, and had $\Delta\text{NABA} = +185.70$. Its summary recommended defection under the observed payoff structure. The o3-mini/Tetris summary recommended look-ahead placement and hole avoidance; the agent cleared four lines in episode 10 ($+28.60$). It selected `8 0` twice, with self-score and reward 1.0 on each move. Gemini-2.5-Flash/Minesweeper used local constraints, selected `flag (2,2)`, and received 0.0625 ($+42.87$). Gemini-3.5-Flash/Nullify constructed exact opposites, then selected `0 1` to cancel -19 and 19, receiving 1.0 ($+16.13$).

Failures and regressions. Generic rules also coexisted with failures. GPT-5.5/PvZ opened with `X 1 0` and received reward 1.0. Its later strict score fell from 38.67 to 32.33 ($\Delta\text{NABA} = -19.58$). Gemini-3.5-Flash/Chess returned `END` with zero reward despite a detailed rule summary (-76.75). GPT-4.1/Minesweeper selected `uncover (8,0)` after exhausting its lives; its subsequent checkpoint score fell from 0.0606 to 0 (-54.86). GPT-4o/Snake selected `DOWN` without reward, and its strict score fell from 1.0 to 0. Nevertheless, its overall difference was positive ($+5.75$), illustrating that NABA does not penalize local regressions.

The relative benefits of Summary Memory and History ICL vary across models and games. Summaries that connect observed outcomes to explicit, state-dependent action rules appear more useful for the next iteration than generic reminders. Favorable cases encode procedures such as look-ahead placement, local constraint propagation, or constructing exact opposites. In contrast, general advice about defense or obstacle avoidance may omit the configurations and timing needed to choose the next action. These observations suggest retaining each rule's applicability conditions and the evidence for repeating or revising it.

6 Conclusion

S^3Gym connects self-scored interaction experience with executable verification and held-out evaluation across seven text-based games. Its experiments distinguish local judgment agreement from subsequent performance gains and reveal model- and task-dependent benefits of history and summary memory. A separate parameter-training study captures both task-specific gains and persistent degradation. Together, these results demonstrate how S^3Gym makes judgment quality, experience reuse, and performance retention separately observable when evaluating experience-driven agents.

AI USE STATEMENT

Generative language models generate the actions, self-scores, summaries, and exploration-derived training examples examined in this study. These uses form part of the experimental protocol described in the main text and appendices. We also used a generative AI assistant to improve the clarity and fluency of the manuscript.

ETHICS STATEMENT

S^3Gym evaluates agents in text-based games; its scores do not establish reliability or readiness for deployment in high-stakes settings. The protocol separates exploration from held-out evaluation through disjoint random seeds and excludes evaluation trajectories from experience reuse and training. Improvements in game scores do not certify safety or the preservation of capabilities outside the evaluated tasks. Self-generated judgments and summaries may retain errors, and parameter updates may degrade previously effective behavior. Releasing evaluation instances and raw interaction traces requires care to prevent held-out content from entering subsequent learning data.

REPRODUCIBILITY STATEMENT

Section 3 describes the exploration/evaluation separation and experience representations. Appendix A provides game configurations and runtime budgets; Appendix G gives the game interfaces and prompt templates. The main result tables contain all context-level aggregate metrics. Appendices H, B, C, and D provide trajectories and detailed analyses. Appendix E records the extended interaction formalization.

REFERENCES

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Self-Improving LLM Agents at Test-Time. *arXiv preprint arXiv:2510.07841*, 2025. URL https://arxiv.org/abs/2510.07841.

Chris Argyris and Donald A. Schön. *Organizational Learning: A Theory of Action Perspective*. Addison-Wesley, Reading, MA, 1978. ISBN 9780201001747.

Yizhe Chi, Wenyi Li, Deyao Hong, Xiaoqiu Wang, Mingju Gao, Kaisen Yang, Bingxiang He, Youjie Zheng, Calvin Xiao, and Qinhuai Na. AI4AI-Bench: Benchmarking LLM Agents in Algorithmic Design for Recursive Self-Improvement. *arXiv preprint arXiv:2608.20318*, 2026. URL https://arxiv.org/abs/2608.20318.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A Learning Environment for Text-Based Games. *arXiv preprint arXiv:1806.11532*, 2018. URL https://arxiv.org/abs/1806.11532.

Bo Deng, Kang Zhou, Lifan Guo, Chongyang Tao, Xuanren Chen, Chenggang Xie, Renzhao Liang, Feng Chen, and Chi Zhang. FinEvo-Bench: A Longitudinal Benchmark for Self-Evolving Agents in Professional Financial Workflows. *arXiv preprint arXiv:2608.06144*, 2026a. URL https://arxiv.org/abs/2608.06144.

Zihao Deng, Yining Zhu, Leiming Wang, Junbo Wang, and Jingfei Lu. Tree-of-Experience: Hierarchical Experience Management for Self-Evolving Agents. *arXiv preprint arXiv:2608.09044*, 2026b. URL https://arxiv.org/abs/2608.09044.

John Dewey. *How We Think: A Restatement of the Relation of Reflective Thinking to the Educative Process*. D. C. Heath and Company, Boston, MA, 2 edition, 1933.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-ReST: Reflection-Reinforced Self-Training for Language Agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15394–15411, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.861. URL https://aclanthology.org/2024.emnlp-main.861/.

Tianyi Guan, Yiding Wang, Haotong Yang, Siyuan Cao, Shirui Liu, Yi Hu, Jiaqi Li, and Muhan Zhang. ContinualSkillBench: Can LLM Agents Truly Evolve Their Capabilities? *arXiv preprint arXiv:2608.03874*, 2026. URL https://arxiv.org/abs/2608.03874.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced Self-Training (ReST) for Language Modeling. *arXiv preprint arXiv:2308.08998*, 2023. URL https://arxiv.org/abs/2308.08998.

Keyu He, Xuhui Zhou, and Maarten Sap. Social Gym and SPaRTan: Benchmarking and Improving LLM Social Reasoning via Multi-Agent Game Tournaments. *arXiv preprint arXiv:2608.09128*, 2026. URL https://arxiv.org/abs/2608.09128.

Sihang Jiang, Lipeng Ma, Zhonghua Hong, Keyi Wang, Zhiyu Lu, Shisong Chen, Jinghao Zhang, Tianjun Pan, Weijia Zhou, Jiaqing Liang, and Yanghua Xiao. SEA-Eval: A Benchmark for Evaluating Self-Evolving Agents Beyond Episodic Assessment. *arXiv preprint arXiv:2604.08988*, 2026. URL https://arxiv.org/abs/2604.08988.

David A. Kolb. *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall, Englewood Cliffs, NJ, 1984. ISBN 9780132952613.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2401.13178.

James G. March. Exploration and Exploitation in Organizational Learning. *Organization Science*, 2(1):71–87, 1991. doi: 10.1287/orsc.2.1.71.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. *arXiv preprint arXiv:2509.25140*, 2025. URL https://arxiv.org/abs/2509.25140.

Karl R. Popper. *The Logic of Scientific Discovery*. Hutchinson, London, 1959.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM Agents Automate LLM Post-Training? In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2603.08640.

Donald A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, NY, 1983. ISBN 9780465068746.

Linfang Shang, Ming Xu, Yiding Sun, Tianle Xia, Lingxiang Hu, Lan Xu, and Ning Zheng. When Self-Evolution Backfires: Pre-Commit Gating against Skill Contamination in LLM Agents. *arXiv preprint arXiv:2608.05810*, 2026. URL https://arxiv.org/abs/2608.05810.

Jiajun Shi, Jian Yang, Jiaheng Liu, Xingyuan Bu, Jiangjie Chen, Junting Zhou, Kaijing Ma, Zhoufutu Wen, Bingli Wang, Yancheng He, Liang Song, Hualei Zhu, Shilong Li, Xingjian Wang, Wei Zhang, Ruibin Yuan, Yifan Yao, Wenjun Yang, Yunli Wang, Siyuan Fang, Siyu Yuan, Qianyu He, Xiangru Tang, Yingshui Tan, Wangchunshu Zhou, Zhaoxiang Zhang, Zhoujun Li, Wenhao Huang, and Ge Zhang. KORGym: A Dynamic Game Platform for LLM Reasoning Evaluation. *arXiv e-prints*, art. arXiv:2505.14552, May 2025. doi: 10.48550/arXiv.2505.14552.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *The Ninth International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A Benchmark for Evaluating LLM-Based Judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is Your Agent Smarter than a 5th Grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Wenxiao Wang, Priyatham Kattakinda, and Soheil Feizi. Do Agent Optimizers Compound? A Continual-Learning Evaluation on Terminal-Bench 2.0. *arXiv preprint arXiv:2607.14004*, 2026a. URL https://arxiv.org/abs/2607.14004.

Zefeng Wang, Minxi Yan, Jinhe Bi, Sikuan Yan, Volker Tresp, and Yunpu Ma. MetaSkill-Evolve: Recursive Self-Improvement of LLM Agents via Two-Timescale Meta-Skill Evolution. *arXiv preprint arXiv:2607.05297*, 2026b. URL https://arxiv.org/abs/2607.05297.

Shuhan Xue, Zixin Ding, Yichen Shen, Yinjie Wang, Zhenfei Yin, Yingcheng Wu, Yuxin Chen, Mengdi Wang, and Ling Yang. PAST-Bench: Benchmarking the Foundations of Recursive Self-Improvement in Personal Agents. *arXiv preprint arXiv:2608.04003*, 2026. URL https://arxiv.org/abs/2608.04003.

Qinyuan Ye, Yu Li, Yada Pruksachatkun, Jiaxin Zhang, and Chien-Sheng Wu. On the Fragility of Self-Improving Agents: Variance, Task Order, and Underspecification. *arXiv preprint arXiv:2608.18066*, 2026. URL https://arxiv.org/abs/2608.18066.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E. Weston. Self-Rewarding Language Models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488, 2022. URL https://arxiv.org/abs/2203.14465.

Haiyue Zhang. Credit Without Ground Truth: Auditing Step-Level Credit Assignment in LLM Agents Against Executed Replay. *arXiv preprint arXiv:2608.19760*, 2026. URL https://arxiv.org/abs/2608.19760.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. RetroAgent: From Solving to Evolving via Retrospective Dual Intrinsic Feedback. *arXiv preprint arXiv:2603.08561*, 2026. URL https://arxiv.org/abs/2603.08561.

Congjie Zheng, Chuanyi Xue, Bin Liang, Jun Yang, and Changshui Zhang. SEAGym: An Evaluation Environment for Self-Evolving LLM Agents. *arXiv preprint arXiv:2606.17546*, 2026. URL https://arxiv.org/abs/2606.17546.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. URL https://arxiv.org/abs/2306.05685.

A Environment and runtime configurations

The following configurations are those used in the reported experiments. Game-specific horizons override the controller cap. The game configurations appear in Table 5; Table 6 gives the runtime budget.

Table 5: Exploration and evaluation configurations for the seven S^{3}Gym games. Exploration generally uses more permissive rules, while evaluation uses stricter or more difficult configurations.

Game	Exploration	Evaluation	Max steps
Chess	15×15 board, 12 pieces, and a 7-step history.	Same rules as exploration but 22 pieces.	5
Minesweeper	9×9 board with 10–20 mines and two extra lives.	The first mine hit terminates the episode.	64
Nullify	Expressions generated with 3–7 construction steps.	Expressions generated with 5–10 construction steps.	50
Plants-vs-Zombies	3×7 battlefield.	5×6 battlefield.	64
Snake	Collisions and invalid actions are treated as no-ops.	Collisions and invalid actions terminate the episode.	64
Tetris	10×10 board.	8×8 board.	64
Trust Evolution	Opponent sampling is biased toward easier strategies.	Opponent strategies are sampled uniformly.	10

Table 6: Runtime configuration for the main context-level experiments. Game-specific horizons smaller than 64 override the controller-level step cap.

Setting	Value
History-ICL input limit	100,000 tokens
Output generation cap	14,000 new tokens per response
Controller interaction cap	64 steps per episode
Exploration budget	30 exploration episodes
Evaluation interval	Every 3 exploration episodes
Evaluation set size	3 strict-mode episodes per checkpoint
Evaluation checkpoints	$x \in \{0, 3, 6, \ldots, 30\}$

B Detailed Self-Judging and summary analyses

Self-Judging is a central component of S^{3}Gym because the model-generated score s_i is stored with each transition and subsequently influences trajectory selection, summary construction, and training-data generation. If s_i is misaligned with the environment signal, the agent may preserve harmful actions as useful experience or discard transitions that should have been retained. We therefore evaluate Self-Judging as an independent capability and examine whether judgment quality predicts subsequent improvement.

Step-level reliability. For each transition, we compare the model-generated score s_i with the environment reward r_i. We first measure event agreement, which indicates whether the model and environment agree on the presence of a positive outcome:

$$\text{Agree} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\mathbf{1}(s_i > 0) = \mathbf{1}(r_i > 0)\right]. \tag{6}$$

Here N is the number of recorded transitions in one run. We compute each metric within a run, then average equally across the 14 model–pathway runs for each game. Because reward scales differ across games, we additionally report normalized mean absolute error:

$$\mathrm{NMAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{s_i - s_{\min}}{s_{\max} - s_{\min}} - \frac{r_i - r_{\min}}{r_{\max} - r_{\min}}\right|. \tag{7}$$

The normalization bounds are pooled within each game across the selected runs. For self-scores, the observed ranges are $[0, 225]$ (Chess), $[0, 1]$ (Minesweeper), $[0, 393.0002]$ (Nullify), $[0, 64]$ (PvZ), $[0, 1]$ (Snake), $[0, 4]$ (Tetris), and $[0, 30]$ (Trust Evolution). The corresponding reward ranges are $[0, 1]$, $[-0.1, 0.1]$, $[0, 1]$, $[0, 1]$, $[0, 1]$, $[0, 4]$, and $[-1, 3]$. A zero-width range is mapped to zero in the normalized representation. These empirical ranges make NMAE sensitive to extreme self-scores; it is a relative discrepancy measure, not a proper scoring rule for probabilistic confidence.

We also measure over-confidence, $\Pr(s_i > 0, r_i \leq 0)$, and under-confidence, $\Pr(s_i \leq 0, r_i > 0)$. The analysis covers the seven target models and games under both History ICL and Summary Memory, comprising 98 runs and 116,117 transitions. This step-level audit pools available exploration and evaluation records, with 57,062 exploration and 59,055 evaluation transitions. It includes logs beyond the 30-episode window used for the main context-level metrics. The blockwise analysis uses only exploration transitions and includes 985 intervals, of which 18 end after episode 30, with the latest ending at episode 54. Four intervals span six rather than three episodes because intermediate checkpoints are missing. Thus, the judgment audit and the 30-episode performance comparison have different observation windows.

Table 2 shows different levels of agreement with reward occurrence and normalized reward magnitude. PvZ, Snake, Tetris, Minesweeper, and Nullify exhibit relatively high event agreement, but this result is partly explained by the large proportion of zero-reward transitions. NMAE measures agreement in relative magnitudes after separately rescaling self-scores and rewards; it does not establish absolute reward calibration. Tetris and Nullify achieve low NMAE, whereas PvZ exhibits the largest magnitude error despite its high binary agreement. Binary event agreement and normalized score error can therefore give different assessments of judgment reliability.

Chess and Trust are the clearest failure cases. Chess has event agreement of 0.496 and the highest over-confidence rate. A prediction may be syntactically complete while still misplacing several pieces, so its self-score may diverge from the environment reward. Trust similarly exhibits weak agreement, high NMAE, and substantial over-confidence.

Judgment–improvement coupling. For self-improvement, reliable local judgments must also be translated into better subsequent behavior. To test this relationship, we divide exploration trajectories into blocks B_t between adjacent evaluation checkpoints. For each block, we calculate event agreement A_t, normalized magnitude error E_t, and the normalized score change at the following evaluation:

$$g_{m,p,\gamma,t} = \frac{y_{m,p,\gamma,t} - y_{m,p,\gamma,t^-}}{R_\gamma}, \qquad R_\gamma = \max_{m,p,k} y_{m,p,\gamma,k} - \min_{m,p,k} y_{m,p,\gamma,k}. \tag{8}$$

Here m indexes models, p pathways, γ games, and t logged evaluation checkpoints. For each run, t^- is the preceding available checkpoint. The range is computed within each game across the observed block-endpoint scores; a zero range is replaced by one.

We then measure the Spearman correlations $\rho(A_t, g_t)$ and $\rho(-E_t, g_t)$, together with the difference in average score gain between the highest- and lowest-quality judgment tertiles.

The coupling results are close to zero across the full benchmark: $\rho(A, g) = -0.010$ and $\rho(-E, g) = -0.018$. The corresponding tertile gaps are also small and slightly negative. Blocks from the same run are dependent, so these pooled statistics are descriptive rather than evidence from independent replicates. Minesweeper exhibits a weak positive relationship between event agreement and subsequent score gain, but its magnitude-error coupling remains negligible. Overall, local scoring accuracy does not reliably predict improvement at the next evaluation checkpoint.

For the run-level analysis, NABA follows Equation 5. Let $y(x)$ interpolate the recorded checkpoint scores linearly over exploration episodes $x \in [0, 30]$. We compute

$$\mathrm{AUC}^+ = \int_0^{30} \max\{0, y(x) - y_0\}\, dx, \qquad \epsilon = \min_{k: y_k > 0} y_k. \tag{9}$$

For an all-zero run, NABA is defined as zero. For each adjacent checkpoint interval of width h, let a and b be its endpoint scores minus y_0. Its exact contribution to AUC^+ is

$$I(a,b;h) = \begin{cases} h(a+b)/2, & a \geq 0,\ b \geq 0, \\ ha^2/[2(a-b)], & a > 0 > b, \\ hb^2/[2(b-a)], & b > 0 > a, \\ 0, & a \leq 0,\ b \leq 0. \end{cases} \tag{10}$$

This calculation splits intervals at baseline crossings and uses their actual episode widths. The accompanying source includes the recorded checkpoints and a reference implementation in `analysis/compute_naba.py`. Small positive baselines can produce large NABA values.

Across the 98 runs, pooled exploration/evaluation event agreement has weak negative correlations with 30-episode NABA (Pearson $r = -0.24$; Spearman $\rho = -0.17$). Because event agreement also includes evaluation transitions, this is a retrospective association rather than a prospective prediction.

B.1 Summary Memory versus History ICL

Summary Memory imposes a stronger requirement than direct history reuse. Rather than merely conditioning on previous trajectories, the model must identify which low-scoring behaviors should be suppressed, which high-scoring behaviors should be retained, and how these observations should be converted into an executable strategy for future episodes. We evaluate this capability by comparing score-conditioned summaries with direct-history ICL and by inspecting representative successful and unsuccessful summaries.

Summary Memory versus direct history. For every model–game pair, we calculate $\Delta\mathrm{NABA}$ as Summary Memory NABA minus History ICL NABA. Positive values indicate greater normalized positive above-baseline area under Summary Memory, using the metric defined in Equation 5.

Mean NABA favors Summary Memory on Chess, Nullify, and Trust, while Tetris is nearly balanced and has equal win counts. Mean NABA favors direct history on Minesweeper, PvZ, and Snake. The selected cases in Appendix C illustrate how summaries can retain reusable action rules or omit state details, including a local failure despite a positive overall NABA difference.

C TRAJECTORY-LEVEL CASE STUDIES

The following eight selected illustrative cases connect summary content with recorded actions, rewards, and evaluation outcomes. Categories distinguish the sign of the overall ΔNABA from local trajectory failures.

POSITIVE OVERALL DIFFERENCES

GPT-5.5 / Trust

Summary direction. Always defect when the observed payoff structure consistently favors defection; avoid unnecessary reciprocal exploration.

Trajectory evidence. At the first decision, the agent selected `cheat` and received an environment reward of 3.0.

Outcome. $\Delta\text{NABA} = +185.70$.

Interpretation. The summary translates payoff feedback into an explicit rule conditioned on the observed payoff structure.

o3-mini / Tetris

Summary direction. Jointly inspect the current and next blocks, test alternative rotations, prioritize line-clearing placements, and maintain a low surface without isolated holes.

Trajectory evidence. In episode 10, the agent cleared four lines. It selected `8 0` twice, with both its self-score and the environment reward equal to 1.0 on each move.

Outcome. $\Delta\text{NABA} = +28.60$.

Interpretation. The summary converts successful line clears into an executable look-ahead rule.

Gemini-2.5-Flash / Minesweeper

Summary direction. Apply local number constraints: expand zero regions, flag forced mines, and uncover cells that are provably safe.

Trajectory evidence. The agent selected `flag (2,2)` on a partially revealed board and received a reward of 0.0625.

Outcome. $\Delta\text{NABA} = +42.87$.

Interpretation. This case pairs executable constraint rules with a positive overall difference; it does not isolate their contribution.

Gemini-3.5-Flash / Nullify

Summary direction. Plan backward to create exact opposites, use floor or ceiling operations to remove decimals, and track index changes after every merge.

Trajectory evidence. With terminal units `-19` and `19`, the agent selected `0 1` and received a reward of 1.0.

Outcome. $\Delta\text{NABA} = +16.13$.

Interpretation. The summary specifies a concrete arithmetic planning procedure consistent with the observed cancellation.

NEGATIVE OVERALL DIFFERENCES AND LOCAL FAILURE

The first three cases report negative overall differences. The Snake case reports a local failure despite a positive overall difference.

GPT-5.5 / PvZ

Summary direction. Prioritize sun production, early defense, Wall-nut stalling, and coverage of threatened lanes.

Trajectory evidence. The agent opened with `X 1 0` and received a reward of 1.0, but its later strict score fell from 38.67 to 32.33.

Outcome. $\Delta\text{NABA} = -19.58$.

Interpretation. The advice may omit the lane timing, zombie positions, and sun budget needed to choose an effective action.

Gemini-3.5-Flash / Chess

Summary direction. Track the movement rule of every piece and verify all coordinates before submitting the answer.

Trajectory evidence. In a low-scoring episode, the model produced `END` and received zero total reward despite having generated a detailed rule summary.

Outcome. $\Delta\text{NABA} = -76.75$.

Interpretation. The rule summary did not lead to a successful multi-piece prediction in this episode.

GPT-4.1 / Minesweeper

Summary direction. Begin from corners, expand zero regions, flag only certain mines, and stop when no safe progress is available.

Trajectory evidence. The agent selected `uncover (8,0)` after exhausting its available lives, and the subsequent checkpoint score fell from 0.0606 to 0.

Outcome. $\Delta\text{NABA} = -54.86$.

Interpretation. The generic heuristics were insufficient to prevent the reported failure.

GPT-4o / Snake

Summary direction. Move toward food while avoiding walls, obstacles, the snake body, and direct reversals.

Trajectory evidence. Given a board containing food, obstacles, and its own body, the agent selected `DOWN`; it received no reward, and the strict score fell from 1.0 to 0.

Outcome. $\Delta\text{NABA} = +5.75$.

Interpretation. The local failure is consistent with missing geometric detail, but the overall positive NABA difference favors Summary Memory. Positive-area integration does not penalize the reported regression.

These selected cases suggest that useful summaries need both informative feedback and action rules precise enough to apply in a new state. They suggest preserving state-specific conditions, concrete action rules, and observed outcomes when compressing experience for the next iteration.

D ADDITIONAL PARAMETER-TRAINING RESULTS

D.1 CHECKPOINT-LEVEL OBSERVATIONS

The separate Qwen3-8B study in RQ1 tests experience reuse through parameter updates, without inference-time history or summary memory.

We evaluate Qwen3-8B at 20 consecutive checkpoints, with episode 0 representing the original model and episodes 1–19 representing models updated using exploration trajectories. At each checkpoint, the model is evaluated on the strict configurations of all seven S^3Gym games without History ICL or Summary Memory. We report the post-training average and maximum scores, the final change $\Delta_{\text{Final}} = y_{19} - y_0$, and the positive area above the initial baseline,

$$\text{AUC}^{+} = \int_{0}^{19} \max(0, y(e) - y_0)\, de,$$

where e indexes episodes and $y(e)$ is obtained by linearly interpolating adjacent checkpoints.

The training trajectories are shown in Figure 3.

- **Training yields substantial self-improvement on Trust Evolution.** The score rises from 0 to a maximum of 30 and remains above the initial baseline at 18 of the 19 updated checkpoints. The post-training average reaches 8.684 and AUC^{+} reaches 163.5, providing the clearest observed improvement without inference-time memory.
- **The gains on Chess and Snake are limited or transient.** Chess remains at zero for most checkpoints but reaches 0.0667 at the final checkpoint, indicating a sparse yet positive gain. Snake reaches a score of 1 at episodes 8, 12, and 15 but subsequently returns to zero. These checkpoints exhibit occasional successful behavior without consistent retention.
- **Minesweeper, Nullify, and Tetris show no measurable improvement.** Their evaluation scores remain zero throughout training.
- **Training degrades performance on Plants-vs-Zombies.** The initial score is 23, whereas every updated checkpoint obtains a score of 6. This persistent degradation shows that this training run did not preserve its initial performance.
- **Parameter training produces distinct patterns of improvement and retention across tasks.** Qwen3-8B improves substantially on Trust Evolution, weakly on Chess, and intermittently on Snake, while showing no improvement on three games and severe degradation on Plants-vs-Zombies. Together, these trajectories distinguish sustained gains, intermittent successes, and persistent degradation across tasks.

E Extended interaction formalization

E.1 Interaction Protocol and Formalization

Each game consists of episodes initialized with separate random seeds. Each episode contains a sequence of interaction steps. For episode x at step i, we denote the current observation by $O_{x,i}$, the within-episode interaction history by $H_{x,i}$, the agent action by $a_{x,i}$, the agent's action-level self-score by $s_{x,i}$, the observable environment feedback by $F_{x,i}$, the verifier-computed step reward by $r_{x,i}$, and the final environment score by y_x.

An episode is initialized as

$$O_{x,0} = \mathrm{Reset}(\mathrm{seed}_x, c_x), \qquad H_{x,0} = \varnothing, \tag{11}$$

where c_x denotes the game configuration.

At each step, the agent receives the current observation, the within-episode history, and the pathway-specific experience state $Z_t^{(p)}$ inherited from previous exploration episodes. Following the actual interaction interface, the model jointly produces an action and an action-level self-score:

$$(a_{x,i}, s_{x,i}) = \pi_{\theta_t}\left(O_{x,i}, H_{x,i}; Z_t^{(p)}\right), \tag{12}$$

where $p \in \{\mathrm{History}, \mathrm{Memory}, \mathrm{Training}\}$. The score $s_{x,i}$ is intended to estimate the reward associated with the proposed action, rather than free-form confidence. Some task instructions also describe cumulative scores; Appendix G discusses this ambiguity. In the context-level pathways, $Z_t^{(p)}$ contains either retained raw history or summarized experience; for parameter training, acquired experience is instead reflected primarily through the updated parameters θ_t.

The stateful game engine executes the proposed action and returns the next observation, feedback, reward, and termination signal. The notation below does not assume that the observation reveals the complete environment state:

$$(O_{x,i+1}, F_{x,i}, r_{x,i}, d_{x,i}) = \mathrm{Env}(O_{x,i}, a_{x,i}). \tag{13}$$

$F_{x,i}$ denotes environment information available to the agent for subsequent interaction, whereas $r_{x,i}$ is retained by the benchmark as an external verification signal. In the main setting, verifier-computed rewards are not exposed to the agent during exploration. This separation allows S^3Gym to compare

the agent's self-judgment $s_{x,i}$ against the actual environment reward $r_{x,i}$ without directly exposing verifier rewards. Observable state changes may nevertheless reveal aspects of the outcome.

The agent-visible transition is then appended to the within-episode history:

$$H_{x,i+1} = H_{x,i} \oplus \text{Format}\left(O_{x,i}, a_{x,i}, s_{x,i}, F_{x,i}, O_{x,i+1}\right), \tag{14}$$

where $\oplus$ denotes ordered concatenation. Thus, the interaction history records the agent's own score together with the state transition, while the verifier reward $r_{x,i}$ remains benchmark-side information. At the next step, the agent again applies the action-and-score policy to the updated observation and history.

When the episode terminates or reaches the maximum number of steps, the environment returns the final verifier-computed score

$$y_x = \text{Score}(\tau_x), \tag{15}$$

where τ_x denotes the complete executed trajectory of episode x. The step-level reward $r_{x,i}$ is used to evaluate the reliability of Self-Judging, whereas y_x measures the episode-level behavioral performance used for benchmark evaluation. Neither signal is provided to the agent during exploration in the main setting.

E.2 Self-Judging and Experience Consolidation

Self-Judging provides an internal signal for identifying useful and harmful experience when precise environment rewards are unavailable to the agent. Low-scoring trajectory segments indicate possible mistakes or strategies to avoid, whereas high-scoring segments provide candidate strategies to retain.

S^3Gym organizes the judged experience in two context-level forms.

History ICL. Score-annotated trajectories are directly serialized and appended to future contexts:

$$C_{t+1} = \text{Serialize}(C_t, \mathcal{T}_t^{\text{exp}}, \mathcal{S}_t^{\text{exp}}), \tag{16}$$

where $\mathcal{T}_t^{\text{exp}}$ and $\mathcal{S}_t^{\text{exp}}$ are the exploration trajectories and their self-judgment scores in cycle t.

Summary Memory. The model compresses the judged trajectories into reusable experience:

$$M_{t+1} = \text{Summarize}(\text{Truncate}(C_{t+1}), \text{Rules}_g). \tag{17}$$

Here C_{t+1} contains the accumulated score-annotated exploration history, and Rules_g denotes the game instructions. The implementation recomputes the summary at each episode boundary and caches it within the episode; it does not recursively summarize the previous summary. At evaluation-cycle boundaries, this process gives M_{t+1}. Conceptually, its recommendations can be grouped as strategies to retain, mistakes to avoid, and directions for the next interaction cycle:

$$M_{t+1} = \left(R_t^{\text{retain}}, R_t^{\text{avoid}}, D_t^{\text{next}}\right). \tag{18}$$

E.3 Exploration and Evaluation Phases

Each game provides an exploration distribution $\mathcal{C}_g^{\text{exp}}$ and an evaluation distribution $\mathcal{C}_g^{\text{eval}}$. Exploration environments preserve the main task structure but use more permissive conditions, such as smaller state spaces, weaker failure penalties, or additional trial opportunities. Evaluation environments use stricter rules or more complex configurations.

A self-improvement cycle contains N_{exp} exploration episodes followed by N_{eval} evaluation episodes:

$$\left\{\tau_{t,x}^{\text{exp}}\right\}_{x=1}^{N_{\text{exp}}} \rightarrow \text{Improve} \rightarrow \left\{\tau_{t,x}^{\text{eval}}\right\}_{x=1}^{N_{\text{eval}}}. \tag{19}$$

Exploration provides diverse experience for judgment and consolidation, whereas evaluation verifies whether the resulting improvement transfers to stricter conditions. The two phases use disjoint random seeds, and evaluation trajectories are not added to history, memory, or training data.

For pathway p and game g, evaluation performance after cycle t is

$$\overline{Y}_{t,g}^{(p)} = \frac{1}{N_{\text{eval}}} \sum_{x=1}^{N_{\text{eval}}} y_{t,g,x}^{(p)}, \tag{20}$$

and change from the corresponding pathway-specific baseline is

$$\Delta_{t,g}^{(p)} = \overline{Y}_{t,g}^{(p)} - \overline{Y}_{0,g}^{(p)}. \tag{21}$$

These quantities describe checkpoint-level capability and change from initialization. The context-level result tables summarize average score, maximum score, and positive area above baseline; the training analysis additionally reports final changes.

E.4 Self-Improving Pathways

S^3Gym evaluates three pathways for incorporating exploration experience.

History ICL. Score-annotated trajectories are directly inserted into the context of later episodes:

$$Z_{t+1}^{(\text{History})} = C_{t+1}. \tag{22}$$

This pathway preserves detailed interaction evidence but is constrained by context length.

Summary Memory. The agent receives the consolidated summary rather than the complete trajectory:

$$Z_{t+1}^{(\text{Memory})} = M_{t+1}. \tag{23}$$

This pathway evaluates whether the model can abstract transferable strategies from lengthy interaction histories.

Parameter Training. Exploration trajectories and self-judgments are converted into supervised fine-tuning examples:

$$\mathcal{D}_t^{\text{SFT}} = \text{BuildSFT}(\mathcal{T}_t^{\text{exp}}, \mathcal{S}_t^{\text{exp}}), \tag{24}$$

followed by the parameter update

$$\theta_{t+1} = \text{SFT}\big(\theta_t, \mathcal{D}_t^{\text{SFT}}\big). \tag{25}$$

High-scoring actions are retained as positive examples, while low-scoring actions are filtered or replaced with corrected actions. Training is treated as an auxiliary pathway because it introduces optimization variables absent from context-level improvement.

Within each evaluated model, History ICL and Summary Memory use matched exploration seeds, evaluation seeds, interaction budgets, and decoding settings. Parameter training follows the separate Qwen3-8B study described in Section 4.1.

F Extended related work

F.1 Interactive Agent Benchmarks

Interactive benchmarks evaluate language agents through multi-turn decision-making, tool use, planning, and environment interaction. AgentBench (Liu et al., 2024) evaluates LLM agents across diverse domains, including operating systems, databases, and games, while AgentBoard (Ma et al., 2024) provides fine-grained analyses of agent progress and failure in multi-step tasks. Text-based environments such as TextWorld (Côté et al., 2018), ALFWorld (Shridhar et al., 2021), and ScienceWorld (Wang et al., 2022) further offer controllable testbeds for language-grounded planning, embodied reasoning, and scientific problem solving.

These benchmarks provide rich substrates for evaluating agent behavior, but they generally treat the evaluated model as a fixed policy. Interaction trajectories are primarily used to measure task completion or diagnose failures within a predefined evaluation setting, rather than to determine whether agents can transform their own experience into improved future behavior. S^3Gym builds on the controllability and objective verification of interactive environments, while shifting the evaluation target from static task competence to experience-driven behavioral improvement.

F.2 Experience-Based Agent Improvement

A growing body of work studies how language agents can improve by reusing their own interaction experience. At the context level, Reflexion (Shinn et al., 2023) introduces verbal reinforcement learning, where agents reflect on previous attempts and use textual feedback to guide future decisions. At the external-memory level, Voyager (Wang et al., 2023) accumulates reusable skills in an evolving library, enabling embodied agents to retain and reuse knowledge acquired during open-ended interaction. Recent memory-based approaches similarly compress trajectories into rules, failure patterns, or reusable strategies that can be retrieved in later episodes. ReasoningBank (Ouyang et al., 2025) distills generalizable reasoning strategies from an agent's self-judged successful and failed experiences. Tree-of-Experience (Deng et al., 2026b) organizes accumulated experience into a hierarchy that supports feedback attribution and cross-task transfer. MetaSkill-Evolve (Wang et al., 2026b) extends skill-file rewriting to the improvement procedure itself through two-timescale meta-skill evolution.

Experience can also be incorporated through parameter updates. STaR (Zelikman et al., 2022), ReST (Gulcehre et al., 2023), and Re-ReST (Dou et al., 2024) improve language models by generating, filtering, or refining self-produced reasoning trajectories. Self-Rewarding Language Models (Yuan et al., 2024) use model-generated preference signals for iterative optimization. In interactive environments, RetroAgent (Zhang et al., 2026) trains agents with retrospective dual intrinsic feedback so that experience acquired across episodes can be retrieved and reused rather than only implicitly encoded in parameters. Test-time self-improvement (Acikgoz et al., 2025) constructs targeted training samples from cases the agent finds uncertain. These studies demonstrate that interaction histories and self-generated supervision can improve model performance. However, they typically investigate a specific improvement mechanism within a particular task or training setting, making it difficult to disentangle whether improvement or failure results from the quality of collected experience, the reliability of experience interpretation, or the mechanism used to incorporate that experience. S^3Gym compares History ICL and Summary Memory under matched interaction budgets and examines parameter training in a separate Qwen3-8B study. All three pathways use the same exploration/evaluation distinction.

F.3 Evaluating Adaptive and Self-Improving Agents

The reliability of model-generated supervision has been studied extensively in work on LLM-based evaluation. LLM-as-a-Judge (Zheng et al., 2023) and JudgeBench (Tan et al., 2025) show that language models can provide useful evaluation signals, but their judgments remain vulnerable to biases, calibration errors, and limitations in reasoning. A recent ALFWorld audit compares step-level credit signals against executed replay. It finds no reliable incremental fidelity over matched shuffled controls for judge scores, outcome-conditioned logprob ratios, or policy confidence (Zhang, 2026). Such limitations are particularly important for self-improving agents, where inaccurate judgments may cause harmful experiences to be retained or valuable experiences to be discarded.

Recent benchmarks have started to move beyond static capability evaluation toward adaptive and self-improving systems. PostTrainBench (Rank et al., 2026) evaluates automated post-training pipelines that improve model parameters on held-out tasks. SEA-Eval (Jiang et al., 2026) studies long-term agent evolution through sequential task interactions, while SEAGym (Zheng et al., 2026) investigates the evolution of persistent agent harnesses, including prompts, memories, tools, and workflows. PAST-Bench (Xue et al., 2026) tests whether retained experience actually improves personal agents by running matched task sequences with retained experience turned on and off. ContinualSkillBench (Guan et al., 2026) measures in-context continual skill learning and finds that gains vary substantially across models and domains. FinEvo-Bench (Deng et al., 2026a) provides a longitudinal benchmark for self-evolving agents in professional financial workflows. Social Gym (He et al., 2026) benchmarks multi-agent social reasoning in games with rule-verifiable outcomes. AI4AI-Bench (Chi et al., 2026) isolates the design of training algorithms for recursive self-improvement. Beyond benchmarks, recent analyses reveal that self-improvement itself is often unreliable: memory-based methods exhibit high variance and strong sensitivity to task order (Ye et al., 2026). Accumulated skills can contaminate later behavior once a defective skill enters the pool (Shang et al., 2026). Optimizer-driven agent gains need not compound across continual-learning rounds (Wang et al., 2026a). These benchmarks and analyses represent important steps toward evaluating adaptive systems, but they focus on different improvement artifacts or specific aspects of the adaptation process. S^3Gym jointly records self-judgments and verifier outcomes during experience-driven adaptation. Table 7 summarizes the key differences between existing benchmarks and S^3Gym.

S^3Gym complements these benchmarks by recording both model-generated judgments and executable outcomes during experience reuse. It compares context-level representations under matched interaction budgets and tests updated agents on disjoint, generally stricter configurations. These records support separate analyses of judgment reliability and subsequent performance. Their association helps characterize how local judgment quality relates to subsequent gains.

Table 7: Adaptive-agent benchmarks and their principal evaluation targets. The entries describe their focus rather than asserting that a capability is absent.

Benchmark	**Experience use or updated artifact**	**Evaluation focus**
PostTrainBench	Parameter post-training	Held-out task performance
SEA-Eval	Sequential agent adaptation	Long-term evolution
SEAGym	Prompts, memories, tools, and workflows	Held-out ID/OOD transfer and replay
PAST-Bench	Retained cross-session experience	Matched memory on/off conditions
ContinualSkillBench	Skill-library accumulation	Cross-task reuse
FinEvo-Bench	Longitudinal agent experience	Financial task sequences
Social Gym	Tournament self-play	Rule-verifiable social reasoning
S^3Gym	History ICL, Summary Memory, and auxiliary parameter training	Strict held-out scores and step-level self-judgment alignment

G Game Prompt Templates

This appendix presents condensed versions of the game prompts used in the main experiments, based on `test/S3GYM/envs/*/prompt.py`. We abridge the wording and replace episode-specific values with angle-bracket placeholders (e.g., `<BOARD>`); the game rules, action spaces, coordinate conventions, and required answer syntax follow the implementation. At inference time, the game-specific system prompt is concatenated with the current step prompt and, for memory-based agents, the available *Hint Messages*.

G.1 Shared Interaction Protocol

Six games prepend the shared base instruction described below, whereas Trust Evolution uses its own game-specific system instruction. Both memory conditions retain the same game rules, dynamic

state, legal action space, and self-judging contract. They differ in history representation, prompt composition, and compression calls, as detailed below.

History ICL: raw-history in-context learning

History ICL serializes every retained step from previous episodes as an environment–agent pair, `Env: <STATE>` followed by `LLM: Answer: <ACTION>, Score: <SELF-SCORE>`. At action time, the model receives the following composition:

```
Hint Messages:
<RAW HISTORY FROM PREVIOUS EPISODES>

Game history of current episode:
<RAW HISTORY FROM EARLIER STEPS IN THIS EPISODE>

<GAME SYSTEM PROMPT + CURRENT STEP PROMPT>
```

The implementation preserves the current-episode history and current game prompt in full, while truncating the tail of the previous-episode history when the context budget is exceeded.

Summary Memory: score-conditioned strategy compression

At step 0 of each episode, Summary Memory first makes a separate compression call over the game rule and truncated previous-episode history:

```
Rule:
<GAME SYSTEM PROMPT>

History:
<TRUNCATED RAW HISTORY FROM PREVIOUS EPISODES>

Please summarize practical experience and recommendations
...
Tips: <BULLET-POINTED STRATEGY SUMMARY>
```

The extracted `Tips` are cached for the session and reused at later steps. The action-time input is then

```
Hint Messages:
<SUMMARIZED TIPS>

<GAME SYSTEM PROMPT>

Game history of current episode:
<RAW HISTORY FROM EARLIER STEPS IN THIS EPISODE>

<GAME SYSTEM PROMPT + CURRENT STEP PROMPT>
```

Thus, only cross-episode experience is compressed; the current episode remains available as raw state–action–score history. In the implementation described here, the game system prompt appears once with the cached summary prefix and again with the current step prompt. This prompt duplication and the extra compression call are implementation differences between the pathways.

Regardless of memory condition, the model must report both an action and its action-level self-score. The shared final-line contract is

Shared response contract

`Answer: <ACTION>, Score: <SELF-JUDGED SCORE>.`
If the agent determines that the episode has terminated, it returns `Answer: END, Score: 0.`

The self-score is intended to estimate the reward associated with the proposed action, rather than free-form confidence. However, some game instructions describe cumulative or terminal scores:

Minesweeper specifies the correctly flagged mine ratio, and PvZ specifies survived turns. This ambiguity can affect agreement with step rewards, so the reliability analysis should not be interpreted as a pure test of judgment ability. Table 8 summarizes the observation and action interfaces before the condensed game templates are presented.

Table 8: Observation and action interfaces exposed by the canonical game prompts.

Game	Main dynamic observation	Valid action form
Chess	$N \times N$ board with labeled pieces and empty cells	Coordinates for one or more labeled pieces
Minesweeper	Partially revealed grid with numbers, flags, and unknown cells	`uncover`, `flag`, or `unflag`
Nullify	Indexed list of signed values and arithmetic operators	Two unit indices
Tetris	Fixed-cell board and next-block rendering	Drop column and rotation angle
Snake	Grid containing walls, head, body, food, and obstacles	`LEFT/RIGHT/UP/DOWN`
Plants-vs-Zombies	Battlefield, sun budget, plants, zombies, and turn state	Zero or more plant placements
Trust Evolution	Interaction round and action/payoff history	`collaborate` or `cheat`

G.2 State Inference and Deduction

Chess-style state prediction

System instruction. You are playing a step-by-step piece-position prediction game on an $N \times N$ board. Empty cells are denoted by `.`, pieces are denoted by letters, coordinates are zero-based, and every piece follows its own hidden movement rule. Predict the next position of as many pieces as possible from the observed board-state sequence.
Dynamic state. `Current Game State:` followed by `<BOARD>`.
Decision instruction. Predict the next-step location of each recoverable piece. Use the syntax `A:(x,y), b:(x,y), c:(x,y), ...`. The agent may return `END` when it elects to stop.

```
Answer: A:(<ROW>,<COL>), b:(<ROW>,<COL>), ..., Score:
<SELF-JUDGED SCORE>.
```

Minesweeper

System instruction. You are playing Minesweeper on a `<BOARD_ROWS>` × `<BOARD_COLS>` board with `<MINES>` hidden mines. Unknown cells are `?`, flags are `F`, revealed counts are `0-8`, and revealed mines are `X`. Coordinates are zero-based. Win by uncovering every safe cell or correctly flagging every mine; the episode ends at `<MAX_EPOCHS>` or when a mine is uncovered, depending on the evaluation mode.
Dynamic state. `Current Game State:` followed by the partially observed `<BOARD>`.
Decision instruction. Choose exactly one of `uncover (r,c)`, `flag (r,c)`, or `unflag (r,c)`. The score is the ratio of correctly flagged mines to the total number of mines.

```
Answer: uncover (<ROW>,<COL>), Score: <SELF-JUDGED SCORE>.
```

G.3 Symbolic and Spatial Planning

Nullify

System instruction. The board contains indexed independent units. Signed numbers (`+n` or `−n`) can be combined with another unit; available operators include multiplication, division, square root, square, reciprocal, floor, and ceiling. Combining equal-magnitude values with opposite signs produces zero and removes both units. The objective is to eliminate all units, yielding a final value of zero.
Dynamic state. `Current game board:` followed by one `index,content` pair per unit in `<INDEXED UNITS>`. Unit indices start at zero and are updated after each combination.
Decision instruction. Select the two indices to combine on the current turn. A successful complete cancellation receives score 1; otherwise the terminal score is 0.

```
Answer: <INDEX I> <INDEX J>, Score: <SELF-JUDGED SCORE>.
```

Tetris

System instruction. You are playing simplified Tetris on a `<BOARD_WIDTH>` × `<BOARD_HEIGHT>` board. The board shows fixed blocks as `*` and empty cells as `.`; the next-block panel uses `#`. A placed block falls until obstructed, each completed row is cleared for one point, and the episode terminates when a falling block extends above the board.
Dynamic state. `Current Game State:` `<BOARD>` and `Next Block:` `<NEXT_BLOCK>`.
Decision instruction. Choose the one-based drop column of the block's leftmost cell and a clockwise rotation in {0, 90, 180, 270}. The objective is to maximize cleared rows.

```
Answer: <DROP COLUMN> <ROTATION ANGLE>, Score:
<SELF-JUDGED SCORE>.
```

G.4 Reactive Control and Strategic Interaction

Snake

System instruction. You control a snake on a `<GRID_WIDTH>` × `<GRID_HEIGHT>` grid. Walls are `#`, the head is `H`, body cells are `S`, apples are `A`, and obstacles are `X`. Eating an apple increases both length and score by one. A wall, self, obstacle, or invalid-action collision terminates the game under strict rules. The snake cannot reverse direction and the episode is capped at `<MAX_STEPS>`.
Dynamic state. `Current Game State:` followed by `<BOARD>` and the current movement direction.
Decision instruction. Choose exactly one next movement from `LEFT`, `RIGHT`, `UP`, or `DOWN`.

```
Answer: <LEFT|RIGHT|UP|DOWN>, Score: <SELF-JUDGED SCORE>.
```

Plants-vs-Zombies

System instruction. You are playing simplified Plants-vs-Zombies on a `<ROWS>` × `<COLS>` grid. Each turn gains at least 25 sun; zombies spawn every five turns and move left, becoming stronger over time. Available plants are Sunflower (`X`), Peashooter (`W`), Three-Line Shooter (`S`), Wall-nut (`J`), Torch Stump (`H`), and Fire Chili (`F`), each with the costs and effects specified in the system prompt. The score increases by one per survived turn; the game ends when a zombie reaches column −1 or after `<MAX_STEPS>`.
Dynamic state. `Current Game State:` followed by `<BATTLEFIELD>`, including the turn, sun budget, plant layout, and zombie states.
Decision instruction. Place zero or more plants for the next turn. Each placement is `PlantType Row Col`; multiple placements are separated by semicolons. Rows and columns are zero-based, and an empty action is allowed.

```
Answer: X <ROW> <COL>; W <ROW> <COL>, Score: <SELF-JUDGED
SCORE>.
```

Trust Evolution

System instruction. You play a fixed-length repeated interaction against an opponent with a stable within-episode strategy. The two actions are `collaborate` and `cheat`. Payoffs to the agent are $+2$ for mutual collaboration, -1 for collaborating against cheating, $+3$ for cheating against collaboration, and 0 for mutual cheating.
Dynamic state. `Current Game State:` followed by `<ROUND AND INTERACTION HISTORY>` for an episode of `<MAX_STEPS>` rounds.
Decision instruction. Infer the opponent's behavior from the observed interaction and select exactly one action for the next round.

```
Answer: <collaborate|cheat>, Score: <SELF-JUDGED SCORE>.
```

H CONTEXT-LEVEL EVALUATION TRAJECTORIES

Each row compares History ICL (left) and Summary Memory (right) on the same game and vertical scale. Each checkpoint averages three strict-mode evaluation episodes; the horizontal axis counts exploration episodes.

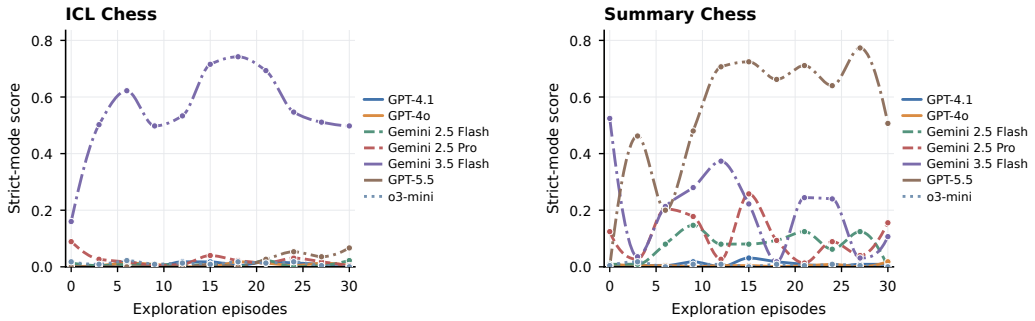


Figure 4: Chess: History ICL (left) and Summary Memory (right).

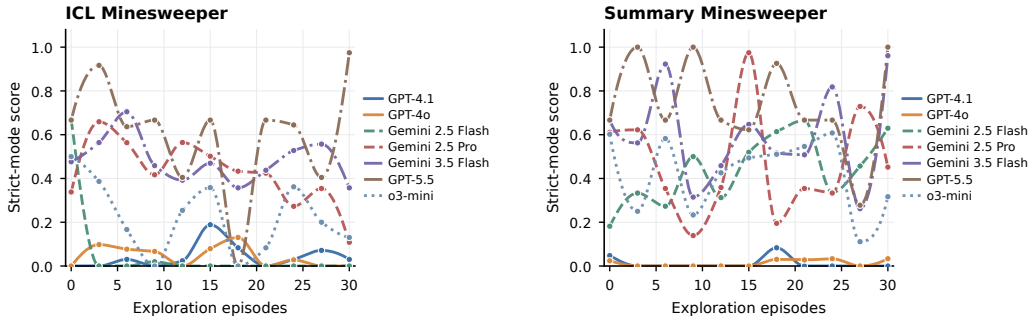


Figure 5: Minesweeper: History ICL (left) and Summary Memory (right).

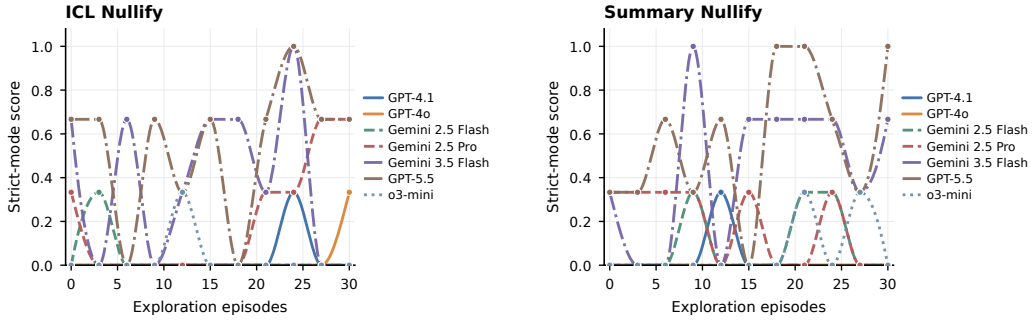


Figure 6: Nullify: History ICL (left) and Summary Memory (right).

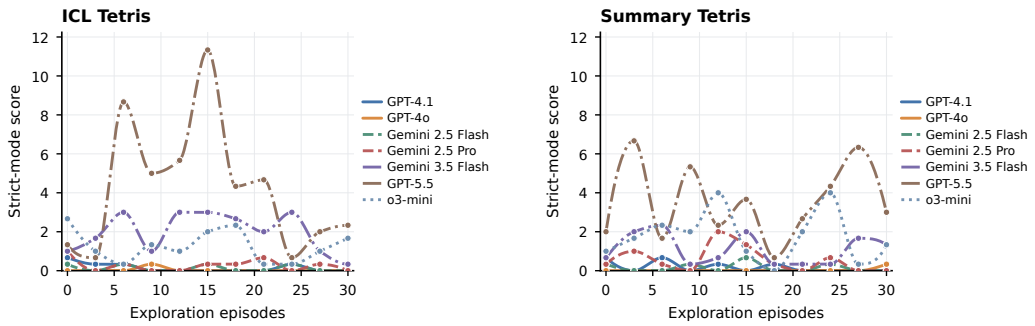


Figure 7: Tetris: History ICL (left) and Summary Memory (right).

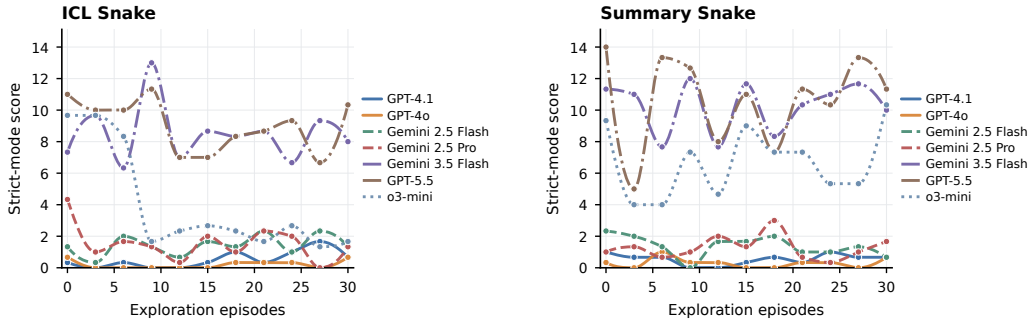


Figure 8: Snake: History ICL (left) and Summary Memory (right).

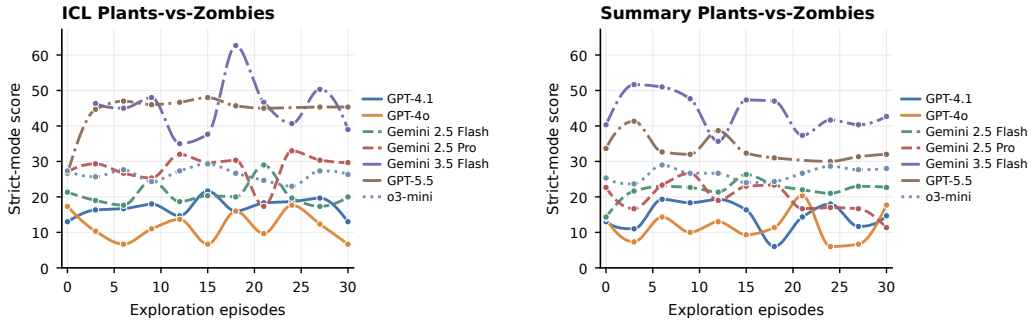


Figure 9: Plants-vs-Zombies: History ICL (left) and Summary Memory (right).

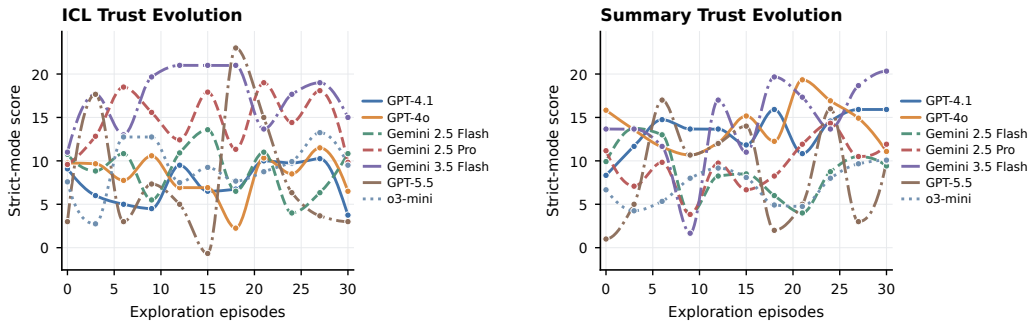


Figure 10: Trust Evolution: History ICL (left) and Summary Memory (right).