CONTINUAL LEARNING EXAM

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents are increasingly designed for continual learning beyond parameter updates, acquiring and updating knowledge from sequential experience while retaining useful prior knowledge. Current evaluations offer limited insight into this capability, as they largely assess individual task performance rather than improvement through accumulated experience. We introduce Continual Learning Exam (CLE), a benchmark developed by 50+ domain experts for measuring experience-driven improvement in realistic professional workflows. CLE comprises 120 human-authored tasks, each organized as a five-level ladder, for a total of 600 levels spanning 89 professional roles across 12 occupational domains. Agents must pass each level to unlock the next, with later levels building on experience accumulated during earlier assignments. Each submission is assessed by an evaluation agent against human-authored rubrics. We compare a single submission per level with a setting that also provides diagnostic feedback on unmet requirements after failure, allowing the agent to reflect on the feedback and resubmit once. Across 17 harness–model configurations, this feedback and revision opportunity yields a median completion gain of 5.3 percentage points. Yet even the best-performing system clears only 47.7% of levels and completes all five levels in just 23.3% of tasks, highlighting continual learning as a key challenge for current LLM agents.

1 INTRODUCTION

For large language model (LLM) agents, continual learning concerns acquiring and updating knowledge from sequential experience while retaining useful prior knowledge (Parisi et al., 2019; Shi et al., 2025). Agents can incorporate this experience into model parameters through supervised fine-tuning or reinforcement learning (Xi et al., 2024; 2025), or adapt through their harness while keeping parameters fixed. Through their harness, agents can, for example, retain information in long-term memory (Packer et al., 2023), extract lessons through reflection (Shinn et al., 2023), and develop reusable skills (Wang et al., 2024; 2025). Evaluating continual learning through these mechanisms requires measuring whether accumulated experience improves agents' subsequent performance.

However, most existing agent benchmarks assess individual task performance without directly measuring such improvement. This limitation arises from how they are constructed. General capability benchmarks such as GAIA evaluate whether agents solve individual problems (Mialon et al., 2024). Tasks are presented and scored independently; their scores therefore characterize task-solving performance, rather than how that performance improves through continued learning. Memory benchmarks such as LoCoMo test retrieval and reasoning over interaction histories through question answering (Maharana et al., 2024), without evaluating how retained knowledge improves subsequent task execution. Benchmarks for Claw-based automation assistants, such as WildClawBench, evaluate agents in executable environments with persistent state, long tool-use trajectories, and cross-application workflows (Ding et al., 2026; Meng et al., 2026). Persistent state supports continuity in workflow execution, but does not by itself establish that an agent becomes more capable through experience. Evaluating cumulative improvement requires a sequence of task attempts that supports experience reuse under controlled feedback conditions.

To address this evaluation gap, we introduce the Continual Learning Exam (CLE), a benchmark for evaluating continual learning in realistic human–AI cowork settings. Created by 50+ industry experts, CLE comprises 120 human-authored and human-verified tasks across twelve professional domains, each organized as a five-level ladder, for a total of 600 levels. In the main evaluation set-

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.

ting, agents must pass each level to unlock the next. CLE also supports custom evaluation through blocks, defined as contiguous segments of a task ladder that can begin at any level, with the required support files provided. Later levels introduce new assignments that build on knowledge and experience accumulated through earlier work. We design the task ladders around five forms of knowledge that agents can acquire, retain, and reuse across levels, as illustrated in Figure 1. Tools and Skills capture reusable knowledge of how to use tools and carry out tasks, while Data Dependency connects later assignments to earlier information and artefacts. Strategy guides the choice of effective approaches and the avoidance of recurring pitfalls. User Context and Domain Knowledge help align task execution with users' preferences and professional rules and conventions.

CLE provides a realistic environment for executing and evaluating continual learning assignments. It includes 23,981 real-world input files spanning 77 distinct file extensions. To simulate the software environments used in these assignments, CLE provides 19 stateful mock applications exposing 1,024 APIs. Together, these resources support collaborative workflows such as integrating collaborators' changes in Git repositories, coordinating meeting times around participants' availability and preferences, and synchronizing project updates across applications. The stateful applications preserve the effects of agents' actions across levels within each task. Submissions are assessed against human-authored and human-verified rubrics containing 16,125 criteria across the benchmark. For each criterion, task authors specify a single, unambiguous expected outcome and the evidence required to verify it. A multi-turn evaluation agent acts as a verifier, using tools to extract evidence from submitted artefacts and final environment state and check it against these predefined outcomes.

We evaluate eleven backbone model variants in 17 configurations across four agent harnesses. All systems execute the same fixed sequence of task ladders with model parameters frozen, retaining reusable harness state across tasks. Trial 1 (K=1) permits one submission per level without diagnostic feedback. In Trial 2 (K=2), a failed first submission returns diagnostic feedback based on the rubric, after which the agent can revise and submit once more. Even with diagnostic feedback and an opportunity to revise failed submissions, the best-performing system completes only 47.67% of levels. These results highlight continual learning as a key challenge for current agent harnesses.

Our main contributions are:

- We introduce CLE, a continual learning benchmark comprising 120 five-level task ladders created by 50+ industry experts. The ladders connect distinct assignments through five forms of transferable knowledge, allowing agents to acquire, retain, and reuse experience in realistic human–AI cowork settings.
- We develop an executable environment with real-world input files, stateful applications, and an evaluation agent that grades submissions against 16,125 human-authored rubric criteria.
- We evaluate 17 configurations spanning 4 agent harnesses and 11 backbone model variants. Diagnostic feedback and an additional submission yield a median completion gain of 5.3 percentage points, yet even the best-performing system completes only 47.67% of levels.

2 Related Work

Research on self-evolving agents explores how agents improve through accumulated experience. Parametric approaches incorporate experience into model weights through supervised fine-tuning or reinforcement learning. Non-parametric approaches keep model weights fixed and support adaptation through mechanisms such as memory, reflection, and skill acquisition. Memory mechanisms retain and retrieve information from earlier interactions, making relevant knowledge available beyond the current context, as in MemGPT (Packer et al., 2023). Reflection mechanisms analyze execution outcomes and feedback to derive lessons that guide subsequent attempts, as in Reflexion (Shinn et al., 2023). Skill mechanisms capture procedural knowledge as reusable workflows that agents can invoke for later tasks, as illustrated by Voyager (Wang et al., 2024) and Agent Workflow Memory (Wang et al., 2025). These mechanisms are complementary: reflections and acquired skills can themselves be retained in memory and retrieved to support subsequent task execution.

Agent benchmarks such as GAIA (Mialon et al., 2024), WebArena (Zhou et al., 2024), and SWE-bench (Jimenez et al., 2024) primarily assess task-solving performance. Recent benchmarks explicitly evaluate how experience supports subsequent task execution. MemoryArena (He et al., 2026) couples memory and action across interdependent sessions; LifelongAgentBench (Zheng et al.,

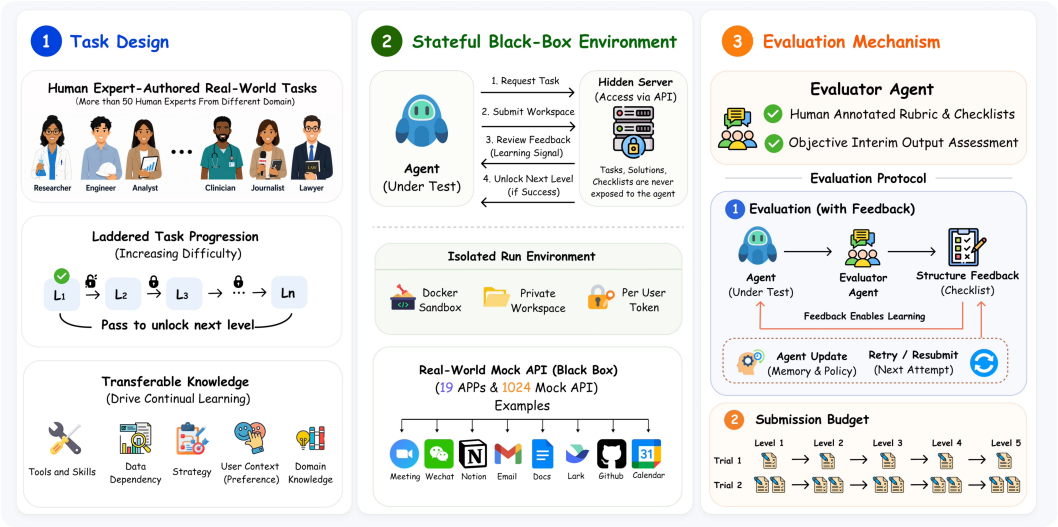

Figure 1: **Overview of CLE.** Each human-authored task comprises five sequential levels linked by transferable knowledge. Agents interact with stateful applications through APIs, and an evaluation agent verifies submissions against human-authored rubrics. A per-level submission budget of $K=1$ allows one submission without diagnostic feedback; $K=2$ provides diagnostic feedback after a failed first submission and permits one resubmission.

2025) evaluates learning across skill-grounded, interdependent tasks; and SkillFlow (Zhang et al., 2026b) studies skill discovery, refinement, and reuse across sequential tasks. CLE complements these efforts with expert-authored professional assignments organized into five-level ladders, combining multiple forms of knowledge reuse with a controlled protocol for diagnostic feedback and revision. In the main evaluation setting, passing each level unlocks the next, while a predefined submission budget controls opportunities for feedback and resubmission. We provide a detailed comparison with other benchmarks in Appendix B.

3 Continual Learning Exam

3.1 Benchmark Design

CLE comprises 120 tasks, each representing a realistic human–AI cowork setting and organized as a five-level ladder, yielding 600 human-authored and human-verified levels in total. Developed by domain professionals, each task is grounded in a specific occupational role, and its five levels represent different work assignments that a practitioner in that role might encounter. Each level has a human-authored rubric specifying the requirements that a submission must satisfy to pass. We describe the benchmark design in detail below.

Coupled ladders with transferable knowledge. We design task ladders by connecting distinct professional assignments through information and artefacts encountered or produced during earlier work. Later levels introduce new objectives and expected deliverables, with requirements that draw on information, constraints, and procedures established in preceding levels. This structure supports evaluating whether agents acquire and retain knowledge from earlier assignments and use it to improve their performance on later ones. Across the benchmark, we organize these connections around five forms of transferable knowledge. **Tools and Skills** require agents to apply or adapt tool-use knowledge and procedures acquired during earlier assignments to later work. **Data Dependency** makes earlier information and artefacts inputs to subsequent deliverables, requiring agents to retrieve and integrate them with current task inputs. **Strategy** connects earlier experience to later decisions through the reuse of effective approaches and lessons for avoiding recurring pitfalls. **User Context** requires preferences learned through earlier interactions to inform subsequent decisions and outputs. **Domain Knowledge** requires professional rules and conventions established or inferred earlier to be

applied under new task requirements. These connections extend beyond adjacent levels, requiring agents to identify and use relevant experience from different points in the ladder. Human-authored rubrics specify the outcomes and constraints that submissions must satisfy, allowing the evaluator to assess the resulting deliverables and environment state.

Diagnostic feedback. Feedback is derived from human-authored rubrics that group detailed checklist items into broader assessment categories. When the submission budget permits, a failed submission receives feedback identifying the relevant rubric category, while individual checklist items and reference answers remain hidden. This feedback provides an explicit learning signal that the actor can use to investigate the problem, revise its approach, and test the revision through another submission. Lessons from this process can be retained in the harness and reused in later levels or tasks to avoid repeating the same mistakes. The grading procedure is described in Section 3.2.

Stateful mock service. Each task supplies real-world input files and specifies the workspace and application state required for its assignments. CLE provides **19** stateful mock applications exposing **1,024** APIs, covering software used by the represented professional roles. These applications support collaborative workflows such as integrating collaborators' changes in shared repositories and coordinating meetings around participants' availability and preferences. Each application maintains its own database, initialized separately for each run. Within a task ladder, files and application state can carry forward the effects of earlier actions, allowing later assignments to build on prior work.

Block-based evaluation. An **evaluation block** is a contiguous segment of a task ladder. For task $j \in \{1, \ldots, 120\}$, a block spanning levels k through m is represented as

$$B = (j, k, m), \qquad 1 \leq k \leq m \leq 5. \tag{1}$$

A single-level block has $k = m$, and a full-ladder block has $(k, m) = (1, 5)$. The main experiments use full-ladder blocks $B_j = (j, 1, 5)$ for all 120 tasks, executed as a single serial stream in a fixed order, with persistent harness state retained across tasks. Levels are completed in order; failure after exhausting the submission budget ends the block, and its unreached levels count as failed.

Blocks can also be selected, repeated, or interleaved across tasks. We manually prepare and load the support files required to start at each selected level, including necessary predecessor artefacts. Each level retains its original assignment and rubric.

3.2 Agentic Evaluation

CLE evaluates each submission against a human-authored and human-verified rubric. For each level, the task file describes the assignment, while a separate evaluation file contains the rubric. The evaluation file and its reference materials remain hidden from the agent under test. For each checklist item, task authors specify a concrete target, an expected outcome, and the evidence required for verification. Each item is designed to admit a single, unambiguous pass/fail judgment for a given submission. A multi-turn evaluation agent uses tools to extract evidence from submitted artefacts and authoritative application state and check it against these predefined targets. The evaluator verifies the submitted work against human-verified reference materials rather than independently solving the task to generate its own reference answer. A submission passes only when every checklist item is satisfied. With a per-level submission budget of $K{=}1$, the actor can submit once and receives no diagnostic feedback. With $K{=}2$, a failed first submission returns diagnostic feedback based on the rubric, after which the actor may revise its work and submit once more.

The evaluation agent runs in an isolated grading environment, separate from the agent under test. A unique API credential identifies each run; a unified interface provides task inputs, file transfer, task services, and grading requests, allowing actor execution and grading on separate machines. The actor cannot access the evaluator's private grading materials or workspace. Each grading request starts a fresh Docker container isolated from the actor and other grading executions. Requests are logged to support reproduction, re-grading, and auditing.

3.3 Problem Formulation

Task execution and submission verification form two linked decision processes in CLE. The actor modifies task files and application state, and the evaluator checks submitted work against human-

authored rubrics. The actor can retain information and update reusable procedures in its harness, using experience from execution and available feedback to guide subsequent actions.

Actor POMDP. We model execution as a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998),

$$\mathcal{M}_A = \langle \mathcal{X}, \mathcal{A}, \mathcal{O}, P, \Omega, R_A \rangle. \tag{2}$$

Here $\mathcal{X}$, $\mathcal{A}$, and $\mathcal{O}$ are the state, action, and observation spaces. The environment state x_t comprises the current task and level, remaining budgets, workspace and service state, server-side logs, and private grading materials. The actor observes task instructions, tool outputs, and permitted feedback through o_t; private rubrics and reference answers remain hidden.

Let s_t denote the state maintained by the harness and used by the actor to select actions. The actor's policy is $\pi_\theta(a \mid s_t)$, where θ denotes the backbone parameters. In the main evaluation setting, task files and service state evolve across levels according to the task manifest and are initialized for each new task. Harness state is retained across tasks.

P specifies how actions change the state, Ω specifies the observations returned to the actor, and R_A scores task execution. An action a_t, such as a tool interaction or submission, produces a new state and observation according to

$$x_{t+1} \sim P(\cdot \mid x_t, a_t), \qquad o_{t+1} \sim \Omega(\cdot \mid x_{t+1}, a_t). \tag{3}$$

Each submission consumes one attempt and pauses execution for verification. R_A is 1 when a submission is judged to satisfy every rubric item and 0 otherwise. A pass advances to the next level or task. After a failure, the actor can revise if attempts remain; otherwise the run proceeds to the next task.

Evaluator MDP. On submission, the evaluator receives a fixed verification instance: submitted artefacts, server-side application state, the private rubric, and reference answers. Its tools verify the submission against the human-defined targets described in Section 3.2. We model verification as a fully observable Markov Decision Process (MDP) and specify its dynamics through the state space, action space, and transition kernel:

$$\mathcal{M}_E = \langle \mathcal{Y}, \mathcal{A}_E, P_E \rangle. \tag{4}$$

The state $y \in \mathcal{Y}$ comprises the fixed verification inputs and current verification progress, including completed checks, their results, tool execution state, and remaining budget. Full observability is assumed at the level of this verification workspace. Actions in $\mathcal{A}_E$ extract content, execute checks, or return a verdict. Given y and the chosen action, P_E updates the verification progress while leaving the inputs unchanged. The evaluator's policy and configuration remain fixed. Verification terminates with pass if every rubric item is judged satisfied, and failure otherwise. For a submission action, this verdict determines the actor's reward and progression through R_A and P; the verdict and permitted diagnostic feedback are returned through Ω.

Learning from experience. Let $e_t = (a_t, o_{t+1})$ denote the new experience from an interaction, including any task outcome or permitted diagnostic feedback returned in o_{t+1}. In our evaluation, θ remains fixed. Expressing the update in Equation (7) (Appendix A) at each interaction step gives

$$(\theta, s_{t+1}) = U_t(\theta, s_t, e_t). \tag{5}$$

The possibly stochastic operator U_t incorporates this experience into the harness state using only information available to the actor. Combining this update with the actor's policy and environment dynamics yields the joint transition kernel

$$\begin{aligned} K_{A,t}(x', s' \mid x, s) = \sum_{\substack{a \in \mathcal{A} \\ o' \in \mathcal{O}}} & \pi_\theta(a \mid s)\, P(x' \mid x, a)\, \Omega(o' \mid x', a) \\ & \times \Pr\big[U_t\big(\theta, s, (a, o')\big) = (\theta, s')\big]\,. \end{aligned} \tag{6}$$

The probability in the last factor is over randomness in U_t. Updates can support both revision after feedback within a level and experience reuse across assignments. Through $\pi_\theta(a \mid s)$, changes to the harness state can influence subsequent actions without changing the backbone parameters.

3.4 Data Construction

The task environments and grading specifications are authored by domain professionals. We recruited **50+** industry experts across a broad range of knowledge-intensive occupations. They include PhD researchers across disciplines, software and hardware engineers, accountants, and financial and operations analysts, among others. Each expert authors benchmark tasks around work from their own role that would benefit from agent assistance and grounds those tasks in authentic input files rather than AI-generated artefacts. The data construction pipeline is summarized in Figure 2.

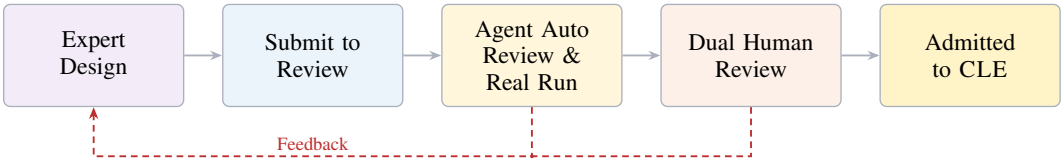

Figure 2: **Data Construction Pipeline.** A domain expert drafts each task from a workflow in their own professional practice. The draft is audited by the review agent and cross-reviewed by two randomly assigned human reviewers, and it enters CLE only after clearing both gates. A rejection at either gate returns the task to its author for revision.

For each level, the task designer defines the occupational role and working context, provides the input artefacts, specifies the expected deliverable, and writes a checklist of criteria that a passing submission must satisfy. A task-review agent then audits the draft using a review skill authored and refined by human experts over multiple iterations. Following the skill's predefined review criteria, the agent checks whether each task meets the benchmark's design requirements. The review includes checks on format compliance, the authenticity of input files, answer leakage, rubric verifiability, and continual-learning design. The review agent also switches to an actor role and runs the task end to end through the same interface available to an evaluated agent. This real execution verifies that the task functions as intended, poses a realistic and non-trivial challenge, and exhibits a clear progression in difficulty across levels. These checks help keep the benchmark challenging for current agents, leaving room to measure gains from continued learning. Each review returns one of five conference-style verdicts, Strong Accept, Accept, Borderline, Major Revision, or Reject, together with itemized findings and concrete revision suggestions. Task designers revise and resubmit their drafts until they meet the acceptance standard. We provide further details in Appendix C.

Tasks receiving Accept or higher proceed to final human review. Each task is assigned two randomly selected reviewers, who independently conduct a full manual review. A task is admitted only if both reviewers accept it; otherwise, it returns to the task designer for revision. The 120 tasks that clear both the automated and human review stages constitute the released benchmark.

3.5 Data Overview

CLE contains **120** tasks spanning occupational roles from twelve domains, with between 7 and 13 tasks in each domain (Figure 3A). Each task is organized as a ladder of **five** gated levels, for **600** levels in total. Tasks are written in the working language of their contributors, with 508 levels in English and 92 in Chinese.

The released benchmark contains 23,981 files, including spreadsheets, JSON records, PDF documents, email messages, images, audio recordings, and source code (Figure 3B). Task inputs consist of real-world files spanning 77 extensions and retain noise such as duplicate records and conflicting information. Submissions are evaluated against **16,125** human-authored checklist items, averaging 26.9 per level and 134.4 per task. The number of checklist items increases across levels (Figure 3C). We present an example of a complete five-level task in Appendix F (Figure 9).

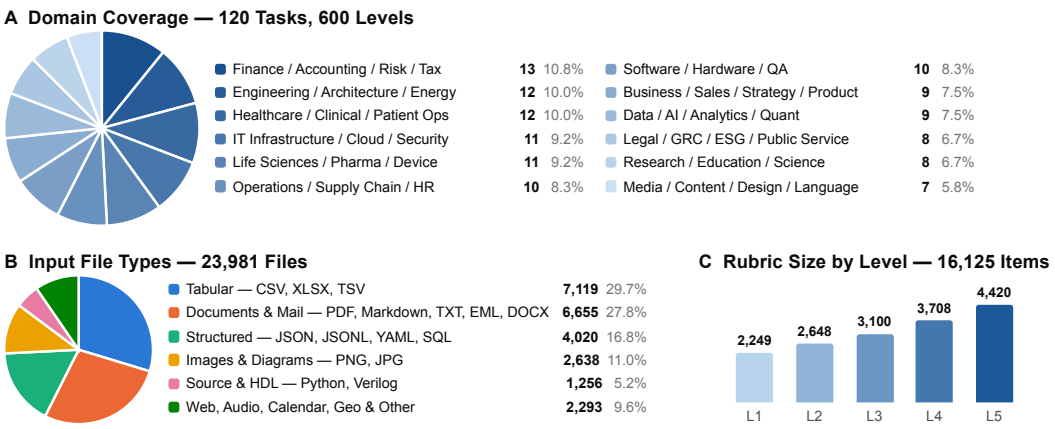

Figure 3: **CLE Statistics. (A)** The 120 tasks (600 levels) grouped into twelve occupational families and counted in tasks. **(B)** The 23,981 input files grouped by format family, counted in files. **(C)** The 16,125 human-authored checklist items by level, counted in items summed over the 120 tasks.

4 Experiment

4.1 Evaluation Setup

Systems under evaluation. We evaluate **17** agent configurations spanning four agent harnesses and eleven backbone models (Table 1). The harnesses are Claude Code (Claude Code Team, 2026), Codex (Codex Team, 2026), Hermes-Agent (Hermes Team, 2026), and OpenClaw (OpenClaw Team, 2026). The backbone models are GPT-5.6-Sol (OpenAI, 2026), GPT-5.6-Terra, GPT-5.6-Luna, Claude Opus 5, Claude Sonnet 5 (Anthropic, 2026a;b), Gemini 3.7 Flash (Google DeepMind, 2026), Doubao-Seed-Evolving (Volcengine, 2026), Qwen3.8-Max (Qwen Team, 2026), Hy3 (Tencent Hunyuan, 2026), DeepSeek V4 Pro (DeepSeek-AI, 2026b), and GLM 5.3 (Zhipu AI, 2026).

Evaluation protocol. Every system is run over the full benchmark as a single serial stream of all 120 ladders in one fixed order. The agent under test uses the benchmark API provided by CLE (Section 3.2) as its sole interface for retrieving tasks and input files, observing the task environment, and submitting deliverables for grading. In Trial 1 ($K{=}1$), each agent can submit once per level without diagnostic feedback. In Trial 2 ($K{=}2$), it can use diagnostic feedback from a failed first submission to revise and resubmit once.

Claude Code and Codex use official Anthropic and OpenAI subscriptions, respectively, while the remaining agents access their models through a unified OpenRouter endpoint. For reasoning effort and thinking mode, we use the default settings of the respective agents and models. Every verdict is produced by the evaluation agent of Section 3.2, instantiated over DeepSeek V4 Flash (DeepSeek-AI, 2026a). To assess alignment with human grading, we compare the evaluator's judgments with adjudicated human labels on a validation sample. The evaluator achieves 96.86% agreement at the rubric-item level (Appendix D, Table 4). The task design supports this agreement. Each checklist item specifies an unambiguous expected outcome and the evidence required for verification.

Metrics. We evaluate every system on all 120 tasks, and a level the agent never unlocks is scored as failed. The per-level pass rate L_k is the proportion of tasks whose first k levels are all cleared within the trial's budget, so L_5 is the proportion of tasks completed in full. Overall completion (Compl. %) is the proportion of the 600 levels cleared. In addition to these pass rates, we report the resources each trial consumes along three axes. **Actor Tok.** reports the total number of tokens consumed by the agent under test. **Time** is the cumulative agent execution time, obtained by summing per-task active runtime over the 120 tasks; we sum active runtime rather than the wall-clock span of the run so that queueing delays, provider throttling, and parallelism across machines do not affect the measurement. **Cost** is the corresponding end-to-end expenditure in USD.

4.2 Results and Analysis

Table 1 reports the results for all 17 agent configurations in Trials 1 and 2. Completing all five levels of a task remains challenging for current agents, even with diagnostic feedback and an opportunity to revise their work. We provide a more detailed analysis in Appendix G.

Table 1: **Main results on CLE.** Rows are grouped by agent harness, with one backbone model per row. Trial 1 allows one submission per level. Trial 2 allows a second submission after diagnostic feedback on a failed first submission. Completion and level pass rates are reported as percentages. ↑/↓ indicate whether higher/lower is preferred.

Setting	Trial 1									Trial 2								
	Compl. ↑	Per-Level Pass ↑					Actor Tok. ↓	Time ↓	Cost ↓	Compl. ↑	Per-Level Pass ↑					Actor Tok. ↓	Time ↓	Cost ↓
		L1	L2	L3	L4	L5					L1	L2	L3	L4	L5			
Claude Code																		
Claude Opus 5	42.17%	79.17%	49.17%	37.50%	25.00%	20.00%	380.87M	32.80h	$410.92	47.67%	85.83%	56.67%	39.17%	33.33%	23.33%	520.62M	58.90h	$720.56
Claude Sonnet 5	27.00%	65.00%	35.00%	20.00%	8.33%	6.67%	470.51M	19.25h	$220.95	40.17%	74.17%	48.33%	34.17%	25.00%	19.17%	1008.60M	37.10h	$506.30
Codex																		
GPT-5.6-Sol	27.83%	58.33%	32.50%	23.33%	13.33%	11.67%	131.30M	23.10h	$168.32	33.17%	65.00%	38.33%	26.67%	18.33%	17.50%	152.30M	28.20h	$210.83
GPT-5.6-Terra	27.83%	62.50%	31.67%	20.00%	13.33%	11.67%	130.34M	18.60h	$73.49	33.17%	68.33%	40.83%	25.83%	18.33%	12.50%	135.20M	20.20h	$83.16
GPT-5.6-Luna	20.33%	50.83%	25.83%	13.33%	7.50%	4.17%	130.60M	16.20h	$28.62	25.33%	56.67%	30.83%	19.17%	11.67%	8.33%	133.50M	19.10h	$33.53
Hermes-Agent																		
Gemini 3.7 Flash	25.33%	55.00%	33.33%	19.17%	10.83%	8.33%	486.25M	46.10h	$215.64	33.67%	62.50%	40.83%	30.00%	19.17%	15.83%	901.82M	102.20h	$394.84
Doubao-Seed-Evolving	20.33%	48.33%	27.50%	14.17%	6.67%	5.00%	232.01M	64.40h	$74.05	30.00%	62.50%	38.33%	22.50%	14.17%	12.50%	345.81M	81.20h	$102.61
Qwen3.8-Max	25.67%	70.83%	33.33%	15.83%	5.83%	2.50%	267.82M	65.17h	$118.71	26.67%	74.17%	34.17%	16.67%	5.83%	2.50%	286.43M	70.12h	$134.54
Hy3	20.33%	50.83%	26.67%	13.33%	6.67%	4.17%	193.69M	35.90h	$8.26	27.33%	60.83%	35.00%	19.17%	11.67%	10.00%	390.92M	80.30h	$16.18
DeepSeek V4 Pro	21.83%	56.67%	27.50%	15.00%	5.83%	4.17%	312.56M	25.70h	$44.81	27.33%	55.83%	33.33%	20.00%	14.17%	13.33%	685.93M	85.00h	$96.51
GLM 5.3	22.50%	60.00%	27.50%	13.33%	7.50%	4.17%	290.76M	41.50h	$82.61	24.00%	62.50%	30.00%	15.00%	7.50%	5.00%	320.54M	48.90h	$102.58
OpenClaw																		
Gemini 3.7 Flash	18.67%	48.33%	24.17%	12.50%	5.83%	2.50%	323.78M	16.30h	$182.65	24.00%	59.17%	32.50%	15.83%	6.67%	5.83%	524.45M	23.77h	$212.33
Doubao-Seed-Evolving	14.50%	40.00%	15.00%	8.33%	5.00%	4.17%	150.83M	33.10h	$32.60	21.17%	48.33%	26.67%	15.83%	8.33%	6.67%	172.62M	31.30h	$40.50
Qwen3.8-Max	18.17%	60.00%	18.33%	9.17%	1.67%	1.67%	227.52M	52.42h	$92.51	19.83%	65.00%	20.00%	10.83%	1.67%	1.67%	245.38M	58.42h	$101.54
Hy3	9.83%	31.67%	7.50%	4.17%	3.33%	2.50%	190.89M	20.40h	$10.27	14.00%	37.50%	12.50%	7.50%	6.67%	5.83%	198.80M	26.10h	$11.95
DeepSeek V4 Pro	11.33%	30.83%	10.83%	6.67%	5.00%	3.33%	159.21M	24.70h	$32.14	10.00%	30.00%	6.67%	5.83%	4.17%	3.33%	208.87M	32.70h	$48.10
GLM 5.3	6.17%	26.67%	4.17%	0.00%	0.00%	0.00%	184.75M	14.60h	$73.46	11.50%	27.50%	14.17%	9.17%	4.17%	2.50%	190.64M	25.60h	$80.50

Overall performance. Even the best-performing configuration, Claude Code with Opus 5, completes only 42.17% of levels in Trial 1 and 47.67% in Trial 2 (Figure 4A). In Trial 2, it completes all five levels in 28 of the 120 tasks. Across the 17 configurations, the median cumulative pass rate in Trial 2 is 62.50% at L1 but only 8.33% at L5 (Table 1). These results indicate that CLE remains far from saturated for the evaluated models and agent harnesses.

Feedback and revision. Higher completion in Trial 1 does not necessarily imply larger absolute gains in Trial 2. Of the 17 configurations, 16 achieve higher overall completion in Trial 2, with a median increase of 5.33 percentage points (pp). Although Claude Code with Opus 5 leads in Trial 1 completion, Sonnet 5 shows a larger increase, at 13.17 pp compared with 5.50 pp (Figure 4A–B). These results motivate evaluating gains from the additional feedback and revision opportunity alongside performance under a single submission.

Improvements in overall completion do not always extend to later levels of the task ladder. With Sonnet 5, the number of fully completed tasks increases from 8 in Trial 1 to 23 in Trial 2. By contrast, both Qwen3.8-Max configurations show higher cumulative pass rates at L1–L3 but unchanged rates at L4–L5 (Table 1). These results motivate examining both how agents improve with feedback and revision and whether those gains extend to later assignments that build on earlier experience.

Harness comparison. CLE evaluates the continual learning capability of the complete agent, including how its harness retains and reuses experience and how its model reflects on feedback. To examine differences across harnesses, we compare Hermes-Agent and OpenClaw on their six shared backbones. Hermes-Agent achieves higher completion for every backbone under both submission budgets (Figure 4A). Its mean advantage increases from 9.56 pp in Trial 1 to 11.42 pp in Trial 2, where their mean completion rates are 28.17% and 16.75%, respectively. The differences extend to gains from feedback and revision. With DeepSeek V4 Pro, Hermes-Agent improves by 5.50 pp, whereas OpenClaw declines by 1.33 pp (Figure 4B). Appendix G provides the detailed comparisons.

Resource costs. Resource comparisons depend on both the completion metric and the resource considered. In Trial 2, Codex with Terra and Sol both achieve 33.17% overall completion, but Terra

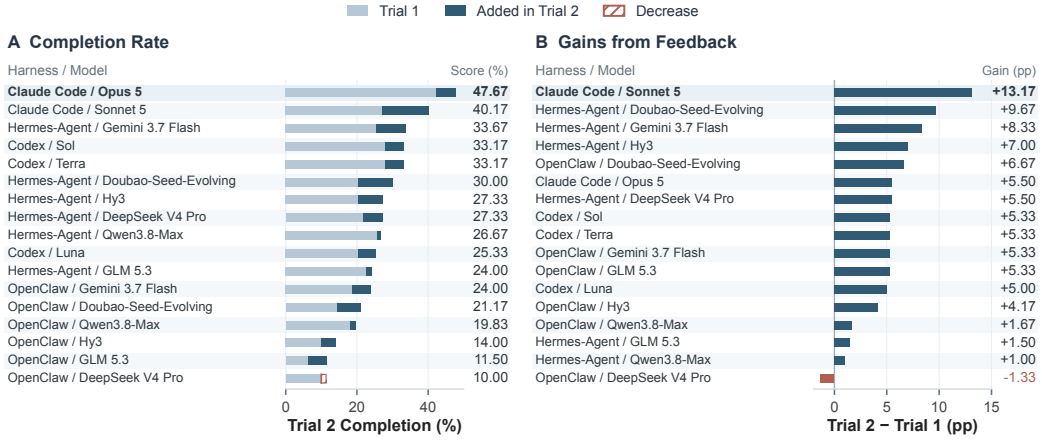

Figure 4: **Completion Rates and Feedback Gains.**

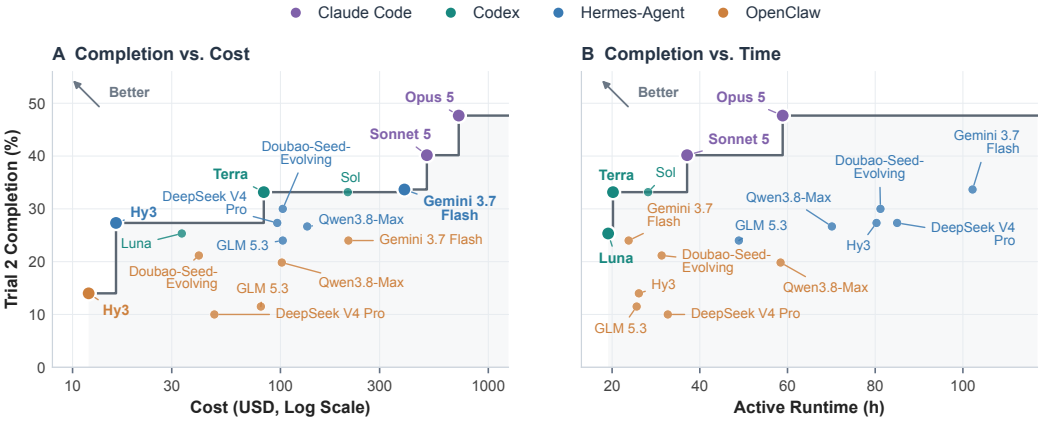

Figure 5: **Completion versus Cost and Runtime.** A–B show all 17 Trial 2 configurations; colors identify harnesses. Upper left is better. Bold labels and larger markers identify the observed Pareto frontier, traced by the step. Resource totals cover full evaluation runs.

costs $83.16 and takes 20.20 hours, compared with $210.83 and 28.20 hours for Sol. However, Sol completes 21 full tasks compared with Terra's 15 (Table 1). Equal overall completion thus masks differences in full-task completion, which matter when interpreting total resource use.

Monetary cost and runtime also yield different trade-offs. Hermes-Agent with Hy3 lies on the observed cost–completion frontier but not the runtime–completion frontier, achieving 27.33% completion for $16.18 while requiring 80.30 hours (Figure 5A–B). These totals cover complete evaluation runs, whose workloads vary with the levels reached and revisions performed. Together, these results motivate assessing resource use alongside both overall and full-task completion, while treating monetary cost and runtime as separate constraints.

5 Conclusion

We introduced the Continual Learning Exam (CLE), a benchmark for evaluating continual learning in realistic human–AI cowork settings. CLE comprises 120 human-authored and human-verified tasks, each organized as a five-level ladder that connects successive assignments through reusable information and artefacts. Stateful environments and human-defined rubrics support task execution and verification throughout each ladder. Across 17 agent configurations with fixed model parameters, the best-performing system completes only 47.67% of levels, even with diagnostic feedback and an opportunity for revision. Feedback and revision improve most configurations, but these gains do not consistently extend to completing entire task ladders. CLE remains challenging for current models and agent harnesses, providing a setting for developing and evaluating methods that acquire, retain, and reuse experience to improve performance across successive assignments.

REFERENCES

Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994. doi: 10.3233/AIC-1994-7104.

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, volume 29, 2016. arXiv:1606.04474.

Anthropic. Introducing Claude Opus 5. `https://www.anthropic.com/news/claude-opus-5`, 2026a.

Anthropic. Introducing Claude Sonnet 5. `https://www.anthropic.com/news/claude-sonnet-5`, 2026b.

ARC Prize Foundation. ARC-AGI-3: A new challenge for frontier agentic intelligence. *arXiv preprint arXiv:2603.24621*, 2026.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020. arXiv:2005.14165.

Yuxuan Cai, Yipeng Hao, Jie Zhou, Hang Yan, Zhikai Lei, Rui Zheng, Zhenhua Han, Yutao Yang, Junsong Li, Qianjun Pan, et al. Building self-evolving agents via experience-driven lifelong learning: A framework and benchmark. *arXiv preprint arXiv:2508.19005*, 2025.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

Claude Code Team. Claude Code. `https://github.com/anthropics/claude-code`, 2026.

Codex Team. Codex. `https://github.com/openai/codex`, 2026.

DeepSeek-AI. DeepSeek V4 Preview Release. `https://api-docs.deepseek.com/news/news260424/`, 2026a. Introduces DeepSeek-V4-Pro and DeepSeek-V4-Flash.

DeepSeek-AI. DeepSeek-V4-Pro general availability release. `https://api-docs.deepseek.com/news/news260813/`, 2026b.

Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, et al. WildClawBench: A benchmark for real-world, long-horizon agent evaluation. *arXiv preprint arXiv:2605.10912*, 2026.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, et al. CL-bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*, 2026.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. A comprehensive survey of self-evolving AI agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017. URL `https://proceedings.mlr.press/v70/finn17a.html`.

Romain Froger, Pierre Andrews, Matteo Bettini, et al. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments. *arXiv preprint arXiv:2602.11964*, 2026.

Irving John Good. Speculations concerning the first ultraintelligent machine. In *Advances in Computers*, volume 6, pp. 31–88. Academic Press, 1965. doi: 10.1016/S0065-2458(08)60418-0.

Google DeepMind. Introducing Gemini 3.7 Flash. `https://blog.google/innovation-and-ai/models-and-research/gemini-models/introducing-gemini-3-7-flash/`, 2026.

Zexue He et al. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026.

Hermes Team. Hermes. https://github.com/nousresearch/hermes-agent, 2026.

Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f4dd765c12f2ef67f98f3558c282a9cd-Abstract.html.

Haonian Ji, Kaiwen Xiong, Siwei Han, Peng Xia, Shi Qiu, Yiyang Zhou, Jiaqi Liu, Jinlong Li, Bingzhou Li, Zeyu Zheng, et al. ClawArena: Benchmarking AI agents in evolving information environments. *arXiv preprint arXiv:2604.04202*, 2026.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66. arXiv:2310.06770.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. arXiv:1909.08383.

Bing Liu, Sahisnu Mazumder, Eric Robertson, and Scott Grigsby. AI autonomy: Self-initiated open-world continual learning and adaptation. *AI Magazine*, 44(2):185–199, 2023. doi: 10.1002/aaai.12087.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ. arXiv:2308.03688.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of LLM agent memory mechanisms. *arXiv preprint arXiv:2605.06716*, 2026. URL https://arxiv.org/abs/2605.06716.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. arXiv:2402.17753.

Fanqing Meng, Lingxiao Du, Zijian Wu, et al. ClawMark: A living-world benchmark for multi-turn, multi-day, multimodal coworker agents. *arXiv preprint arXiv:2604.23781*, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. https://openai.com/index/gpt-5-6/, 2026.

OpenClaw Team. OpenClaw. https://github.com/openclaw/openclaw, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's Last Exam. *arXiv preprint arXiv:2501.14249*, 2025.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, 2026.

Mark Bishop Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, The University of Texas at Austin, August 1994. URL https://www.cs.utexas.edu/~ring/Diss/index.html.

Jürgen Schmidhuber. Gödel machines: Self-referential universal problem solvers making provably optimal self-improvements. *arXiv preprint cs/0309048*, 2003.

Jürgen Schmidhuber, Jieyu Zhao, and Nicol N. Schraudolph. Reinforcement learning with self-modifying policies. In *Learning to Learn*, pp. 293–309. Kluwer Academic Publishers, 1997. URL https://people.idsia.ch/~juergen/ssabook/ssabook.html.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, et al. Agents' Last Exam. *arXiv preprint arXiv:2606.05405*, 2026.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, pp. 9229–9248, 2020. arXiv:1909.13231.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025. URL https://arxiv.org/abs/2512.23675.

Tencent Hunyuan. Tencent Hunyuan officially releases Hy3. https://www.tencent.com/en-us/articles/2202386.html, 2026.

Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2): 25–46, 1995. doi: 10.1016/0921-8890(95)00004-Y.

Volcengine. Doubao-Seed-Evolving model release announcements. https://docs.volcengine.com/docs/ark/model-release-announcement?lang=zh, 2026.

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. *Trans. Mach. Learn. Res.*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a. arXiv:2305.16291.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html. arXiv:2409.07429.

Zijian Wu, Xiangyan Liu, Xinyuan Zhang, et al. MCPMark: A benchmark for stress-testing realistic and comprehensive MCP use. *arXiv preprint arXiv:2509.24002*, 2025.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. AgentGym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html. arXiv:2404.07972.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Bowen Ye, Rang Li, Qibin Yang, et al. Claw-Eval: Toward trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*, 2026.

Eliezer Yudkowsky. Levels of organization in general intelligence. In Ben Goertzel and Cassio Pennachin (eds.), *Artificial General Intelligence*, Cognitive Technologies, pp. 389–501. Springer, 2007. doi: 10.1007/978-3-540-68677-4_12.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Wendong Xu, Yunzhuo Hao, Songcheng Cai, Xiaochen Wang, et al. ClawBench: Can AI agents complete everyday online tasks? *arXiv preprint arXiv:2604.08523*, 2026a.

Ziao Zhang, Kou Shi, Shiting Huang, et al. SkillFlow: Benchmarking lifelong skill discovery and evolution for autonomous agents. *arXiv preprint arXiv:2604.17308*, 2026b.

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*, 2025.

Zhipu AI. GLM-5.3. https://z.ai/blog/glm-5.3, 2026.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*, 2025.

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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx. arXiv:2307.13854.

A Background and Preliminaries

A.1 Continual Learning

Continual learning (CL) concerns acquiring and updating knowledge from sequential experience while retaining useful prior knowledge (Parisi et al., 2019; Lange et al., 2022). At learning stage n, an agent with model parameters θ_n and persistent state s_n uses available experience e_n to update:

$$(\theta_{n+1}, s_{n+1}) = U_n(\theta_n, s_n, e_n). \tag{7}$$

Here s_n denotes the persistent state of the agent's *harness*, the system surrounding the model that orchestrates how it observes, acts, retains experience, and learns. This state is carried into stage n; e_n denotes the data and signals available for learning at that stage, whether obtained through interaction or provided externally. The operator U_n denotes the mechanism by which the learner updates itself from experience. It encompasses parametric learning through changes to model parameters θ_n, non-parametric learning through changes to persistent state s_n, or a combination of both.

A.2 Related Concepts and Learning Mechanisms

Continual learning intersects with several concepts in agent research. Two prominent directions in recent work are learning through memory and skills, and recursive self-improvement (RSI) (Luo et al., 2026; Fang et al., 2025).

Memory and skills. Memory preserves information from earlier experience, while reflection can consolidate that experience into lessons for later use (Packer et al., 2023; Shinn et al., 2023). Skill learning abstracts experience into reusable procedures, which can take the form of executable code or workflows (Wang et al., 2024; 2025). With model parameters fixed, both memories and learned procedures are carried across assignments in the persistent harness state s_n. As new experience arrives, the agent can revise this state and apply the updated knowledge and procedures to subsequent work. These mechanisms support CL when this process improves later performance while retaining useful prior capabilities.

Recursive self-improvement. RSI concerns self-modifications that enhance a system's capacity to produce further improvements (Yudkowsky, 2007), building on Good's idea that intelligent machines could design more capable machines (Good, 1965). For an LLM agent, this can involve revising its harness, using the revised system on subsequent tasks, and applying the resulting experience to further revisions. The recursive element is that these revisions improve how the agent proposes, evaluates, or implements later changes, potentially modifying the learning mechanism U_n itself. For this cycle to accumulate gains from experience, what the agent learns in one round must be retained and used in later rounds. CL is therefore necessary for experience-driven RSI. This connection was explored through self-modifying policies in lifelong reinforcement learning (Schmidhuber et al., 1997). CL can also proceed under a fixed learning mechanism, so it does not require recursive self-modification.

We provide a detailed comparison of related concepts and learning mechanisms in Table 2 to clarify differences in their underlying principles and update targets.

Table 2: **Related concepts and learning mechanisms.** The categories may overlap, and the listed update targets reflect representative implementations.

Concept	Main idea	What is updated
Learning goals		
Continual Learning	Acquiring and updating knowledge from sequential experience while retaining useful prior knowledge (Ring, 1994; Parisi et al., 2019).	Model parameters, persistent memory, or reusable procedures, depending on the method

Continued on next page

Table 2 (continued)

Concept	Main idea	What is updated
Lifelong Learning	Accumulating and reusing knowledge throughout a system's lifetime. The term is often used interchangeably with CL (Parisi et al., 2019; Thrun & Mitchell, 1995).	Model parameters, persistent memory, or reusable procedures, depending on the method
Training and inference approaches		
Multi-Task Learning	Joint training on multiple tasks to improve generalization through shared representations (Caruana, 1997).	Shared model parameters and, where applicable, parameters specific to each task
Meta-Learning	Learning across tasks to improve subsequent learning and adaptation (Finn et al., 2017; Andrychowicz et al., 2016; Javed & White, 2019).	Model parameter initializations, feature representations, or learned update rules
In-Context Learning	Conditioning predictions on examples or prior interactions provided in the current context (Brown et al., 2020).	Input context and transient model activations; model parameters remain fixed
Test-Time Training	Learning from inputs encountered during execution. The adapted state may be reset between sequences or retained across them (Sun et al., 2020; 2025; Tandon et al., 2025).	Model parameters or trainable internal memory; backbone parameters may remain fixed
Test-Time Scaling	Allocating additional inference computation to candidate generation, search, or response verification and revision (Snell et al., 2025).	No update to model parameters or persistent memory is required
Mechanisms for retaining and reusing experience		
Parametric Experience Learning	Supervised or reinforcement learning from interaction trajectories and feedback (Xi et al., 2024; 2025).	Model parameters
Reflection and Memory	Retaining experience and distilling feedback into lessons for subsequent decisions (Shinn et al., 2023; Packer et al., 2023).	Memory entries containing experience records and distilled lessons
Case-Based Reasoning	Retrieving relevant cases, adapting their solutions, evaluating outcomes, and retaining new cases (Aamodt & Plaza, 1994; Zhou et al., 2025).	Stored cases and, in some methods, parameters governing case retrieval
Skill / Workflow Learning	Deriving reusable procedures from experience for subsequent tasks (Wang et al., 2024; 2025).	Skill libraries, scripts, or stored workflows
Forms of self-improvement		
Harness Self-Evolution	Revising the agent harness using feedback from prior tasks to improve execution and experience management (Zhang et al., 2025; Agrawal et al., 2025).	Prompts, tool implementations, rules for memory management, or control code
Recursive Self-Improvement (RSI)	Self-modification that enhances the system's capacity to generate, evaluate, or implement subsequent improvements (Yudkowsky, 2007; Schmidhuber, 2003).	System code or architecture, potentially including the learning mechanism
Gödel Machine	A theoretical system that executes code modifications only after proving an increase in expected utility under its formal assumptions (Schmidhuber, 2003).	Any part of the system's code, including procedures for learning and proof search

A.3 Continual Learning Capability Hierarchy

We organize continual learning capabilities into five cumulative **tiers** (T1–T5), based on learning autonomy and the range of tasks and domains in which agents can learn (Figure 6). Each tier requires the capabilities of all preceding tiers.

At **T1**, predefined rules determine which experience the agent learns from, when and how it updates, and how it retains and reuses knowledge. At **T2**, the agent makes these learning decisions during execution within a given task family, using experience and feedback to check and revise what it has learned. At **T3**, it adapts this learning process to unfamiliar domains without a new human-designed procedure for each domain, while preserving and transferring useful prior knowledge. At **T4**, the agent additionally designs investigations for an externally supplied research problem, without a supplied investigation or validation plan. It formulates testable subquestions, chooses and carries out experiments, analyses, or proofs, and determines how to check the resulting findings. It retains and reuses verified findings in later investigations. At **T5**, the agent also formulates and selects research problems to pursue within a broad objective and constraints, revising its research agenda as verified findings accumulate.

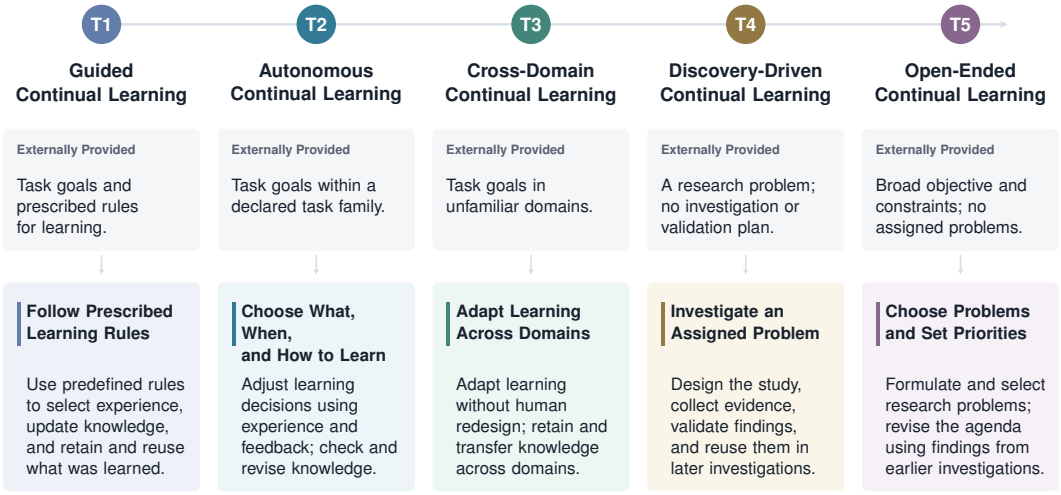

Figure 6: **Continual Learning Capability Tiers.** The tiers are cumulative and distinguish learning autonomy, cross-domain adaptation, research design, and problem selection.

Across all tiers, assessment should establish that agents use experience to improve subsequent work while retaining useful prior knowledge. Agents may use human-designed harnesses and seek external feedback (Liu et al., 2023); autonomy concerns the learning decisions they make during execution. These capability tiers are distinct from the five sequential levels (L_1–L_5) within each CLE task. CLE provides a foundation for studying continual learning through connected assignments that support the accumulation and reuse of experience.

B Benchmark Comparison

We compare CLE with other agent benchmarks across task design, learning protocol, and feedback and grading (Table 3). The criteria for each column are defined below.

Task design. **Human-authored / verified**: every task is human-authored or individually human-verified; ◐ marks mixed provenance such as generated tasks with only sampled human review. **Stateful env.**: the agent interacts with a mutable web, app, API, OS, database, or service environment, rather than only local static files or final-answer strings; ◐ marks environments available for only part of the tasks. **Multi-Modal**: at least one task includes a non-text input the agent must process, such as images, screenshots, audio, video, or visually grounded documents.

Learning protocol. **Tiered tasks**: tasks unlock in an ordered curriculum; ◐ marks a fixed order without gating, or gating present only in a subset of tasks. **Cross-level reuse**: later levels require artefacts, facts, or strategies produced in earlier levels; ◐ marks settings where earlier work only helps, or where the dependence appears only in a subset of tasks. **Config. attempts**: the protocol

grants a configurable per-level submission budget in which a failed attempt returns feedback the agent can act on before resubmitting.

Feedback and grading. **Milestone feedback** reports progress through task stages without disclosing failures on individual evaluation criteria. **Agentic judge** denotes grading by an evaluation agent with tools; ◐ indicates LLM-based grading without tool use. **Audit trail** provides access to judgments on individual criteria, supporting evidence, or evaluator logs; ◐ indicates task-level pass/fail results with execution logs only.

Table 3: **Comparison of agent benchmarks.** ✓ indicates that the benchmark meets the criterion in that column. ◐ indicates that it meets the criterion only partially or for some tasks or settings. ✗ indicates that the feature is absent or is not a focus of the benchmark.

Benchmark	Task Design			Learning Protocol			Feedback and Grading		
	Human Authored / Verified	Stateful Env.	Multi-Modal	Tiered Tasks	Cross-level Reuse	Config. Attempts	Milestone Feedback	Agentic Judge	Audit Trail
WebArena (Zhou et al., 2024)	✓	✓	✗	✗	✗	✗	✗	◐	◐
OSWorld (Xie et al., 2024)	✓	✓	✓	✗	✗	✗	✗	✗	◐
Terminal-Bench (Merrill et al., 2026)	✓	✓	✓	✗	✗	✗	✗	✗	✓
GAIA (Mialon et al., 2024)	✓	✗	✓	✗	✗	✗	✗	✗	◐
CL-bench (Dou et al., 2026)	✓	✗	✗	◐	✓	✗	✗	◐	✓
Agents' Last Exam (Sun et al., 2026)	✓	✓	✓	✗	✗	✗	✗	◐	✓
Humanity's Last Exam (Phan et al., 2025)	✓	✗	✓	✗	✗	✗	✗	◐	✓
AgentBench (Liu et al., 2024)	◐	✓	✗	✗	✗	◐	◐	◐	◐
SWE-bench (Jimenez et al., 2024)	◐	✗	✗	✗	✗	✗	✗	✗	✓
τ-bench (Yao et al., 2024)	✓	✓	✗	✗	✗	✗	✗	✗	✓
MCPMark (Wu et al., 2025)	✓	✓	✗	✗	✗	✗	✗	✗	◐
Gaia2 (Froger et al., 2026)	✓	✓	✗	◐	◐	✗	✗	◐	✓
ARC-AGI-3 (ARC Prize Foundation, 2026)	✓	✓	✓	✓	✓	◐	✓	✗	✓
Claw-Eval (Ye et al., 2026)	✓	✓	✓	✗	✗	✗	✗	◐	✓
ClawBench (Zhang et al., 2026a)	✓	✓	✓	✗	✗	✗	✗	◐	✓
ClawMark (Meng et al., 2026)	✓	✓	✓	◐	✓	✗	✗	✗	✓
WildClawBench (Ding et al., 2026)	✓	✓	✓	✗	✗	✗	✗	◐	✓
ClawArena (Ji et al., 2026)	◐	✓	✗	◐	✓	✗	◐	✗	✓
LifelongAgentBench (Zheng et al., 2025)	◐	✓	✗	◐	◐	✗	✗	✗	◐
SkillFlow (Zhang et al., 2026b)	◐	✓	✓	◐	◐	✗	◐	✗	✓
StuLife (Cai et al., 2025)	◐	✓	✗	◐	✓	✗	✗	✗	◐
CLE (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓

C Agent Auto Review with Feedback

We use a task-review agent to conduct the initial review of each task draft and provide feedback to its author for revision before final review by two human reviewers.

Human expertise as a review skill. The auto-review process follows criteria derived from human expertise in task design. We consolidate review criteria developed and iteratively refined by the benchmark development team into a reusable review skill. The skill specifies what the review agent should inspect, how to apply the review criteria, and how to organize its findings into actionable feedback for the task contributor.

Feedback-driven human–agent iteration. As illustrated in Figures 2 and 7, an author submits a draft and the review agent applies the skill to the full ladder, including its instructions, artefacts, environment state, and evaluation rubric. It returns evidence-grounded findings that identify the affected component, explain why it violates a design criterion, and suggest a concrete direction for

revision. Task authors use the feedback to revise their drafts and submit them for another round of review. This cycle can repeat throughout task construction.

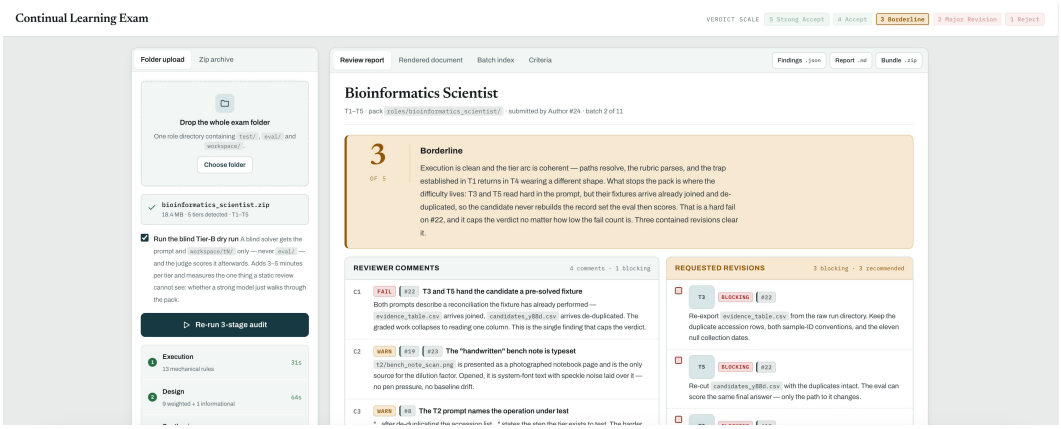

Figure 7: **Review-Agent Interface Used during Task Construction.** A task contributor uploads a complete five-level task together with its supporting files and receives a conference-style verdict, review comments, and revision suggestions after the review agent completes its execution and design checks.

Final human review. Tasks receiving an automated verdict of Accept or Strong Accept proceed to independent review by two human reviewers. A task is admitted to the benchmark only if both reviewers accept it; otherwise, it returns to the author for revision. This separation of roles combines the coverage and consistency of agent-assisted review with the contextual judgment and accountability of human experts. The development team can refine the review skill based on recurring issues identified during task construction, allowing lessons from earlier review rounds to improve feedback for later task authors.

D Human–Agent Grading Agreement

Alignment with human grading. We assess the evaluation agent's agreement with human grading at the rubric-item level. The validation sample covers all five levels of 12 tasks, comprising 60 submissions and 732 rubric items. Human annotators independently label each item and resolve disagreements to establish reference labels. All judges are provided with the same fixed submissions and environment evidence. Our evaluation agent with DeepSeek V4 Flash achieves 96.86% agreement with the human reference labels. Table 4 reports pairwise agreement among the human reference labels and the three agent judges.

Stability across repeated runs. To measure repeatability, we grade a fixed set of submissions covering 556 rubric items three times using the evaluation agent with DeepSeek V4 Flash. Each run starts from a fresh session, with the model, tool access, rubrics, and submitted evidence held fixed. As shown in Table 5, 553 of the 556 items receive the same label in all three runs, yielding 99.46% agreement. These results show high repeatability of item-level grading under fixed evaluation conditions.

E Additional Benchmark Data Analysis

Level-wise statistics. The average checklist size increases from 18.74 items at L_1 to 36.83 at L_5, task-brief length from 246 to 466 words, and the number of rubric sections from 4.10 to 5.74 (Table 6). Increases in checklist size and task-brief length are larger at the upper levels. Since a submission must satisfy every checklist item to pass, later levels require more conditions to be met simultaneously.

Table 4: **Agreement on Individual Rubric Items.** Each cell reports the percentage of rubric items receiving the same label from the corresponding pair of judges. Column numbers correspond to the numbered rows. Human annotators independently assess each rubric item and resolve disagreements to establish the reference labels. Judge 2 is CLE's evaluation agent with DeepSeek V4 Flash. Judges 3 and 4 provide additional comparisons using representative frontier agent–model combinations: Claude Code with Claude Opus 5 and Codex with GPT-5.6-Sol, respectively.

		Judge			
		1	2	3	4
1	Human Annotators	100.00%	96.86%	97.27%	95.49%
2	Ours w/ DeepSeek V4 Flash	96.86%	100.00%	96.58%	92.90%
3	Claude Code w/ Claude Opus 5	97.27%	96.58%	100.00%	95.22%
4	Codex w/ GPT-5.6-Sol	95.49%	92.90%	95.22%	100.00%

Table 5: **Stability of the evaluation agent across repeated runs.** A fixed set of submissions is graded in three fresh sessions. Agreement is the percentage of the 556 rubric items that receive the same label in all three runs.

Judge	Total checklist items	Consistent items	Agreement (%)
Ours w/ DeepSeek V4 Flash	556	553	99.46

Table 6: **Task Statistics by Level.** Each column summarizes checklist size, task-brief length, and the number of rubric sections across the 120 tasks at the corresponding level.

	L_1	L_2	L_3	L_4	L_5
Checklist Items (Mean)	18.74	22.07	25.83	30.90	36.83
Checklist Items (Median)	16.00	19.00	22.00	27.50	34.50
Checklist Items (Max)	75	85	82	87	112
Brief Length (Mean Words)	246	271	300	375	466
Rubric Sections (Mean)	4.10	4.39	4.52	5.21	5.74

Cross-level file references. References to earlier-level files are common across CLE and become more prevalent at later levels. Overall, 88 of the 120 task ladders contain at least one such reference in a task specification or evaluation rubric. The number of ladders with these references increases from 73 at L_2 to 84 at L_5 (Figure 8).

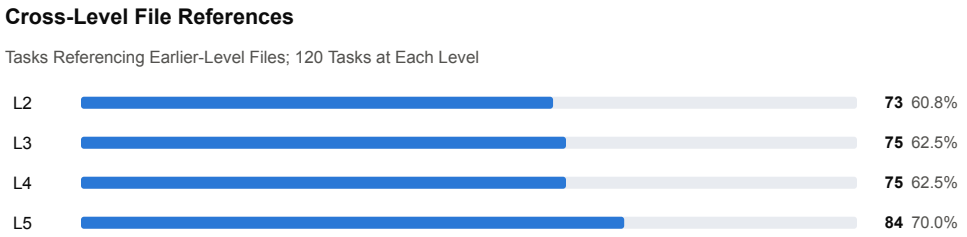

Figure 8: **Cross-Level File References.** Number and percentage of the 120 task ladders whose task specification or evaluation rubric at each level refers to files from an earlier level.

F TASK EXAMPLE

To illustrate CLE more concretely, we present a complete five-level task example in Figure 9. Each row presents one level's assignment, representative inputs and deliverables, the software involved, and the number of human-authored rubric criteria.

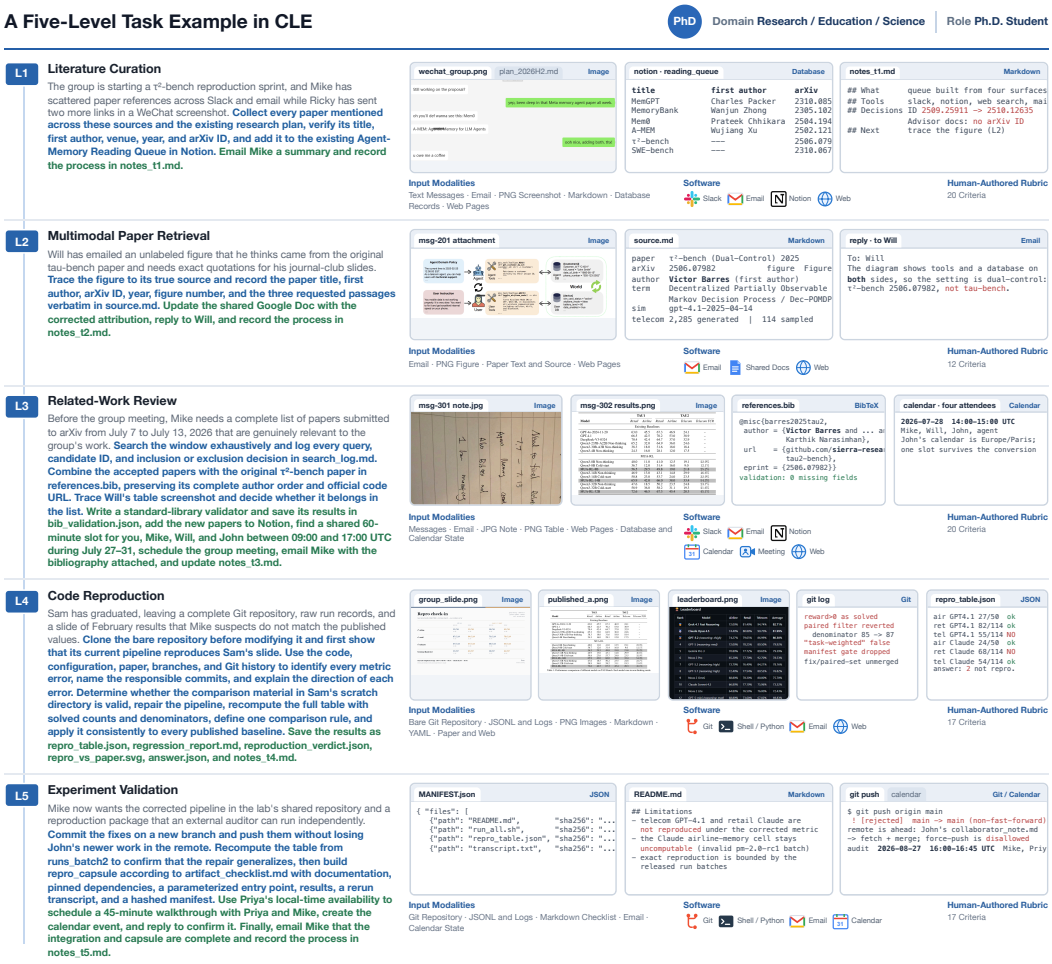

Figure 9: **A Five-Level Task Example from the Continual Learning Exam.**

G ADDITIONAL ANALYSIS

G.1 TASK PROGRESSION

Early termination remains common under both submission budgets (Figure 10A–B). For every configuration in Trial 1, more than half of the 120 tasks end with zero or one completed level, from 61 tasks (50.83%) for Claude Code with Opus 5 to 115 (95.83%) for OpenClaw with GLM 5.3. Across the 17 configurations, the median number of such tasks falls only from 87 (72.50%) in Trial 1 to 80 (66.67%) in Trial 2, and Claude Code with Opus 5 is the only configuration that drops below half in Trial 2, at 52 tasks (43.33%).

Performance at L1 constrains but does not determine how far a configuration gets (Figure 10C). In Trial 2, the Spearman correlation across the 17 configurations between the L1 pass rate and overall completion is $\rho = 0.75$, but between the L1 and L5 pass rates it is only $\rho = 0.42$. The two Qwen3.8-Max configurations are the clearest case. At L1 they pass 74.17% of tasks with Hermes-Agent and

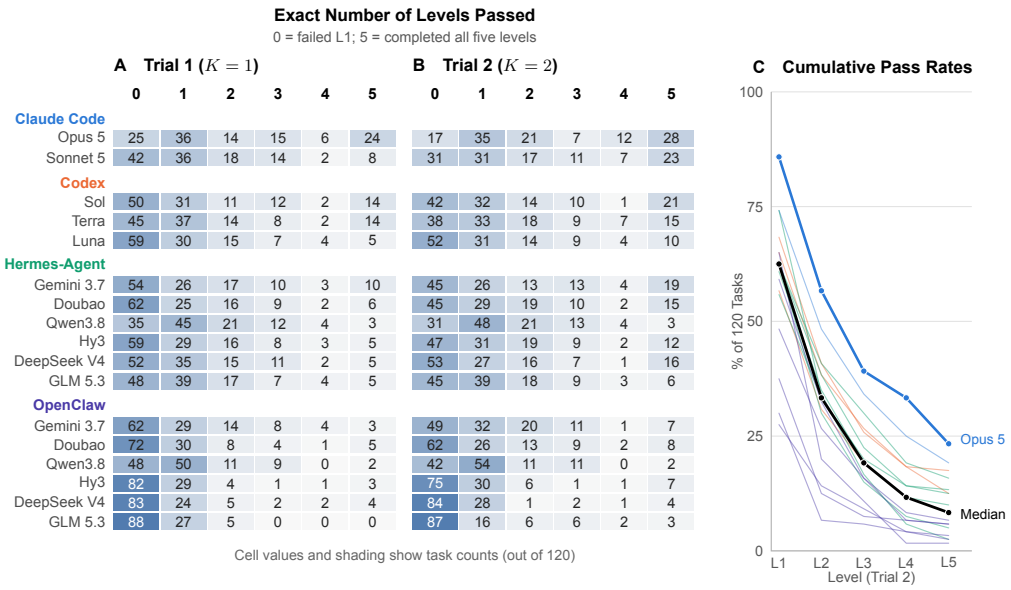

Figure 10: **Task Progression across Five Levels. (A–B)** Distribution of tasks by the number of levels completed (0–5) in Trial 1 ($K = 1$) and Trial 2 ($K = 2$). Each row sums to 120 tasks; darker cells indicate larger counts. **(C)** Cumulative pass rates in Trial 2, measured over all 120 tasks at each level. Colours identify harnesses. The black curve shows the median across the 17 agent configurations. Claude Code with Opus 5, the configuration with the highest overall completion, is highlighted.

65.00% with OpenClaw, both above the median of 62.50%. At L5 they pass 2.50% and 1.67%, both below the median of 8.33%, and the OpenClaw rate is the lowest of the 17 configurations. OpenClaw falls below the median at L2 and Hermes-Agent at L3.

The median conditional pass rate across configurations rises at every transition (Figure 11), from 50.00% (L1→L2) to 75.00% (L4→L5) in Trial 1 and from 55.17% to 80.00% in Trial 2. In Trial 1, the last two medians exclude OpenClaw with GLM 5.3, for which no task passed L3. These increases do not show that later assignments are easier. Each rate is measured only on tasks that cleared every earlier level, so it is computed over a progressively more favourable subset of tasks, and on those tasks the agent also carries forward the experience of the levels it has already cleared. The conditional pass rates reflect both effects.

Despite the rising medians, later bottlenecks remain common. In Trial 2, 8 of the 17 configurations have a later transition with a lower conditional pass rate than their L1→L2 rate. OpenClaw with Qwen3.8-Max, for example, passes L4 on only 2 of the 13 tasks that passed L3 (15.38%). These late rates also rest on few tasks. Across configurations, the median denominator in Trial 2 falls from 75 tasks at L1→L2 to 14 at L4→L5, and 7 of the 17 configurations have fewer than 10 tasks at the final transition. The same OpenClaw configuration's L4→L5 rate is 100.00%, from 2 tasks.

G.2 Feedback and Revision Gains

Small absolute gains at later levels can represent substantial improvements over their low baselines. Across configurations, the median increase from Trial 1 to Trial 2 is 7.50 pp at L2 and 3.33 pp at L5 (Figure 12). Dividing each configuration's increase by its own Trial 1 pass rate before taking the median, the corresponding median relative increases are 22.50% and 75.00%; the L2 median uses all 17 configurations and the L5 median 16, excluding OpenClaw with GLM 5.3, whose Trial 1 L5 pass rate is zero.

Changes in first-level performance alone do not capture the gains in full-task completion. For Claude Code with Sonnet 5, gains reach +16.67 pp at L4 and +12.50 pp at L5, compared with +9.17 pp at

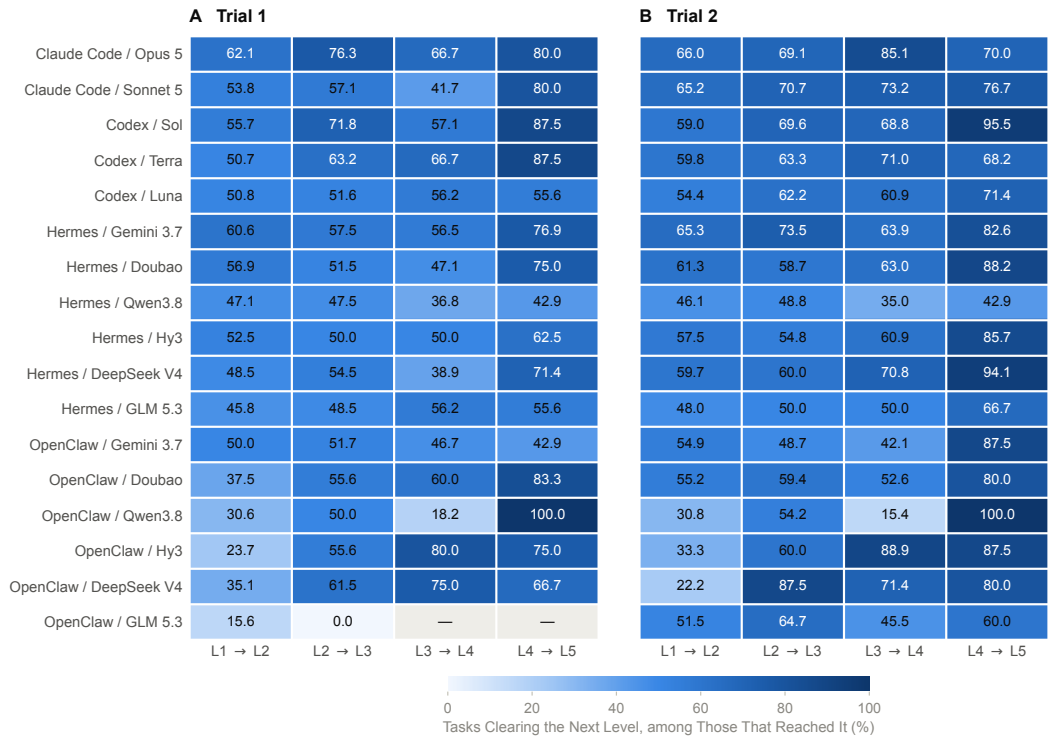

Figure 11: **Conditional Progression between Adjacent Levels.** Each cell shows the percentage of tasks that passed L_{k+1} among those that passed L_k. Values are rounded to one decimal place. A dash indicates that no task passed L_k.

L1. Hermes-Agent with DeepSeek V4 Pro loses 0.83 pp at L1 while gaining 9.17 pp at L5, showing that more tasks can be completed in full even when fewer pass L1. Each difference compares two complete gated runs, so it reflects both access to later levels and success on the levels reached.

G.3 Harness Comparison

Hermes-Agent achieves higher full-task completion than OpenClaw for every shared backbone under both submission budgets (Figure 13). The margin is only one task in four of the twelve backbone–budget comparisons. Its mean advantage in full-task completion across the six backbones grows from 2.36 pp in Trial 1 to 5.56 pp in Trial 2. Its Trial 2 advantage in overall completion ranges from 6.83 pp for Qwen3.8-Max to 17.33 pp for DeepSeek V4 Pro. In Trial 2, its mean actor-token usage and cost are 1.90 and 1.71 times those of OpenClaw, respectively (Table 1). These totals cover runs that reach different levels and perform different amounts of revision, so they do not establish efficiency on matched workloads.

G.4 Effect of Model Generation

To examine whether newer model generations improve performance on CLE, we compare Gemini 3.6, 3.7, and 3.8 Flash with Hermes-Agent under both submission budgets (Table 7). Across generations, we hold the harness configuration, task set and order, tools, and state retention fixed.

Gemini 3.7 Flash achieves the highest overall completion and full-task completion in both trials. In Trial 1, its overall completion reaches 25.33%, compared with 21.00% for Gemini 3.6 and 21.17% for Gemini 3.8. Gemini 3.8 also consumes more actor tokens than Gemini 3.7 (795.78M versus 486.25M), while completing fewer levels and full tasks.

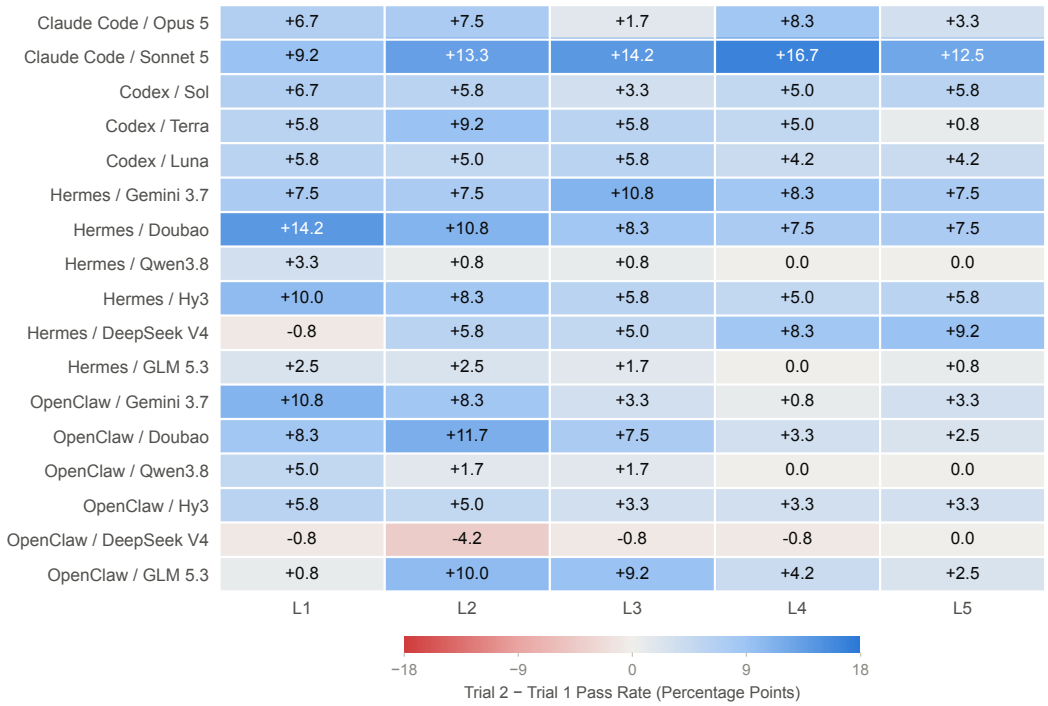

Figure 12: **Feedback and Revision Gains at Each Level.** Cells show Trial 2 minus Trial 1 pass rate in percentage points across all 120 tasks. Blue indicates an increase, red a decrease, and the neutral midpoint no change. Differences use integer pass counts before rounding.



Figure 13: **Harness Comparison on Six Shared Backbones.** Grouped bars compare Hermes-Agent and OpenClaw under the same submission budget. **(A–B)** Overall completion out of 600 levels. **(C–D)** Full-task completion out of 120 tasks.

All three models improve in Trial 2. Gemini 3.8 shows the largest increase in overall completion, gaining 9.50 percentage points, but still trails Gemini 3.7 in both overall completion (30.67% versus 33.67%) and fully completed tasks (14 versus 19). Across these three generations, performance therefore does not improve consistently with model updates, even when diagnostic feedback and revision are available.

Table 7: **Gemini model-generation comparison with Hermes-Agent.** All variants use the full 120-task protocol and the metrics of Table 1. Gemini 3.7 Flash reuses the main-table runs.

Model	Trial 1									Trial 2								
	Compl.	Per-Level Pass					Actor Tok.	Time	Cost	Compl.	Per-Level Pass					Actor Tok.	Time	Cost
		L1	L2	L3	L4	L5					L1	L2	L3	L4	L5			
✦ Gemini 3.6 Flash	21.00%	50.00%	27.50%	13.33%	9.17%	5.00%	389.18M	28.40h	$198.62	23.50%	48.33%	31.67%	19.17%	10.83%	7.50%	602.72M	51.70h	$328.79
✦ Gemini 3.7 Flash	25.33%	55.00%	33.33%	19.17%	10.83%	8.33%	486.25M	46.10h	$215.64	33.67%	62.50%	40.83%	30.00%	19.17%	15.83%	901.82M	102.20h	$394.84
✦ Gemini 3.8 Flash	21.17%	51.67%	28.33%	14.17%	8.33%	3.33%	795.78M	62.90h	$269.20	30.67%	61.67%	39.17%	25.00%	15.83%	11.67%	1741.21M	142.50h	$569.26

H AI USE STATEMENT

We used generative AI tools to assist with language editing and polishing. We also used AI agents for task review and grading, as described in Section 3.2 and Appendix C. Benchmark tasks and rubrics were authored and verified by human contributors. All AI-assisted revisions were reviewed by the authors, who take full responsibility for the final content of this work.