LIFT-BENCH: EVALUATING LEARNING INVESTMENT FOR FUTURE TRANSFER IN LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language agents can improve through interaction and persistent memory, but these gains incur reasoning and learning costs. As agents acquire reusable knowledge for diverse future tasks, they must decide what to learn under limited resources. Existing benchmarks typically emphasize task performance or per-task cost, providing limited support for jointly evaluating long-term learning gains and resource expenditure from a global perspective. We introduce **LIFT-Bench** (*Learning Investment for Future Transfer*) to evaluate allocation of a finite global token budget across competing learning opportunities. It contains 690 learning resources and 329 evaluation tasks across eight domains spanning code, tool use, mathematical reasoning, and local physical laws, organized by reusable knowledge units with explicit support relations and domain-specific verification. Inspection, study, practice, and memory construction share one learning budget; transfer is then evaluated with frozen memory and separate per-task inference allowances. A unified interface supports comparisons across three models and eight frameworks using domain-macro performance and return on learning-token investment. We evaluate three policies: Selective Investment for Future Transfer (SIFT), sequential learning, and no learning. Results distinguish the transfer value of learning expenditures and show that larger budgets need not improve future-task performance. LIFT-Bench measures what agents should learn, how they allocate learning budgets, and whether acquired knowledge benefits future tasks.

1 INTRODUCTION

As language agents evolve from task-bound solvers toward long-lived systems that operate across diverse and evolving environments, learning itself becomes an increasingly important part of agent behavior.An agent that will solve many future tasks faces a decision before any one task is assigned: *which knowledge is worth acquiring now?* Reading an API manual, practicing a mathematical method, and studying an unfamiliar code library all consume resources. Their value depends on whether the acquired knowledge helps with later tasks and whether a different use of the same budget would have been more useful. Evaluating this decision requires making the choice of learning experience part of the benchmark.

Cost-aware evaluation has established that accuracy alone is an incomplete measure of agent quality (Kapoor et al., 2025). The cost of solving an assigned task, however, addresses a different question from the allocation of expenditure before future tasks are revealed. Reflection and lifelong-learning methods enable agents to retain experience across interactions (Shinn et al., 2023; Zheng et al., 2025a). This creates a further evaluation problem: given more learning opportunities than an agent can afford, can it select experience whose future benefit justifies its cost? Measuring only final task success leaves the cost and quality of those learning choices unresolved.

We introduce LIFT-Bench (*Learning Investment for Future Transfer*), a benchmark that turns this learning-allocation problem into an executable agent environment. Before evaluation tasks are revealed, an agent explores a pool of learning opportunities, selectively studies resources, and constructs persistent memory under a single global token budget. The agent controls what to inspect and learn as well as when to stop. Once learning ends, memory is frozen and transfer is evaluated on fresh tasks.

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.

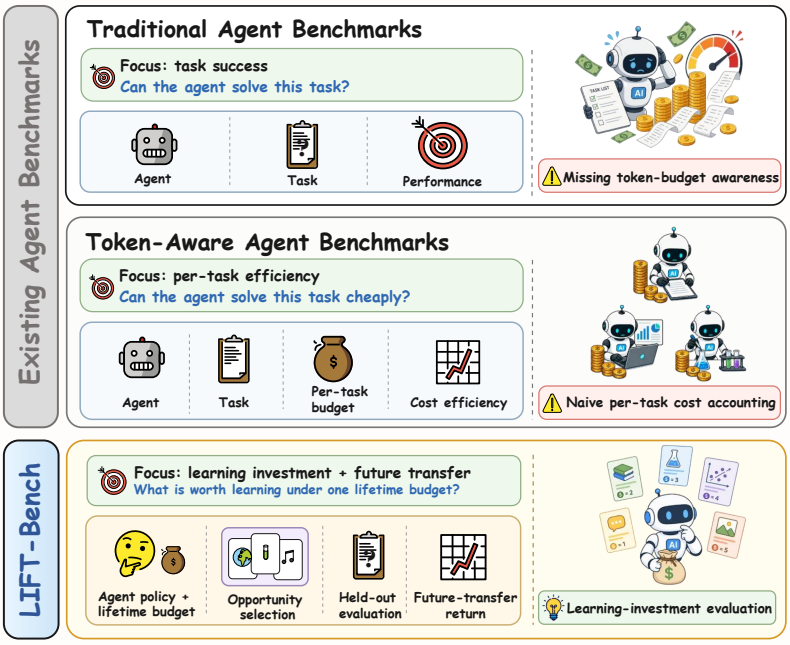


Figure 1: Evaluation objectives in agent benchmarks: task correctness, per-task efficiency, and learning investment for future transfer. LIFT-Bench evaluates resource selection under a shared learning budget before future tasks are revealed.

LIFT-Bench contains 690 learning opportunities and 329 evaluation tasks spanning eight domains, including code, tool use, mathematical reasoning, and local physical laws. Resources and tasks are organized around reusable knowledge units with explicit support relations and domain-specific verifiers. The mixed opportunity pool forces agents to trade off learning across domains rather than receiving a prescribed curriculum or domain-specific budget.

We evaluate three policies—Selective Investment for Future Transfer (SIFT), sequential learning, and no learning—across multiple models, frameworks, and learning budgets. The protocol exposes both the downstream benefit and the acquisition cost of learned knowledge, allowing us to measure absolute transfer performance together with return on learning-token investment.

Our contributions are threefold. (1) We introduce LIFT-Bench, to the best of our knowledge the first benchmark to evaluate how agents allocate a finite global learning budget across competing resources before future tasks are revealed. (2) We construct a heterogeneous learning and evaluation pool with explicit resource–task support, and an executable protocol that meters acquisition costs and measures transfer with frozen memory. (3) We evaluate three learning conditions across models, frameworks, and budgets, showing that allocation gains depend on the model and framework, broader coverage need not yield better transfer, and higher performance can coexist with lower return per learning token.

2 Related Work

Task performance and per-task efficiency. Agent evaluation has examined both task competence and execution cost. AgentBench evaluates reasoning and decision-making across interactive environments (Liu et al., 2024), while WebArena measures functional success on realistic web tasks (Zhou et al., 2024). Kapoor et al. (2025) argue for evaluating accuracy jointly with cost. These perspectives assess task execution but leave a separate question: which learning resources should an agent acquire, at what cost, before future tasks are revealed?

Persistent learning and memory. Persistent learning connects earlier experience to later behavior. Reflexion retains verbal feedback across trials (Shinn et al., 2023), and Voyager combines an automatic

curriculum with a reusable skill library (Wang et al., 2024). LifelongAgentBench and MemoryArena evaluate learning and memory use across interdependent tasks (Zheng et al., 2025a; He et al., 2026). SkillFlow studies skill discovery and revision across task sequences (Zhang et al., 2026), while StreamMemBench examines evidence use and feedback reuse in subsequent assistance (Liu et al., 2026). These works motivate evaluating useful experience rather than memory retention alone.

Knowledge-dependent evaluation. Knowledge-dependent benchmarks help distinguish missing information from execution failures. SE-Bench probes knowledge internalization through a renamed API library (Yuan et al., 2026); BFCL and API-Bank evaluate tool use (Patil et al., 2025; Li et al., 2023); and BigCodeBench evaluates library-rich programming (Zhuo et al., 2025). TheoremQA, Reasoning Gym, and NewtonBench provide complementary settings for scientific, procedural, and local-law reasoning (Chen et al., 2023; Stojanovski et al., 2025; Zheng et al., 2025b). We adapt resources from these settings while retaining domain-specific verification contracts.

Global learning investment for future transfer. Resource allocation also arises during task execution: LIMBO jointly selects memory strategies and inference budgets for incoming tasks (Sharma et al., 2026). LIFT-Bench examines allocation before future evaluation tasks are revealed. Inspection, selection, practice, and memory construction compete for one global learning budget across heterogeneous resources; subsequent evaluation freezes memory and uses a separate inference allowance. This separation makes the acquisition cost and future utility of learning choices jointly measurable.

3 Learning Investment as an Evaluation Problem

We formulate learning investment as a two-stage evaluation problem: an agent first allocates a shared budget to knowledge acquisition, then applies its frozen memory to previously unrevealed tasks. This section defines the agent's observable state and actions, token accounting, the learning–evaluation boundary, and the metrics used to assess transfer and efficiency.

World and state. A benchmark instance is $\mathcal{W} = (\mathcal{L}, \mathcal{T}, \mathcal{V}, \mathcal{G})$: a public learning pool, a private evaluation set, task-specific verifiers, and a support relation linking tasks to relevant knowledge. A resource has a public description and protected execution metadata. The public description contains an opaque opportunity identifier, an interaction type, a family hint, skill tags, and a natural-language brief. Source routing, evaluation roles, targets, and verifier internals remain private. At decision step t, the state is

$$s_t = (b_t, \mathcal{M}_t, h_t, \kappa_t), \qquad a_t \sim \pi(\cdot \mid s_t, \mathrm{observe}(\mathcal{L})), \tag{1}$$

where b_t is the remaining learning budget, $\mathcal{M}_t$ is persistent memory, h_t is the observed interaction history, and κ_t summarizes coverage and failed attempts. An action may inspect a resource, authorize an investment horizon, study, attempt a learning task, write memory, or freeze. The learning phase terminates when the agent freezes or the next admissible request cannot fit the remaining budget.

One budget, including the cost of choosing. Let $c(a_t)$ denote gateway-metered input-plus-output tokens. Every model call, including framework manager and reviewer calls, draws from the same phase budget:

$$C_L = \sum_{t<\tau} c(a_t) \le B_L, \qquad b_{t+1} = b_t - c(a_t). \tag{2}$$

Repeated context and planning calls are charged; verifier computation is logged as execution time, not tokens. The cross-domain budget permits concentration, diversification, repetition, or early stopping, with no additional domain-specific allowance.

Freeze and evaluation. At τ, the runner hashes $\mathcal{M}_\tau$ and rejects further writes. Each $x \in \mathcal{T}$ receives a fresh session, task context, bounded frozen-memory retrieval, and inference cap B_T, without reference answers or feedback for learning. All tasks are evaluated regardless of remaining learning budget. Scores are $v_x(\hat{y}_x) \in [0, 1]$; a terminal outcome need not be correct. Infrastructure failures and unanswered tasks remain distinct statuses.

Table 1: Composition of the learning and evaluation pool. Units denotes reusable knowledge units within each domain; multiple resources may support one unit. Learning resources cover every evaluation family and knowledge unit (Appendix A).

Domain	Learning	Test	Units
API schemas	30	30	30
Library programming	77	26	26
Unfamiliar APIs	63	58	58
Procedural reasoning	58	25	11
Scientific problem solving	96	50	50
Theorem transfer	142	60	14
Local physical laws	154	50	50
Dialogue-grounded calls	70	30	19
Total	690	329	258

Evaluation metrics. We report *domain-macro score* (equal-weight domain performance), *return on learning-token investment (RoTI)* (gain over no learning per actual learning token), and *budget-averaged score* (normalized area under the budget–score curve, AUC). With D domains and $\mathcal{T}_d$ the tasks in domain d, we report

$$S_d(\pi, B_L) = \frac{1}{|\mathcal{T}_d|} \sum_{x \in \mathcal{T}_d} v_x(\hat{y}_x), \qquad S(\pi, B_L) = \frac{1}{D} \sum_{d=1}^{D} S_d(\pi, B_L). \tag{3}$$

Domains have equal weight regardless of task count, not equal semantic difficulty. Relative to protocol-matched no learning π_0,

$$\Delta S = S(\pi, B_L) - S(\pi_0, 0), \qquad \mathrm{RoTI} = \frac{\Delta S}{C_L}, \quad C_L > 0. \tag{4}$$

RoTI is undefined at zero learning cost and reported in score percentage points per 100k learning tokens. Trapezoidal AUC divided by the measured budget range yields the budget-averaged score. Neither replaces absolute performance. Support-conditioned task and memory logs refine transfer analysis; cross-source retrieval alone does not establish causal transfer.

4 Constructing the Opportunity Pool

The opportunity pool makes different kinds of reusable knowledge compete for the same learning budget. We construct it by adapting source tasks into learning resources and evaluation instances, linking them through knowledge units, and preserving domain-specific verification contracts. The following paragraphs describe the pool's organization, domain conversions, and quality controls.

Instance and granularity. The snapshot contains 1,019 records: 690 learning opportunities and 329 test tasks across eight domains, 65 domain-local families, and 258 knowledge-unit identifiers (Table 1). A *domain* defines interaction and scoring; a *family* groups related activities; a *knowledge unit* identifies reusable content (an API schema, theorem, or local law); and a *template* defines task transformations or response patterns. Two instances of one theorem remain one unit. Units describe content, not equal difficulty or breadth.

Code and tools: from task execution to learning resources. We use selected BigCodeBench problems for library programming and SE-Bench resources for unfamiliar APIs. Library resources provide executable context, worked code, or focused interface primers; evaluation uses unit-test fractions in a restricted process. The unfamiliar-API domain pairs documentation for the renamed `zwc` library with coding tasks. Its present verifier measures functional output and logs API usage separately; functional success alone does not certify that an agent used a learned API. BFCL-derived resources instead teach a function schema that is withheld from the corresponding closed-book call task. API-Bank-derived resources pair documentation and a dialogue prefix with a next-call target. Earlier tool observations remain available when present in that prefix, but future turns are excluded. The latter two domains use typed call matching, not live API side effects.

Mathematics: transferring a method, not a renamed answer. The theorem domain uses seeded, exact-answer generators covering 14 knowledge units, including reflection counting, Euler's totient, the determinant lemma, and absorbing random walks. A teaching resource states applicability conditions, a derivation or solution method, a worked instance, and a misuse counterexample. Evaluation changes the instance while preserving a specified mathematical dependency. For example, a reflection-principle resource explains why nonnegative balanced paths of length $2n$ number $\binom{2n}{n} - \binom{2n}{n-1}$, including why merely returning to the origin is insufficient. The learner must retain the condition as well as the formula. TheoremQA-derived scientific problems and fixed-seed Reasoning Gym tasks broaden the setting to source-grounded and procedural reasoning. They retain distinct answer contracts rather than being relabeled as the same type of mathematical evidence.

Physics: explicit, coherent local laws. NewtonBench-derived resources specify a local law, observations, and applicability guards. A local world may use $F = C(m_1 m_2/r)^2$ for positive inputs and zero otherwise. A transfer query asks for a ratio of readings under new settings, so the common constant cancels while the altered exponents remain essential. The prompt explicitly identifies a closed simulation world and its native numerical convention. The task is to learn and apply a documented rule, not to rediscover a hidden physical law or assert its validity in SI units. Scalar replay checks the stated rule; it does not certify the original interactive simulator.

Conversion and quality controls. Records retain source IDs, transformation metadata, execution contracts, and protected verification material. Conversion combines scripted extraction and grouping with AI-assisted teaching and repair. Repeated opportunities expose redundancy-sensitive allocation, not independent skills. All 1,019 records pass structural checks; resources support all 65 test families and 245 test knowledge units. Appendix B separates reference corrections from precision normalization and provides counts and examples. The 329 test tasks remain unavailable until frozen evaluation. Content validity and train–test isolation are assessed separately.

5 Policies and Shared Execution

We define three evaluation policies to distinguish the value of autonomous allocation from the value of learning itself. Under SIFT, the agent requests resource information, estimates utility and cost, and authorizes token expenditure; a shared controller executes these decisions under the budget. Sequential learning provides the same learning tools and budget under a fixed resource order; no learning measures the solver without acquired memory. Frameworks execute all three conditions through the common interface.

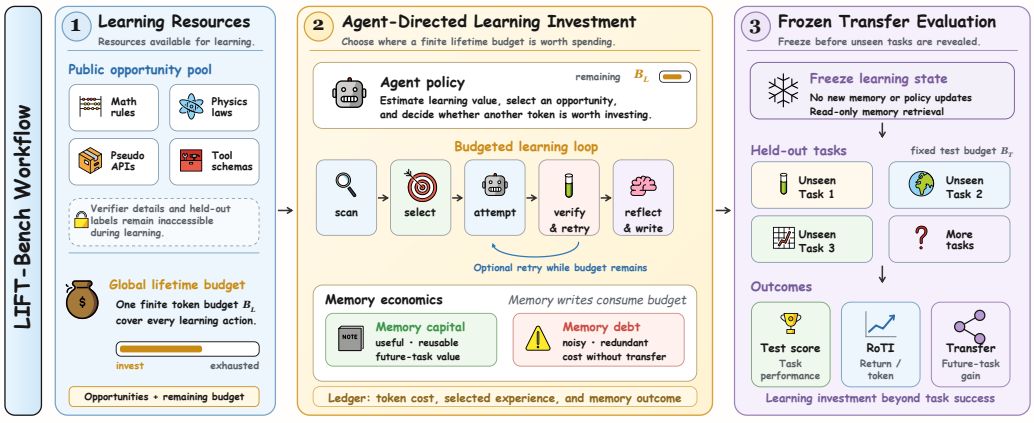


Figure 2: The LIFT-Bench workflow: select and study resources under one learning budget, freeze the resulting memory, and evaluate transfer with a separate task budget. The shared controller meters learning expenditure and routes tasks to domain-specific verifiers. Icons illustrate representative resource types.

Selective Investment for Future Transfer (SIFT). SIFT uses a global family index and on-demand cards rather than placing every resource description in every prompt. The agent estimates value, cost, and risk for selected families, and authorizes a token-denominated investment horizon. The Python controller ranks unprocessed opportunities, executes study episodes within that horizon, and then returns updated budget, memory, and coverage information for replanning. Uninspected families remain eligible through a default prior. For an opportunity associated with a public knowledge group studied n times, define novelty $\nu = (1+n)^{-1}$. The implemented heuristic is

$$q(o) = \frac{\hat{v}(o)\nu + u(o,n) + 0.10\nu - 0.20\hat{r}(o)}{\hat{c}(o)}. \tag{5}$$

Here u is an uncertainty allowance and $\hat{c}$ blends structural, agent-estimated, and observed costs (Appendix C). This is a scheduling heuristic, not a calibrated estimate of causal learning gain. The agent controls what information to request and whether to continue; the controller enforces admissible actions, ranking, and budgets. Studying can expose documentation and learning-only feedback, followed by up to two practice attempts and a bounded memory write. No memory-gate variant is included in these comparisons.

Sequential learning. SEQUENTIAL follows a fixed curriculum instead of adaptively ranking resources. A seed-0 hash orders public resource groups and the items within each group; related items are studied consecutively before advancing to the next group. Groups are defined by public interface, family, skill tags, and explicit world identifiers where applicable. The baseline uses the same study, practice, and memory tools as SIFT and stops at the same global budget boundary. This comparison isolates adaptive investment from a reproducible, redundancy-preserving curriculum; it does not compare against random-order or skill-deduplicated sampling.

No learning. No learning skips acquisition and uses an empty persistent store. Each model–framework pair has a baseline matched to its test protocol, measuring learning gains over the same solver without acquired memory, rather than differences between learning policies. Learning cost is zero, so RoTI is undefined.

Framework interface. Native framework loops issue actions through a common interface; the host executes resources or verifiers and returns observations. One gateway meters all calls, including nested calls, and one store holds persistent memory. Both learning policies have identical frozen-memory retrieval: at most three items and 600 tokens per test task (Appendix C; Figure 2).

6 EXPERIMENTS

Design and comparability. Experiments follow the two-stage protocol in Section 3: agents first spend a global learning budget to selectively acquire knowledge and write persistent memory. After learning, the memory is frozen, and evaluation is performed on fresh tasks with no further learning or memory updates. Domain-macro score equally averages domain-level task scores; return on learning-token investment (RoTI) measures gain over matched no learning per 100k actual learning tokens, in percentage points. The *budget study* evaluates Qwen3.8-27B–OpenClaw at 17 budgets (50k–650k), with a 5k-token test cap, matched no-learning run, and budget-averaged score. The *framework matrix* crosses three models, eight frameworks, three budgets (50k, 150k, 300k), and two learning policies: 144 learning runs plus 24 model–framework-specific no-learning baselines, with a 9k-token test cap (Table 5). Different settings can produce different results for the same model and framework.

The models are Qwen3.8-27B (Qwen Team, 2026), DeepSeek-V4.1-Flash (DS; DeepSeek, 2026), and GLM-5.3-Flash (GLM; Z.ai, 2026). We compare OpenClaw, OpenHands, Deep Agents, mini-SWE-agent, smolagents, CrewAI, LangGraph, and AutoGen through a common world interface. Appendix D specifies the restricted native configurations. Aggregate differences below are computed from the reported two-decimal scores.

6.1 MAIN RESULTS: LEARNING VALUE DEPENDS ON SELECTION AND EXECUTION

Table 2 separates the benefit of selection from the benefit of learning itself. SIFT exceeds SEQUENTIAL in 13/24 Qwen, 18/24 DS, and 24/24 GLM comparisons, with mean differences of -0.48, $+0.91$,

Table 2: Domain-macro scores (%) for three models and eight frameworks. Each row reports one no-learning baseline and two learning policies at three budgets (168 scores in total). Bold marks the better learning policy within each budget. Seq. denotes SEQUENTIAL.

	No learning	50k		150k		300k	
Framework		Seq.	SIFT	Seq.	SIFT	Seq.	SIFT
Qwen3.8-27B							
OpenClaw	24.61	23.71	**23.97**	**26.11**	24.56	**26.03**	25.49
OpenHands	20.49	20.99	**21.81**	21.23	**21.52**	**22.99**	20.83
Deep Agents	23.99	24.63	**25.63**	23.90	**25.83**	**27.18**	25.00
mini-SWE-agent	34.38	33.43	**33.77**	30.90	**34.58**	**33.77**	31.46
smolagents	22.29	26.08	**26.71**	24.13	**25.59**	**25.43**	25.27
CrewAI	33.20	33.69	**34.50**	34.71	**36.09**	**34.71**	34.38
LangGraph	26.57	**26.75**	24.36	**27.99**	23.46	**28.79**	23.32
AutoGen	26.85	28.09	**28.71**	26.21	**26.36**	**27.62**	24.44
DeepSeek-V4.1-Flash							
OpenClaw	38.19	34.45	**39.64**	35.49	**39.66**	36.81	**46.52**
OpenHands	26.63	24.59	**27.02**	26.54	**27.35**	**27.89**	27.22
Deep Agents	30.15	31.47	**31.76**	30.28	**30.32**	30.23	**30.34**
mini-SWE-agent	31.83	31.64	**33.39**	**34.65**	34.57	35.11	**37.69**
smolagents	20.90	19.35	**19.80**	22.15	**22.34**	**23.15**	23.03
CrewAI	29.16	32.09	**32.33**	33.92	**33.99**	31.73	**35.76**
LangGraph	20.38	**18.04**	16.95	**17.26**	14.39	**19.86**	10.67
AutoGen	29.34	30.26	**31.22**	30.50	**32.25**	29.21	**30.19**
GLM-5.3-Flash							
OpenClaw	36.21	36.51	**40.02**	37.83	**42.21**	38.55	**45.53**
OpenHands	32.48	33.64	**34.02**	34.39	**34.48**	33.04	**33.81**
Deep Agents	31.41	32.52	**34.33**	35.56	**36.73**	35.75	**36.19**
mini-SWE-agent	35.99	35.12	**38.88**	36.31	**36.49**	38.16	**38.51**
smolagents	23.08	21.88	**24.33**	23.23	**26.67**	24.51	**27.89**
CrewAI	34.51	35.02	**37.23**	36.12	**38.55**	37.03	**41.11**
LangGraph	36.49	33.36	**33.84**	37.71	**37.81**	38.49	**44.02**
AutoGen	31.97	32.49	**34.11**	33.15	**35.37**	33.84	**36.29**

and $+2.26$ percentage points, respectively. The same policy therefore has different value across model–framework configurations.

Memory debt: learning does not guarantee positive transfer. At 50k, 16 of 48 learning runs score below their matched no-learning baseline (five Qwen, seven DS, and four GLM runs). Limited learning can thus accompany a net loss. Disproportionate reliance on a small, incomplete set of retained skills can interfere with an agent's existing capabilities.

The Qwen–LangGraph 300k SIFT case makes the application problem more concrete. Its theorem score falls from 43.33% without learning to 20.00%. Among 51 theorem tasks that retrieve memory, 13 change from correct to incorrect and none from incorrect to correct. In an interior lattice-point problem, the no-learning agent calculates and cross-checks the correct answer, 162. After learning, the agent retrieves the correct Pick's theorem but submits 168. Correct source knowledge and relevant retrieval therefore need not yield correct use. These observations motivate studying memory-induced application errors as a form of *memory debt*; they do not establish that retrieval alone caused each regression, because evaluation trajectories also differ.

Larger budgets can amplify allocation errors. At 300k, Qwen–SIFT scores below SEQUENTIAL in all eight frameworks, with an average difference of -2.04 points. OpenClaw likewise decreases from 26.03% to 25.49%. The reported Qwen traces devote approximately 24% of the learning budget to browsing and planning on average, while allocating many opportunities to library programming and scientific problem solving without commensurate effective transfer. This pattern is consistent with inaccurate value estimates consuming both deliberation tokens and opportunities to address other knowledge gaps. A larger budget can expand the consequences of an allocation error without improving the estimates that drive it.

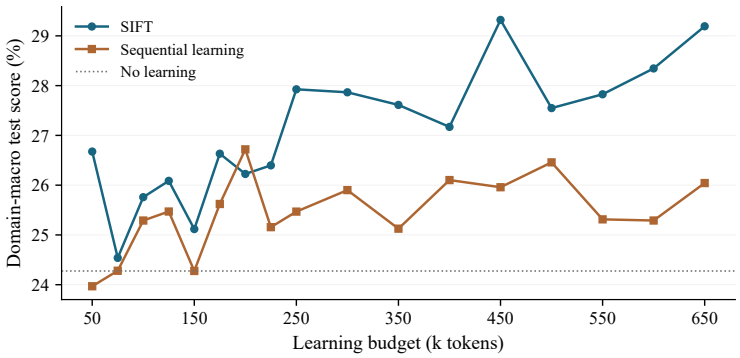


Figure 3: Budget–performance curve for Qwen3.8-27B with OpenClaw under the 5k-per-task protocol. Points are complete runs at discrete learning budgets; the dotted line is the matched no-learning score.

Selective learning benefits from addressable information gaps. DS–OpenClaw at 300k increases from 38.19% without learning to 46.52% with SIFT. It studies 35 opportunities, only one of which concerns difficult mathematical or scientific problem solving. Difficult mathematics may exceed the base model's reasoning capacity even after it acquires the relevant theorem, whereas some API and unfamiliar-code tasks benefit from knowledge that can be retained and applied directly to improve scores. In a criminal-record query, the no-learning agent invents a `record_type` argument; after learning, it follows the interface contract by using `details=true` and passes the verifier. Acquiring an environment-specific contract can close an information gap while model parameters remain fixed. By contrast, a retrieved theorem still requires recognizing its conditions, deriving the relevant expression, and computing accurately. The evidence distinguishes missing knowledge from a reasoning or execution bottleneck: supplying a skill cannot guarantee that the base model can apply it to a difficult task.

Model–framework interactions regulate policy gains. At 300k, the displayed DS scores give SIFT gains over SEQUENTIAL of 9.71 points in OpenClaw, 4.03 in CrewAI, and 2.58 in mini-SWE-agent, but a loss of 9.19 in LangGraph. The evaluated LangGraph adapter is a simplified model–tool loop that terminates when no tool call is emitted. In a PCA task, DS generates code as ordinary text without invoking a tool; its 300k SIFT run yields no execution results for 209/329 tasks. Under the same graph structure, GLM yields no execution results for only 16/329 tasks and reaches 44.02%. These cases show how termination rules and action expression interact with model tool behavior. Role organization and orchestration also differ across frameworks, so the comparisons do not isolate architecture as a single causal factor.

6.2 LARGER BUDGETS DO NOT GUARANTEE BETTER RETAINED KNOWLEDGE

Figure 3 shows that SIFT exceeds SEQUENTIAL at 16 of the 17 measured budgets. Its budget-averaged score, computed as normalized partial AUC over 50k–650k, is 27.36%, compared with 25.56% for SEQUENTIAL, a difference of 1.80 percentage points. At 650k, scores are 29.19% and 26.04%, respectively, against a 24.28% no-learning score.

Budget runs start independently, so higher-budget memory need not include useful knowledge acquired at lower budgets or form a better knowledge combination. Performance improves overall but not monotonically.

6.3 RETURN PER TOKEN SEPARATES EFFECTIVENESS FROM EFFICIENCY

Figure 4 separates absolute performance from average efficiency. At 100k, 300k, and 650k, SIFT scores 25.76%, 27.87%, and 29.19%, while RoTI is 1.52, 1.20, and 0.76 score points per 100k actual learning tokens. Additional expenditure can improve performance even when gains grow more slowly than cost. The broader curve shows a downward tendency at larger budgets, not a strictly monotonic decline. RoTI is an average return relative to no learning, not the marginal benefit of the next token, and its decline alone does not diagnose memory debt. The practical question is whether the next

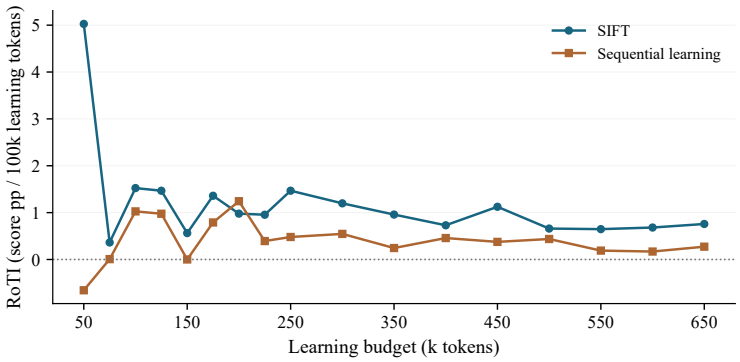


Figure 4: Return on learning-token investment using the same original runs and the matched no-learning baseline. The numerator is domain-macro score gain; the denominator is actual learning expenditure, not the nominal budget. The horizontal zero line separates positive from negative return. This is average, not marginal, return.

investment is worthwhile, or whether the agent should stop or reallocate its budget; answering it requires prospective estimates beyond retrospective average RoTI.

6.4 Trace-level evidence: acquisition is only one link in transfer

Resource access, retrieval, and correct use are distinct. At 650k, SIFT studies 14 theorem opportunities covering all 14 theorem knowledge units, whereas SEQUENTIAL studies 28 opportunities covering only two. The number of theorem tasks with retrieved memory rises from 12/60 to 60/60, but the number solved rises only from 9/60 to 10/60. Selection therefore improves coverage efficiency substantially without a proportional improvement in effective transfer. A `studied` flag records resource access, not demonstrated skill mastery.

Retention can corrupt correct learning material. A teaching resource correctly states $29{,}791 = 31^3$ and $\varphi(29{,}791) = 28{,}830$. The 100k SIFT run compresses this into an erroneous memory treating 29,791 as 7^5 and records 27,390. That item is retrieved on a failed theorem task. Other cases, including the Pick's theorem example, retrieve a correct formula but fail during application. These are different failure locations: erroneous consolidation and incorrect execution of retained knowledge. Neither should be obscured by counting memory entries or retrieval hits as acquired capabilities.

Together, the traces motivate tracking the full path from investment to outcome:

$$\text{Resource coverage} \rightarrow \text{correct retention} \rightarrow \text{relevant retrieval}$$
$$\rightarrow \text{correct application} \rightarrow \text{valid submission} \rightarrow \text{verifier success.}$$

Logging these stages separates knowledge acquisition from memory fidelity, retrieval, reasoning, and framework execution. Controlled removal or replacement of memory would further test whether a particular item causes an observed regression.

7 Conclusion

LIFT-Bench evaluates how agents allocate finite learning budgets before future tasks are revealed. Its results show that transfer depends on resource selection, faithful retention, and successful application, with gains varying across models and frameworks. No-learning baselines expose regressions after learning, while broader coverage need not yield correct execution. Larger budgets can improve overall performance despite local reversals and lower average returns. These findings motivate agents that judge what to learn, when to stop or reallocate expenditure, and whether acquired knowledge produces valid, verified outcomes.

Reproducibility Statement

The accompanying manuscript package contains LaTeX sources, figure PDFs, and the reported framework scores in CSV form. Appendices specify the controller, budget accounting, retrieval permissions, resource ontology, verifier semantics, and framework configurations. The data snapshot is identified by SHA-256.

Ethics Statement

This study evaluates agents in restricted environments rather than deploying them on real users or services. API-dialogue credentials are benchmark examples, not author credentials; actual provider credentials and private connection details are excluded from the artifacts. Source attribution and transformation provenance are retained. Upstream redistribution conditions apply to the source-derived resources.

AI Use Statement

Generative AI tools assisted with implementing and debugging benchmark code. They also supported the preparation and revision of instructional learning materials, as described in the dataset construction section. During manuscript preparation, AI tools assisted with literature search and organization, reference checking, English drafting and revision, and restructuring the presentation of experimental results. Human authors led the research design, determined the revision priorities, and made the final decisions about the protocol and interpretation of results. These research-assistance uses are separate from the language models evaluated as experimental agents.

References

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7889–7901, 2023. URL https://aclanthology.org/2023.emnlp-main.489/.

CrewAI contributors. Crewai, 2026. URL https://github.com/crewAIInc/crewAI. Software; evaluated version 1.15.22.

DeepSeek. Introducing DeepSeek-V4.1-Flash: Smarter, faster, more efficient, 2026. URL https://www.deepseek.com/en/news/deepseek-v4-1-flash/.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.16313.

Hugging Face contributors. smolagents, 2026. URL https://github.com/huggingface/smolagents. Software; evaluated version 1.26.0.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2407.01502.

LangChain contributors. Deep agents, 2026a. URL https://github.com/langchain-ai/deepagents. Software; evaluated version 0.7.18.

LangChain contributors. Langgraph, 2026b. URL https://github.com/langchain-ai/langgraph. Software; evaluated version 1.2.12.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, 2023. URL https://aclanthology.org/2023.emnlp-main.187/.

Guanming Liu, Yuqi Ren, Hansu Gu, Peng Zhang, Weihang Wang, Jiahao Liu, Ning Gu, and Tun Lu. StreamMemBench: Streaming evaluation of agent memory for future-oriented assistance. *arXiv preprint arXiv:2606.14571*, 2026. URL https://arxiv.org/abs/2606.14571. Accepted to Findings of EMNLP 2026.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Microsoft contributors. Autogen, 2026. URL https://github.com/microsoft/autogen. Software; evaluated version 0.7.5.

OpenClaw contributors. Openclaw, 2026. URL https://github.com/openclaw/openclaw. Software; evaluated version 2026.9.5.

OpenHands contributors. Openhands software agent sdk, 2026. URL https://github.com/OpenHands/software-agent-sdk. Software; evaluated version 1.49.5.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Qwen Team. Qwen3.8-27B model card, 2026. URL https://huggingface.co/Qwen/Qwen3.8-27B.

Siddharth Sharma, Nilesh Prasad Pandey, Onat Gungor, and Tajana Rosing. LIMBO: Lifelong inference-time memory and budget optimization for LLM agents. *arXiv preprint arXiv:2609.14138*, 2026. URL https://arxiv.org/abs/2609.14138. Accepted to ICTAI 2026.

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, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2505.24760*, 2025. URL https://arxiv.org/abs/2505.24760.

SWE-agent contributors. mini-swe-agent, 2026. URL https://github.com/SWE-agent/mini-swe-agent. Software; evaluated version 2.4.6.

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. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/pdf/625b7da181479e7642abce270739da66290f0fa3.pdf.

Jiarui Yuan, Tailin Jin, Weize Chen, and Zeyuan Liu. Se-bench: Benchmarking self-evolution with knowledge internalization. *arXiv preprint arXiv:2602.04811*, 2026. URL https://arxiv.org/abs/2602.04811.

Z.ai. GLM-5.3-Flash model card, 2026. URL https://huggingface.co/zai-org/GLM-5.3-Flash.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, Qingnan Ren, Shun Zou, Wenxuan Huang, Lin Chen, Zehui Chen, and Feng Zhao. SkillFlow: Benchmarking lifelong skill discovery and evolution for autonomous agents. *arXiv preprint arXiv:2604.17308*, 2026. URL https://arxiv.org/abs/2604.17308.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025a. URL https://arxiv.org/abs/2505.11942.

Tianshi Zheng, Kelvin Kiu-Wai Tam, Newt Hue-Nam K. Nguyen, Baixuan Xu, Zhaowei Wang, Jiayang Cheng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, Yangqiu Song, Ginny Y. Wong, and Simon See. Newtonbench: Benchmarking generalizable scientific law discovery in llm agents. *arXiv preprint arXiv:2510.07172*, 2025b. URL https://arxiv.org/abs/2510.07172.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.15877.

A Resource Representation, Provenance, and Evaluation Semantics

Snapshot. All results in this draft use the 1,019-record snapshot whose SHA-256 is

`9a486bc2d220f78380e6c62775fc5a9139a41e2f05bc48b1a91d64b40d3690dd`.

There are 82 study-only resources, 30 structured-call records, 172 code records, 83 native Reasoning Gym records, 246 source-answer records, and 406 numeric records across the learning pool and test set.

Public card and protected record. The following is a schema illustration, not a new measured example:

```
{
  "opportunity_id": "<opaque resource identifier>",
  "public_card": {
    "interface_hint": "<interaction type>",
    "family_hint": "<activity family>",
    "skill_tags": ["<public capability tag>"],
    "brief": "<natural-language description of the resource>"
  },
  "attempt_context": {"instruction": "...", "response_format": {}},
  "learning_material": {"method": "...", "worked_steps": []},
  "private_metadata": {
    "domain": "...", "family_id": "...", "skill_unit": "...",
    "split": "learning | test", "source": {},
    "task_type": "...", "answer": "<protected>"
  }
}
```

The schema illustrates the paper-facing terminology. The public pool excludes protected metadata, while selected task context and learning material are revealed through the relevant actions. Answer keys and verifier internals remain host-side. Public-field enforcement, content review, and memory retrieval permissions are distinct checks; the last is specified in Appendix C. The records have passed independent human review.

Support relations. Every evaluation family has at least one corresponding learning resource.

Table 3: Source transformations and score semantics. The source benchmarks are cited in Section 4; these adaptations do not reproduce their original leaderboard protocols.

Source	Conversion	Evaluated contract
BigCodeBench	Library-rich code and interface primers; fixed retained tasks	Unit-test fraction in a restricted process
SE-Bench	Renamed-API documentation paired with coding tasks	Functional test fraction; API use separately logged
BFCL	Function documents separated from call tasks	Typed stateless call matching
API-Bank	Dialogue prefix, discovered schemas, next-call target	One typed next call; no live final-state evaluation
Reasoning Gym	Seeded retained instances and teaching resources	Native reward in an isolated worker
TheoremQA	Typed-answer selection, source-based grouping, teaching	Declared answer contract, not official leaderboard score
Authored theorems	Exact seeded generators and instructional examples	Typed numeric/rational correctness
NewtonBench	Local-law manuals, observations, new numerical settings	Scalar replay of documented laws

Theorem units. The 14 units are absorbing random walks, ballot/reflection counting, rotational orbits via Burnside's lemma, Euler totients, finite-population probability, the matrix determinant lemma, modular-constraint composition, spanning structures in networks, partial derangements, lattice geometry via Pick's theorem, polynomial transfer, recurrence state transitions, perfect assignments via Ryser's formula, and surjective assignment counting. Unit names identify intended reusable dependencies; their presence is not evidence that a model mastered them.

Illustrative learning–evaluation relation. For the local-world law $F = C(m_1 m_2/r)^2$, a learning resource exposes the formula and its nonpositive-input guard, then works through a ratio example. The evaluation context identifies the same local world and gives new numerical settings without restating its manual. For positive settings (a, b, r) and (a', b', r'), the required ratio is $((a'b'r)/(abr'))^2$. A textbook inverse-square law with unsquared masses produces a different answer. The distinction is a coherent change of rule within a declared world, not a hidden change to a familiar physical constant. In contrast, theorem resources teach standard mathematical results, whose prior familiarity is measured rather than assumed absent.

B Quality Checks and Documented Corrections

Evidence and scope. Quality control distinguishes record structure, learning support, reference correctness, response-format acceptance, and content leakage. Human author input sets construction requirements and revision priorities; scripts enforce contracts, and model-assisted review proposes case-level corrections.

Table 4: Record-backed quality checks on the evaluated snapshot.

Check	Count	Share	Interpretation
Structural contract pass	1,019/1,019	100%	Required structure accepted
Duplicate opportunity IDs	0/1,019	0%	Exact identifier uniqueness
Test-family learning support	65/65	100%	Family-level resource coverage
Test-unit learning support	245/245	100%	Direct same-unit resources
Documented reference corrections	2/1,019	0.20%	Verified changes, not an estimated error rate
Documented precision normalization	1/1,019	0.10%	Rounding compatibility, not a wrong-answer count

Reference arithmetic: a perpetuity. Resource `op_577b0c87faec4acb5bde` asks for the intrinsic value of a preferred share paying \$1.90 annually at a 9% required return. The recorded answer 22.11 was corrected to $1.90/0.09 = 21.111\ldots$. Direct recomputation and the repair log agree with the reference in the evaluated snapshot.

Reference expectation: Gaussian cross entropy. For `op_14bf957fa7dfa55aed9d`, samples follow $\mathcal{N}(0, 1)$ but are evaluated under the density of $\mathcal{N}(1, 1)$. Since $\mathbb{E}[(X-1)^2] = 2$, the limiting mean negative log density is $\frac{1}{2}\log(2\pi) + 1 = 1.9189385\ldots$ nats, not the earlier reference 1.4. The revision addresses a missing expectation term rather than an output-format discrepancy.

Precision handling: de Broglie wavelength. For `op_2099079f34fc5474c496`, a 57 g ball traveling at 25 m/s has wavelength $h/(mv) = 4.649873789\ldots \times 10^{-34}$ m. The revised reference stores this value and retains rounding compatibility with 4.7×10^{-34}. This change is classified as precision normalization, not a substantive error in the earlier rounded answer.

Data integrity and independent review. The benchmark has passed scripted validation and independent human review, confirming the absence of semantic data leakage between learning resources and evaluation tasks.

C Controller and Accounting Details

Selection loop. The agent begins with a compact global index containing family counts and representative descriptions. Inspection reveals a bounded page of cards. The agent can update estimates for a small set of families and submit an investment horizon in tokens. The host ranks eligible opportunities using Equation 5 and executes resource episodes while the authorized horizon and global budget permit. It then presents updated expenditure, memory, and coverage for the next decision. Replanning continues until freeze or exhaustion. There is no fixed number of resources that must be studied per round.

```
state = initialize(pool, learning_budget, empty_memory=True)
while not state.frozen and can_fit_next_request(state):
    decision = framework.run(observe(state), tools=learning_tools)
    charge_all_framework_calls(state, decision.usage)
    if decision.action == "freeze":
        freeze(state)
    elif decision.action == "invest":
        ranked = rank_public_candidates(state, decision.estimates)
        execute_within_authorized_horizon(ranked, state, decision)
    else:
        execute_allowed_learning_action(decision, state)
freeze(state)
for task in evaluation_manifest:
    memory = retrieve_read_only(state.memory, task, max_tokens=600)
    result = evaluate_fresh_session(task, memory, per_task_budget)
    log_score_usage_status(result)
assert memory_hash_unchanged(state)
```

This pseudocode describes phase control, not a claim that agent deliberation or tool overhead is cost-free.

Ranking constants. For a family without a submitted estimate, the default value and risk are 0.35 and 0.25. The uncertainty term is $0.12/\sqrt{1+n}$ when estimated and $0.20/\sqrt{1+n}$ otherwise. Cost has a minimum of 200 tokens, initially blending structural and agent estimates equally; where observed costs are available, this estimate is blended equally with the median of recent costs for the same interface. Candidates with explicitly zero value are excluded, and positive-score candidates are ordered deterministically with hashed tie breaking. These constants are fixed within each run; test rewards are not used to update them during learning.

Navigation caps and continuation. Navigation is charged to C_L, not added as a free allowance. Once its cap is reached, the host continues the last approved ranking only if the agent authorized continuation; otherwise it freezes. Table 5 lists the limits for both experimental settings. These settings constrain the deliberation–acquisition trade-off and are held fixed within each policy comparison.

Table 5: Protocol parameters. Navigation tokens count within, not in addition to, the global learning budget. Test allowances are independent of learning expenditure.

Parameter	Budget study	Framework matrix
Learning budgets	50k–650k (17 points)	50k, 150k, 300k
Per-test token cap	5,000	9,000
Maximum completion tokens/call	1,600	1,600
Retrieved memory items/tokens	3 / 600	3 / 600
Ordering seed	0	0
Temperature	0	0

Study and memory. The retriever accesses the frozen memory store and returns at most three items with a total limit of 600 tokens. Both learning policies have identical retrieval permissions.

Failures and recovery. Transport failures are infrastructure events, not wrong answers. Recovery preserves completed evaluations and charged usage; a scored answer is not sampled again to improve its result. A model that exhausts its authorized episode without a valid submission receives the declared task outcome. Source manifests identify adapter versions and recovery provenance. Completion, token limits, prompt contracts, and adapter configuration are checked separately for each comparison.

D Native Framework Configurations

All configurations expose the controlled world actions. Framework code remains responsible for its native reasoning, tool invocation, and orchestration. The host retains model credentials, verifies tool arguments, enforces phase permissions, and records token usage. No framework is granted unrestricted access to source answers or the benchmark filesystem.

Table 6: Evaluated native configurations. Package versions identify the experimental integrations, not necessarily the latest available release.

Framework	Version	Evaluated configuration
OpenClaw	2026.9.5	Native local CLI/tool loop with the world plugin
OpenHands	1.49.5	SDK Agent/Conversation with typed WorldTool actions and observations
Deep Agents	0.7.18	Native harness supplied with the world tool and metered subordinate calls
mini-SWE-agent	2.4.6	DefaultAgent; command actions restricted to world RPC, not an arbitrary shell
smolagents	1.26.0	CodeAgent with the world tool and restricted imports
CrewAI	1.15.22	Hierarchical manager with operator and reviewer; native memory/cache disabled
LangGraph	1.2.12	Explicit model–tool state graph with conditional continuation
AutoGen	0.7.5	MagenticOneGroupChat with operator and reviewer, not the full web/file team

Software sources are OpenClaw (OpenClaw contributors, 2026), OpenHands (OpenHands contributors, 2026), Deep Agents (LangChain contributors, 2026a), mini-SWE-agent (SWE-agent contributors, 2026), smolagents (Hugging Face contributors, 2026), CrewAI (CrewAI contributors, 2026), LangGraph (LangChain contributors, 2026b), and AutoGen (Microsoft contributors, 2026).

The matrix permits up to 30 metered model calls per episode, subject to framework-native limits and the earlier token cap; its maximum completion allowance per request is 1,600 tokens. The native curve uses a 1,600-token completion allowance and a 5k task cap, while the matrix has a 9k task cap.