LEARNING FROM DEPLOYMENT FOR CONTINUAL AGENT ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deployment exposes language-model agents to changing tools and workflows, but model providers often receive only interaction records from private client environments. We study harness adaptation from this deployment experience, without replaying the original environment or sampling new actions during training. We instantiate the setting with two offline updates: experience distillation (OPD) and reinforcement learning (RL) over recorded turns. Distillation extracts procedural lessons, reviews the evidence linking them to recorded operations, and uses an experience-conditioned frozen teacher to supervise the corresponding tokens. The RL update assigns rewards to recorded turns and trains on their complete responses. Across two model sizes and two agent harnesses, both updates improve performance in all four settings on ClawGym dev; distillation also improves three settings on PinchBench. Multi-round experiments with a corrective-and-retention variant of distillation show gains in stationary, cross-agent, and changing-workload settings and examine retention on earlier tasks. These results establish deployment experience as a useful resource for work-agent adaptation, with no access to the client environment during training and no experience prompt at inference time.

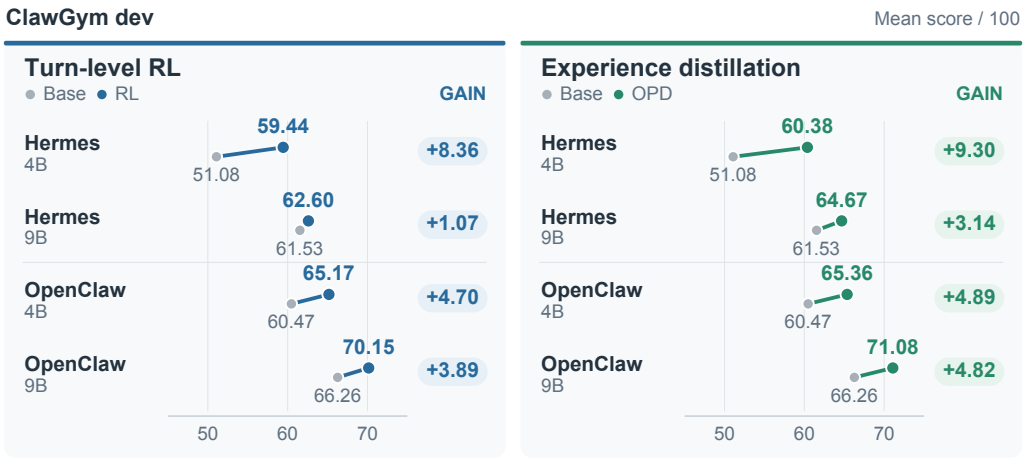


Figure 1: **Both offline updates improve harness adaptation.** Points show three-evaluation mean ClawGym dev scores; gains are score-point changes from Base. Both panels share a 45–75 score axis.

1 INTRODUCTION

Language-model agents carry out multi-step tasks through harnesses that expose tools, format observations, and manage execution (Luo et al., 2025; Xu et al., 2026a). Their behavior depends on both the model and the interface through which it acts. As clients update tools or change workflows, a previously useful procedure can lead to an invalid call, a lost intermediate result, or an incomplete

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.

submission (Xu et al., 2026b; Huang et al., 2026; Vats & Golev, 2026). Adapting deployed agents therefore requires learning how to work within changing interfaces while retaining procedures that remain useful.

Deployment provides a natural source of experience: interaction records show what the agent attempted, how tools responded, and how the agent recovered from mistakes (Ye et al., 2026b; Wu et al., 2026). Yet a model provider may receive these records without access to the client's private files, credentials, or workspace. In this setting, the provider cannot reliably replay a trajectory or test an alternative action in the original environment. The learning problem is to turn the available evidence into model updates without requiring that access. Table 1 places this problem alongside approaches that adapt harnesses, reconstruct executable environments, or update model weights from experience, including Online Experiential Learning (OEL).

Route	No environment access during training	Base-model weights updated	No new response sampling during training
MemoHarness	—	—	—
EvoHarness-RL	—	✓	—
Terminal-Universe	—	✓	—
OEL	✓	✓	—
This setting	✓	✓	✓

Table 1: **Design space for learning from agent experience.** Checkmarks indicate that a route satisfies each heading (Huang et al., 2026; Ning et al., 2026; Wu et al., 2026; Ye et al., 2026b). "No new response sampling" means that policy updates use fixed, logged agent responses rather than newly sampled agent continuations; experience annotation and review may still involve model generation. Environment access includes reconstructed environments.

The central challenge is deciding what to learn from a recorded interaction. A failed task may contain useful actions, while a successful task may include avoidable mistakes. For example, a tool error can identify an invalid argument without providing a reason to change the rest of the response. Whole-response supervision couples these decisions, whereas a generated replacement has no observed outcome in the private environment. The record instead allows us to connect a procedural lesson to the specific operation and feedback that support it.

We study learning from deployment as a setting for work-agent harness adaptation: the provider receives completed interaction records, updates model weights offline, and returns the updated model to the client's native harness. We instantiate this setting with two update rules: experience distillation (OPD) and turn-level reinforcement learning (RL). Both train on recorded responses without executing new actions in the client environment.

For experience distillation, an annotation model extracts procedural lessons, and a separate review links each lesson to supported operations in the original response. A frozen copy of the deployed model scores those operations with the reviewed experience, while the student uses only its original causal context. Following privileged-context distillation (Ye et al., 2026c;b), the probability difference supplies a learning signal on the reviewed token spans. The RL update instead uses a reward assigned to a recorded turn to supervise its complete response. These alternatives let us study how deployment feedback can enter an offline update through either procedural experience or turn rewards. Redeployment requires neither an experience prompt nor a change to the harness.

We evaluate both updates across two model sizes and two harnesses, measuring adaptation on ClawGym dev and transfer to PinchBench. We also compare critical-token and complete-turn supervision for distillation. To study successive updates, we evaluate a corrective-and-retention (CR) variant of distillation that learns from observed mistakes and successful recovery operations. Three continual-learning tracks cover a stationary setting, harness changes, and workload shifts, with evaluations on current and previously encountered tasks.

Both offline updates improve ClawGym dev performance in all four model–harness settings. Experience distillation also improves PinchBench performance in three settings. The continual-learning experiments show gains over the initial model in stationary, cross-agent, and changing-workload

settings, with separate retests measuring retention on earlier tasks. Together, these experiments show how deployment experience can support work-agent adaptation when the original environment is unavailable during training.

2 Learning from Deployment

We study a provider that adapts a work-agent model using interaction records returned by a client, without access to the original private workspace. Figure 2 summarizes this learning setting. We consider experience distillation (OPD) and turn-level reinforcement learning (RL) as two offline update rules. Only deployment through the client creates new environment observations; both updates train on the original recorded responses. We first describe experience distillation, then the RL update and successive rounds of deployment.

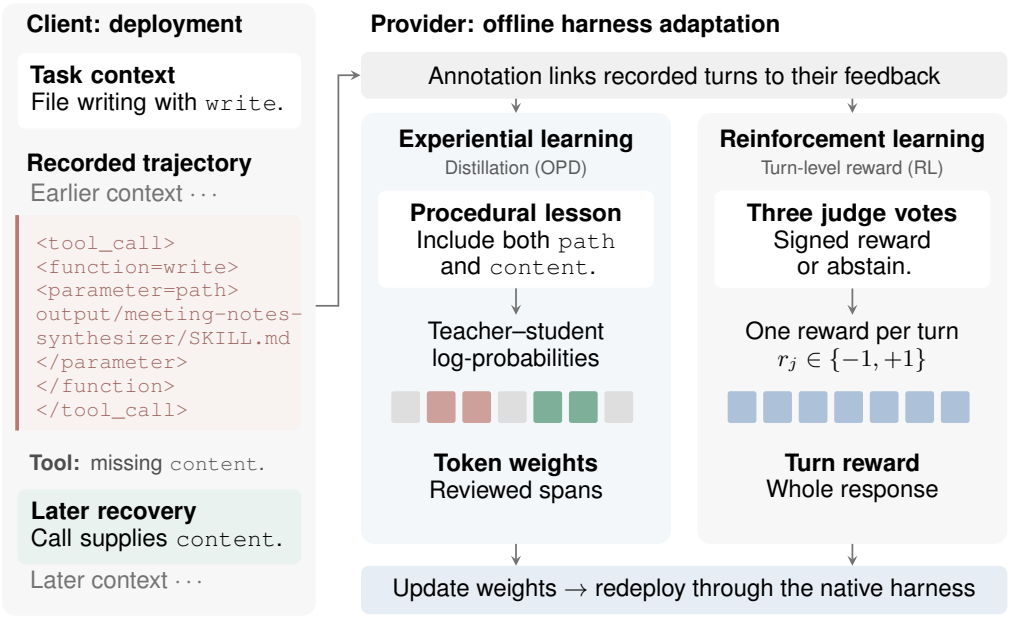


Figure 2: **Two ways to learn from deployment experience.** A recorded write-call error and recovery illustrate both updates. Distillation compares an experience-conditioned teacher with the original-context student on reviewed spans; RL broadcasts one turn reward across the response. Red and green denote correction and retention; token bars are schematic. The call is quoted; task context and recovery are summarized.

2.1 Recover Evidence from Returned Trajectories

A trajectory τ contains model-visible prefixes x_j, original responses y_j, and subsequent tool or environment observations. The provider cannot query hidden state or execute a different action during learning. An annotation model extracts procedural experience e_j from the visible record and cites the events that support it. A separate full-trajectory review checks the extraction. The annotation model is distinct from both the probability teacher and the benchmark evaluator; its implementation is specified in the experimental setup. Benchmark grades and hidden verifier outputs do not enter the training record. The application programming interface (API) logs used here omit sampling token IDs and behavior-policy probabilities, so recorded responses are canonically re-encoded; the objective below does not recover their sampling distribution. The review may use a later tool response to judge an earlier operation, but that later observation never enters the student's earlier causal prefix. This separation lets an experience describe an observed mistake without pretending that the model knew its consequence when it acted.

2.2 Localize Supported Operations

That review distinguishes an avoidable error from an infrastructure failure or an unresolved outcome. For example, a tool rejection supports a negative target only when the original instructions or preceding observations made the call avoidable; an unexpected response to a reasonable probe is not itself a model error. The review requires observed evidence: a supported error and its witnessed recovery can yield separate negative and positive targets. An unobserved counterfactual action is never treated as a successful target. The review first identifies a supported operation in the original response, such as a tool name, an argument fragment, or a complete tool call, and assigns an evidence-derived direction $d_j \in \{-1, +1\}$. The recorded response is encoded with the student tokenizer, and the reviewed operation spans are aligned with token positions to construct a binary mask $m_{j,t} \in \{0, 1\}$. Tokens assigned to the reviewed spans receive $m_{j,t} = 1$; all other response tokens receive $m_{j,t} = 0$. Thus an operation, which may span multiple tokens, defines the region of supervision rather than a new replacement response. Each admitted mask is nonempty, $\sum_t m_{j,t} > 0$. The mask and direction are fixed before consulting teacher probabilities. Neighboring reasoning and operations remain in the causal context but receive no direct loss in the critical-token condition. The source-event, response, review, and mask identities are frozen together. Admission further requires the experience-conditioned probability change over the reviewed span to agree with d_j. Appendix B.4 follows one observed operation through experience extraction, review, and token masking; "critical operation" refers to the reviewed action, while "critical token" refers to its positions in the serialized response.

2.3 Distill Experience into Model Weights

The frozen deployed parent is the probability teacher, distinct from the annotation model. It scores the recorded response with the serialized context c_j containing the reviewed experience e_j, while the student scores the same tokens without that experience. Let θ_0 denote the deployed parent parameters and θ^- the student parameters immediately before an optimizer batch. The teacher is the frozen parent, $\pi_{\text{teacher}} = \pi_{\theta_0}$. At token t, let

$$\begin{aligned}\delta_{j,t}(\theta^-) = \; & \log \pi_{\text{teacher}}(y_{j,t} \mid c_j, x_j, y_{j,<t}) \\ & - \log \pi_{\theta^-}(y_{j,t} \mid x_j, y_{j,<t}).\end{aligned} \tag{1}$$

The distillation update admits a reviewed operation when the signed mean change over its critical tokens reaches the admission threshold:

$$d_j \frac{\sum_t m_{j,t}\, \delta_{j,t}(\theta_0)}{\sum_t m_{j,t}} \geq \kappa, \qquad \kappa = 0.02 \text{ nat/token}. \tag{2}$$

This gate checks that the experience produces a usable probability signal; it cannot establish that the eventual update improves task performance. Teacher probabilities and masks stay fixed within a round; the student term is recomputed before each optimizer batch. The implemented sampled-token loss is

$$\widehat{\mathcal{L}}_j(\theta) = -\frac{\sum_t m_{j,t}\; \mathrm{sg}(\delta_{j,t}(\theta^-)) \log \pi_\theta(y_{j,t} \mid x_j, y_{j,<t})}{\sum_t m_{j,t}}. \tag{3}$$

Here sg stops gradients through the signed weight. This fixed-response, masked surrogate inherits the teacher–student information pattern of On-Policy Context Distillation (OPCD) (Ye et al., 2026c); it is not an exact full-vocabulary Kullback–Leibler (KL) divergence or a continuously on-policy estimator. Positive recovery targets in the distillation update use the same experience-conditioned signed teacher as negative correction targets.

2.4 Reinforcement Learning from Recorded Turns

The RL update uses each original response as its reward unit. An annotation model judges that response from its causal prefix and recorded feedback, without benchmark grades. Three votes yield a signed reward or an abstention. A clipped offline objective applies each non-abstained reward to the complete recorded response, including reasoning. Thus, the two updates share the same access constraint but use different learning signals and supervision regions. RL neither samples new responses nor interacts with the task environment during training. Appendix E.1 gives the reward construction and objective.

2.5 Redeploy and Collect the Next Round

The updated model is served without c_j. Another round collects new client logs from the updated weights, then repeats review and training. The continual study uses a corrective-and-retention (CR) variant that combines correction of observed mistakes with preservation of successful recovery operations. Unlike the single-round distillation update, its retention targets use the bare, frozen preceding parent rather than the experience-conditioned teacher. For those rows, the signed weight in Equation (3) becomes $\log \pi_{\theta_{r-1}}(y_{j,t} \mid x_j, y_{j,<t}) - \log \pi_{\theta^-}(y_{j,t} \mid x_j, y_{j,<t})$; correction rows retain the experience-conditioned weight. Each round starts from the preceding model and collects new trajectories. Correction and retention targets and their training allocation are specified in Appendix B.

3 Experiments

We evaluate adaptation on the source domain, transfer to another benchmark, and adaptation across successive deployments.

Models and training data. We deploy Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026) through HermesAgent and OpenClaw, yielding four model–harness settings. Each setting supplies 512 ClawGym source-task trajectories for experience extraction and review. Source and evaluation task IDs are disjoint. An annotation model extracts and reviews experience; a frozen copy of the original Qwen model supplies teacher probabilities.

Comparisons. We compare the unadapted model (Base) with two offline updates from deployment experience: experience distillation (OPD) and turn-level reinforcement learning (RL). Both are evaluated on the same reported task panels.

Evaluation. ClawGym dev contains 100 tasks, and PinchBench contains 137 tasks (Bai et al., 2026; PinchBench, 2026). PinchBench uses GPT-5.5 as the evaluation judge and allows up to 150 model requests per task to accommodate multi-step execution. All reported scores are means over three evaluations on a 0–100 scale. Task selection, scoring, runtime limits, and annotation settings are detailed in Appendices A and E.

Setting	Panel	Base	OPD	RL	Δ_{OPD}	Δ_{RL}
4B / Hermes	PinchBench	60.24	**68.51**	66.51	**+8.27**	+6.27
	ClawGym dev	51.08	**60.38**	59.44	**+9.30**	+8.36
9B / Hermes	PinchBench	**78.22**	77.84	74.61	−0.38	−3.61
	ClawGym dev	61.53	**64.67**	62.60	**+3.14**	+1.07
4B / OpenClaw	PinchBench	74.30	**76.51**	75.38	**+2.21**	+1.08
	ClawGym dev	60.47	**65.36**	65.17	**+4.89**	+4.70
9B / OpenClaw	PinchBench	76.96	**78.53**	77.42	**+1.58**	+0.46
	ClawGym dev	66.26	**71.08**	70.15	**+4.82**	+3.89

Table 2: **Performance on ClawGym dev and PinchBench.** Three-evaluation means on a 0–100 scale; bold marks each row's best score and largest positive gain. Changes are relative to Base. Δ_{RL} uses displayed means; Δ_{OPD} uses unrounded means except for Hermes 4B / ClawGym dev, both Hermes 9B rows, and OpenClaw 4B / PinchBench.

Performance across settings. Table 2 reports the scores, and Figure 1 highlights adaptation on ClawGym dev. Both updates improve ClawGym dev performance in all four settings. OPD gains range from 3.14 to 9.30 points over Base, and its mean scores exceed those of the RL update. On PinchBench, it improves performance for Hermes 4B and both OpenClaw models, while Hermes 9B scores 0.38 points below Base. Thus, the source-domain gains extend to three of the four transfer settings.

Reinforcement learning update. The offline RL update assigns a reward to each recorded agent response using three annotation-model votes and trains on the full response, including reasoning, without new environment interaction. Table 2 compares Base, OPD, and RL under matched evaluation conditions; reward construction and the training objective are detailed in Appendix E.1.

4 Continual Adaptation

We study three complementary tracks: stationary adaptation, cross-agent transfer, and workload shift (Figure 3). The stationary track tests accumulation under a fixed harness and a fixed held-out 64-task panel. Cross-agent adaptation follows Hermes → OpenClaw → Hermes, testing adaptation to changes in tools, observations, and interaction rules, together with performance when returning to a previously used harness. Workload adaptation changes meetings, reports, and log investigation while keeping the OpenClaw harness fixed, testing adaptation to a changing task distribution.

We evaluate the corrective-and-retention (CR) variant, which corrects observed mistakes while retaining successful recovery operations, for three rounds. Each round acquires 64 new source-task logs from the preceding checkpoint, reviews them, updates those weights, and evaluates the resulting model without an experience prompt. Each track starts independently from the original Qwen3.5-4B model. Source and test identities are separated before collection.

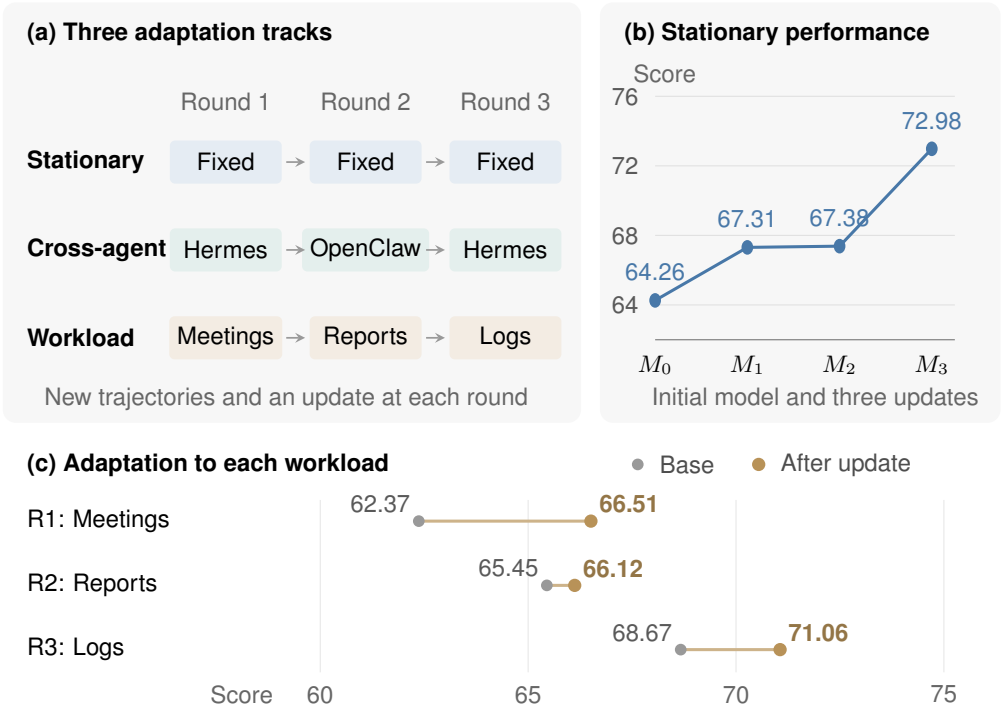


Figure 3: **Continual adaptation across three settings.** (a) Tracks start independently from the original model and vary the harness or workload. (b) Stationary scores on the fixed 64-task panel. (c) Base and post-update scores on each workload panel. Scores average three evaluations. Table 3 reports all tracks; Appendix D details score aggregation.

Accumulation and old-setting behavior. The stationary series scores 64.26, 67.31, 67.38, and 72.98 for M_0 through M_3. In the cross-agent track, final Hermes performance exceeds M_0 by 9.46 points, but is 0.51 points below the preceding M_2 on that same Hermes panel. The return to Hermes therefore does not itself establish recovery from the intervening update. The workload track improves over the matched M_0 reference in all three rounds, by 4.14, 0.67, and 2.39 points on meetings, reports, and log investigation, respectively. Final performance on old reports remains 2.71 points below M_0. Five predeclared old-setting retests include both improvements and regressions. All scheduled tasks and native failures remain in the denominators.

Track	Round / panel	M_0	M_r	Δ
Cross-agent	R1 / Hermes	53.51	**60.04**	**+6.53**
	R2 / OpenClaw	64.26	**66.22**	**+1.95**
	R3 / Hermes	53.51	**62.98**	**+9.46**
Workload	R1 / Meetings	62.37	**66.51**	**+4.14**
	R2 / Reports	65.45	**66.12**	**+0.67**
	R3 / Logs	68.67	**71.06**	**+2.39**
Stationary	R1	64.26	**67.31**	**+3.05**
	R2	64.26	**67.38**	**+3.12**
	R3	64.26	**72.98**	**+8.71**

Table 3: **Current-setting results across three rounds.** Means of three evaluations on 64 paired tasks; bold marks improvement over the matched M_0. M_0 is the initial model and M_r the model after round r. Each panel has its own reference.

Retested checkpoint	Old panel	$M_r - M_0$	$M_r - M_{r-1}$
Cross-agent M_2	Hermes	+9.98	+3.45
Cross-agent M_3	OpenClaw	+3.75	+1.79
Workload M_2	Meetings	+4.86	+0.72
Workload M_3	Meetings	+3.80	-1.05
Workload M_3	Reports	-2.71	-3.38

Table 4: **Performance on prior tasks.** Each row uses the original 64-task panel and harness for that setting. Point changes are computed from mean scores over three evaluations per checkpoint. The preceding-checkpoint contrast need not equal change from the first post-adaptation model.

5 Behavioral and Supervision Analysis

A change in native agent score is an outcome of the model and its harness, not a direct measurement of one model decision. In a retained same-weight `task_memory` pair, Qwen3.5-4B retrieves the correct deadline under both harnesses. Hermes writes its answer to `/answer.txt`, whereas OpenClaw writes to the required `/workspace/answer.txt`; the native grades are 0 and 100. Another retained pair reverses the ranking after OpenClaw only partially recovers from a tool error. These source-linked cases show how an unchanged checkpoint can take different executable paths, but they do not identify the dominant harness component or the population-level frequency of either failure. The complete selection rule and traces appear in Appendix G. Figure 4 shows the proportions of ClawGym tasks that improve, tie, or regress after adaptation; Appendix F provides the counts and task-family breakdowns. The changes are heterogeneous: 11 of 18 Hermes 4B planning tasks improve, while 12 of 17 OpenClaw 4B writing tasks regress. These family counts do not identify the cause.

From a source event to a training target. In stationary round three, an M_2 response calls `write` with a `path` but omits required `content`. A later response supplies the content. Review assigns 35 tokens in the malformed call to correction and two spans of 8 and 13 tokens in the observed recovery to retention. Correction uses experience-conditioned frozen-M_2 probabilities; retention uses the bare frozen M_2. Figure 2 illustrates the recorded operation and the two learning signals. Appendix B.4 gives the exact token mapping and procedural lesson. This example explains target construction; it does not attribute a held-out score gain to one operation.

Critical-token and complete-turn supervision. We compare supervision restricted to reviewed critical tokens with supervision over the complete selected response, including its reasoning and other operations. Both conditions use training seed 42. Table 5 reports mean scores over three evaluations of each checkpoint.

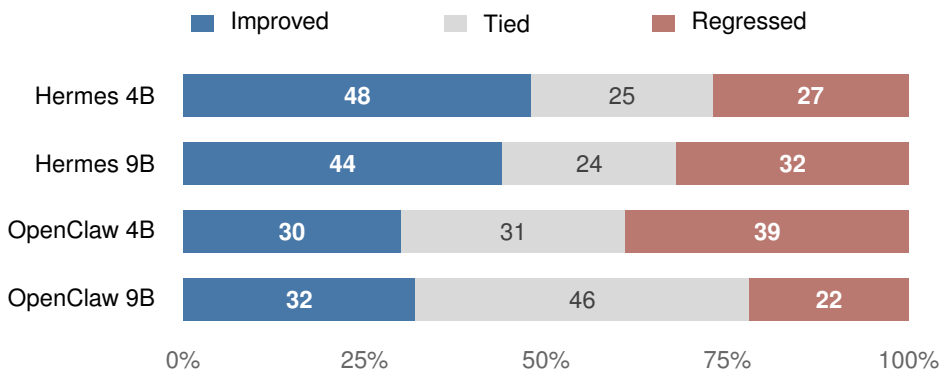


Figure 4: **Task-level changes after adaptation.** Each bar partitions the 100 ClawGym dev tasks by the sign of the OPD-minus-Base difference in three-evaluation mean grades. Labels give task counts; complete counts and family breakdowns appear in Appendix F.

Setting	Panel	Base	Critical token	Complete turn
Hermes 4B	ClawGym dev	51.08	**60.23**	57.12
	PinchBench	60.80	**68.62**	64.10
OpenClaw 9B	ClawGym dev	66.26	**71.08**	66.84
	PinchBench	76.58	**78.54**	77.08

Table 5: **Supervision-region comparison.** Three-evaluation means (0–100) on 100 ClawGym dev and 137 PinchBench tasks; bold marks each row's best score. Critical-token masks supervise 3,539 tokens for Hermes 4B and 2,301 for OpenClaw 9B; complete-turn masks supervise 186,742 and 97,641 tokens, respectively. Evaluation details are given in Appendix E.

The ClawGym Base references are shared with Table 2. PinchBench uses separately evaluated Base references: 60.80 for Hermes 4B and 76.58 for OpenClaw 9B, compared with 60.24 and 76.96 in the main table on the same 137 tasks.

Critical-token supervision yields higher mean scores than complete-turn supervision in all four comparisons while supervising fewer tokens. For OpenClaw 9B on ClawGym, it scores 71.08: 4.82 points above Base and 4.24 above complete-turn supervision. The comparison jointly varies the selected positions and the number of supervised tokens.

6 Related Work

Work agents and benchmarks. ReAct, AgentBench, and WebArena study interactive agents (Yao et al., 2022; Liu et al., 2023; Zhou et al., 2023); WorkArena, WorkArena++, and τ-bench cover enterprise workflows (Drouin et al., 2024; Boisvert et al., 2024; Yao et al., 2024). Software and computer-use agents extend this scope (Yang et al., 2024; Wang et al., 2024; Xie et al., 2024; Xu et al., 2024), while workplace benchmarks assess completion, unintended actions, and file dependencies (Styles et al., 2024; Styles & Miller, 2026; Tang et al., 2026; Li et al., 2026b). ClawGym, PinchBench, and SkillsBench evaluate workspace tasks, native agents, and reusable skills (Bai et al., 2026; PinchBench, 2026; Li et al., 2026a).

Agentic post-training. Agent Lightning, SkyRL-Agent, and Polar provide agent RL infrastructure (Luo et al., 2025; Cao et al., 2025; Xu et al., 2026a), complementing workplace-browser training (Zhuang et al., 2025; Vattikonda et al., 2025) and OpenClaw-RL's streamed-feedback updates (Wang et al., 2026). Frontier reports study agent RL and persistent rollouts (GLM-5 Team, 2026; Z.ai, 2026a;b; Kimi Team, 2026; Qwen Team, 2026), multi-teacher distillation (Xiaomi LLM-Core Team et al., 2026; Ma et al., 2026), and long-context modeling (DeepSeek-AI, 2026).

Harness adaptation and executable trajectories. Harness choice affects performance (Vats & Golev, 2026; Xu et al., 2026b). MemoHarness and SkillAdaptor adapt memory or external skills (Huang et al., 2026; Yu et al., 2026). EvoHarness-RL updates Qwen3-8B through supervised fine-tuning followed by Group Relative Policy Optimization (GRPO) to learn harness-use policies (Ning et al., 2026). Harness self-distillation internalizes reasoning support (Zhao et al., 2026b); Terminal-Universe reconstructs executable workspaces (Wu et al., 2026). Our setting retains the native harness and learns without reconstructing the client environment.

Experience reuse and self-distillation. Reflexion, ExpeL, and Voyager retain lessons or skills without weight updates (Shinn et al., 2023; Zhao et al., 2023; Wang et al., 2023). Online Experiential Learning (OEL) and LLM-as-a-Coach internalize procedural experience (Ye et al., 2026b;a). On-Policy Context Distillation (OPCD) and Self-Distilled Reasoner use privileged context (Ye et al., 2026c; Zhao et al., 2026a). Self-Distillation Policy Optimization (SDPO) distills feedback-conditioned predictions (Hübotter et al., 2026); user-interaction alignment applies this principle to follow-up messages (Kleine Buening et al., 2026). On-Policy Self-Distillation Fine-Tuning (SDFT) uses demonstration-conditioned teachers for continual learning (Shenfeld et al., 2026). Replayed-Prefix On-Policy Distillation (ReOPD) samples student continuations from recorded teacher prefixes without environment interaction (Liao et al., 2026). Experiential Reinforcement Learning (ERL) combines environmental feedback, reflection, renewed attempts, and consolidation (Shi et al., 2026). Here, both updates use fixed, logged work-agent responses: distillation targets reviewed operation tokens, while RL rewards complete turns. Appendix B specifies the sampled-token objective.

7 Conclusion

We study harness adaptation from deployment experience when the original client environment is unavailable during training. Two offline update rules use the available records: experience distillation learns from reviewed operations and procedural lessons, while turn-level reinforcement learning learns from rewards assigned to recorded responses. Both update model weights without new environment interaction, and the adapted agent returns to its native harness without an experience prompt.

Both updates improve performance across the four ClawGym dev settings. Experience distillation also improves three PinchBench settings, and its corrective-and-retention variant improves over the initial model in stationary, cross-agent, and changing-workload settings. The supervision-region comparison favors critical-token over complete-turn supervision in all four evaluated model–benchmark combinations. Together, these results show that deployment experience can support work-agent harness adaptation through different offline learning signals, and motivate further work on transfer and retention across changing environments.

8 Limitations

A returned trajectory supports only operations and outcomes visible in the record. Missing observations, annotation errors, and ambiguous recoveries limit credit assignment, and a frozen teacher cannot validate an unexecuted action. Probability-based admission restricts supervision to spans that respond to the extracted experience. The resulting objective uses fixed recorded responses.

All reported scores average three evaluations. Repeated evaluation measures execution variability; broader coverage of training variability and deployment settings remains valuable. PinchBench task selection and the evaluation judge are specified in Appendix E.

Transfer to PinchBench varies across model–harness settings, and gains are not uniform across successive updates. Cross-benchmark performance does not directly measure catastrophic forgetting. The continual-learning panels and old-setting retests cover selected aspects of adaptation and retention. Old-task retests reveal regressions after workload shifts. The supervision-region comparison uses one training seed per setting. Complete-turn supervision changes both token exposure and relative loss weights.

AI USE STATEMENT

Generative AI tools assisted with language polishing, manuscript organization, literature summaries, figure preparation, and discussion of the interpretation and presentation of experimental results. The use of language models within the learning and evaluation protocols is described in Sections 2 and 3 and Appendix E. The authors are responsible for the final text, claims, figures, and reported results.

REPRODUCIBILITY STATEMENT

Section 2 and Appendix B describe the learning objectives, evidence review, and operation-to-token mapping. Appendix E specifies the training and evaluation protocols, including task selection and runtime limits. Additional continual-adaptation results and task-level summaries are provided in Appendices D and F.

REFERENCES

Fei Bai, Huatong Song, Shuang Sun, Daixuan Cheng, Yike Yang, Chuan Hao, Renyuan Li, Feng Chang, Yuan Wei, Ran Tao, Bryan Dai, Jian Yang, Wayne Xin Zhao, and Ji-Rong Wen. ClawGym: A scalable framework for building effective claw agents. *arXiv preprint arXiv:2604.26904*, 2026.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, et al. WorkArena++: Towards compositional planning and reasoning-based common knowledge work tasks, 2024. URL https://arxiv.org/abs/2407.05291.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, et al. Skyrl-agent: Efficient rl training for multi-turn llm agent, 2025. URL https://arxiv.org/abs/2511.16108.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, et al. WorkArena: How capable are web agents at solving common knowledge work tasks?, 2024. URL https://arxiv.org/abs/2403.07718.

GLM-5 Team. GLM-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Yue Huang, Wenjie Wang, Han Bao, Yuchen Ma, Xiaonan Luo, Yi Nian, Haomin Zhuang, Zheyuan Liu, Yue Zhao, and Xiangliang Zhang. Memoharness: Agent harnesses that learn from experience, 2026. URL https://arxiv.org/abs/2607.14159.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026. URL https://arxiv.org/abs/2601.20802.

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

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions. *arXiv preprint arXiv:2603.12273*, 2026. URL https://arxiv.org/abs/2603.12273.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks, 2026a. URL https://arxiv.org/abs/2602.12670.

Yuetai Li, Yichen Feng, Zhangchen Xu, Zixian Ma, Kaiyuan Zheng, et al. JobBench: Aligning agent work with human will, 2026b. URL https://arxiv.org/abs/2605.26329.

Baohao Liao, Hanze Dong, Christof Monz, Xinxing Xu, Li Dong, and Furu Wei. Multi-turn on-policy distillation with prefix replay. *arXiv preprint arXiv:2607.04763*, 2026. URL https://arxiv.org/abs/2607.04763.

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. *arXiv preprint arXiv:2308.03688*, 2023. URL https://arxiv.org/abs/2308.03688.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning, 2025. URL https://arxiv.org/abs/2508.03680.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, et al. MOPD: Multi-teacher on-policy distillation for capability integration in LLM post-training, 2026. URL https://arxiv.org/abs/2606.30406.

Xuying Ning, Dongqi Fu, Tianxin Wei, Hanqing Zeng, Yuanchen Bei, Bingxuan Li, Zihao Li, Qifan Wang, Xiang Shen, Yifan Wu, Jiayi Liu, Hong Li, Yinglong Xia, Xiangjun Fan, Hanghang Tong, and Jingrui He. Evoharness-rl: Learning self-evolving runtime harness for long-horizon llm agents, 2026. URL https://arxiv.org/abs/2608.05446.

PinchBench. PinchBench. GitHub repository, version 2.0.0, 2026. URL https://github.com/pinchbench/skill/tree/v2.0.0.

Qwen Team. Qwen3.5: Towards native multimodal agents. Official model report, 2026. URL https://qwen.ai/blog?id=qwen3.5.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026. URL https://arxiv.org/abs/2601.19897.

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential reinforcement learning. *arXiv preprint arXiv:2602.13949*, 2026. URL https://arxiv.org/abs/2602.13949.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, et al. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Olly Styles and Sam Miller. WorkBench revisited: Workplace agents two years on, 2026. URL https://arxiv.org/abs/2606.13715.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. WorkBench: a benchmark dataset for agents in a realistic workplace setting, 2024. URL https://arxiv.org/abs/2405.00823.

Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Yukai Wu, et al. Workspace-Bench 1.0: Benchmarking AI agents on workspace tasks with large-scale file dependencies, 2026. URL https://arxiv.org/abs/2605.03596.

Naman Vats and Oleg Golev. The scaffold effect in coding agents: Harness choice as a hidden variable in coding-agent evaluation, 2026. URL https://arxiv.org/abs/2607.22585.

Dheeraj Vattikonda, Santhoshi Ravichandran, Emiliano Penaloza, Hadi Nekoei, et al. How to train your LLM web agent: A statistical diagnosis, 2025. URL https://arxiv.org/abs/2507.04103.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, et al. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for ai software developers as generalist agents, 2024. URL https://arxiv.org/abs/2407.16741.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking, 2026. URL https://arxiv.org/abs/2603.10165.

Jie Wu, Zhenru Zhang, Beichen Zhang, Xuwu Wang, Yuhui Su, Mouxiang Chen, Peng Wang, Zhihai Wang, Que Shen, Hao Zhou, An Yang, Fei Huang, Yujiu Yang, and Dayiheng Liu. Terminal-Universe: Turning agent trajectories into scalable terminal environments, 2026. URL https://arxiv.org/abs/2609.04148.

Xiaomi LLM-Core Team, Bangjun Xiao, Bingquan Xia, et al. MiMo-V2-Flash technical report, 2026. URL https://arxiv.org/abs/2601.02780.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

Binfeng Xu, Hao Zhang, Shaokun Zhang, Songyang Han, Mingjie Liu, Jian Hu, Shizhe Diao, Zhenghui Jin, Yunheng Zou, Michael Demoret, Jan Kautz, and Yi Dong. Polar: Agentic rl on any harness at scale, 2026a. URL https://arxiv.org/abs/2605.24220.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, et al. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks, 2024. URL https://arxiv.org/abs/2412.14161.

Tianshi Xu, Huifeng Wen, and Meng Li. Adapting the interface, not the model: Runtime harness adaptation for deterministic llm agents, 2026b. URL https://arxiv.org/abs/2605.22166.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. *arXiv preprint arXiv:2405.15793*, 2024. URL https://arxiv.org/abs/2405.15793.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, et al. ReAct: Synergizing reasoning and acting in language models, 2022. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Tianzhu Ye, Li Dong, Guanheng Chen, He Zhu, Xun Wu, Shaohan Huang, and Furu Wei. LLM-as-a-coach: Experiential learning for non-verifiable tasks. *arXiv preprint arXiv:2607.18110*, 2026a.

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online experiential learning for language models. *arXiv preprint arXiv:2603.16856*, 2026b.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026c.

Zhuoyun Yu, Xin Xie, Wuguannan Yao, Chenxi Wang, et al. SkillAdaptor: Self-adapting skills for LLM agents from trajectories, 2026. URL https://arxiv.org/abs/2606.01311.

Z.ai. GLM-5.2. Official model card, 2026a. URL https://huggingface.co/zai-org/GLM-5.2.

Z.ai. GLM-5.3: Overview. Official developer documentation, 2026b. URL https://docs.z.ai/guides/llm/glm-5.3.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, et al. ExpeL: LLM agents are experiential learners, 2023. URL https://arxiv.org/abs/2308.10144.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026a. URL https://arxiv.org/abs/2601.18734.

Zhengyang Zhao, Lu Ma, and Wentao Zhang. Training with harnesses: On-policy harness self-distillation for complex reasoning. *arXiv preprint arXiv:2605.08741*, 2026b. URL https://arxiv.org/abs/2605.08741.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, et al. WebArena: A realistic web environment for building autonomous agents, 2023. URL https://arxiv.org/abs/2307.13854.

Yuchen Zhuang, Di Jin, Jiaao Chen, Wenqi Shi, et al. WorkForceAgent-R1: Incentivizing reasoning capability in LLM-based web agents via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.22942.

A Benchmark Composition and Task Examples

The primary update uses trajectories returned from ClawGym source tasks, collected separately for each model–harness setting. The evaluated ClawGym dev task IDs do not occur among those source IDs. ClawGym tasks can specify several verifiable output requirements and receive native partial grades, which makes partial task completion visible. Grades measure the submitted artifacts, rather than individual tool calls.

Work-agent harnesses and source preparation. HermesAgent 0.15.1 and OpenClaw 2026.5.27 are open-source work-agent harnesses (HermesAgent and OpenClaw). Both run the Qwen3.5 model family on file and web tasks, while retaining their own prompts, tool schemas, observations, and termination behavior. We preserve those native interfaces, so cross-harness comparisons measure the combined deployment configuration. Training configurations are specified in Appendix C.

ClawGym dev composition. We group the 100 shared tasks into six families for analysis, separately from ClawGym's native macro-category labels. Figure 5(a) shows the exact task counts.

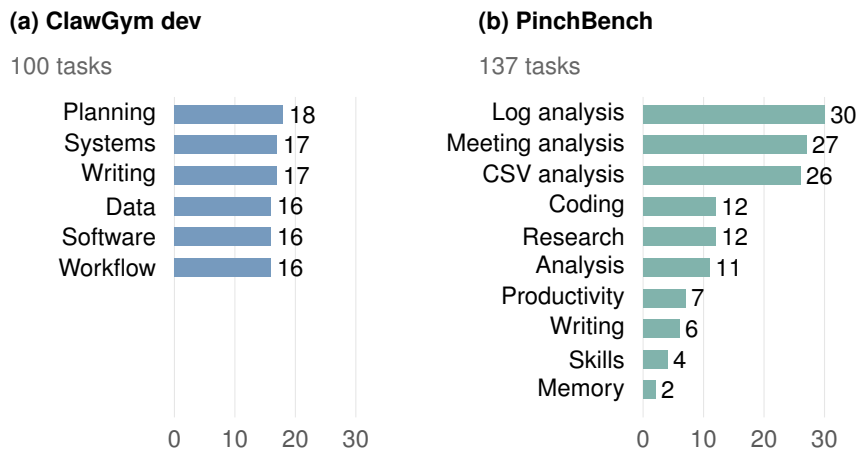


Figure 5: **Evaluation task composition.** Counts by family in ClawGym dev and by category in PinchBench. Both panels use the same task-count scale.

For example, one ClawGym dev *data* task asks the agent to validate a phasor-measurement CSV using a supplied configuration. It requires both a flagged-rows CSV with per-row issue codes and a JSON summary containing exact counts for frequency, magnitude, angle, missing-row, and timestamp-gap conditions. This evaluation task illustrates multiple verifiable deliverables.

Cross-benchmark PinchBench panel. Our PinchBench evaluation uses 137 text tasks. The excluded tasks and their selection rationale are detailed in Appendix E. Figure 5(b) uses the original task-frontmatter categories, with capitalization normalized. PinchBench tasks and scorers are distinct from the ClawGym training source. PinchBench measures transfer to a distinct benchmark; it is not a forgetting measure. Results refer to this text-task panel. Where public-web search is enabled, we use Serper Search. In HermesAgent, direct `web_extract` calls use Firecrawl; OpenClaw's native `web_fetch` is a different tool path.

As one example, `task_meeting_tech_action_items` asks the agent to read a supplied meeting transcript and write `action_items.md`, listing the owner, action, and deadline for each extracted item. This task illustrates document reading and artifact delivery; its prompt and category are preserved in the task index.

B Implementation Details of Deployment Adaptation

This appendix specifies target construction, the training objective, and the corrective-and-retention (CR) variant used in the continual study.

B.1 Recover Evidence from Returned Trajectories

Within a round, the deployed parent π_{θ_0} produces trajectories containing causal prefixes x_j, original responses y_j, and subsequent observations o_j. Training uses the model-visible record, excluding benchmark grades and hidden verifier outputs. The provider cannot query hidden state or execute alternative actions in the client's workspace.

API records preserve response text and tool-argument bytes but omit sampling token IDs and behavior-policy probabilities. We encode them with a frozen tokenizer and canonical serialization. Content hashes bind each target to its source response, original prefix, review, and token mask. Filtering selects responses without altering their operations or causal context.

The annotation model extracts a procedural lesson e_j, its applicability, and supporting source events. Task answers, output payloads, and ephemeral paths cannot substitute for a reusable procedure. Later observations may inform this annotation but never enter the student's earlier prefix.

B.2 Localize Supported Operations

A separate full-trajectory review verifies the lesson and implicated operation. It excludes fabricated authoritative inputs, weakened supplied tests, and unresolved source integrity. An error is eligible for correction only if it was avoidable given the original instructions and preceding observations; infrastructure failures are treated separately. A completion claim or successful process exit alone does not establish a correct result.

Correction and observed recovery form separate targets. Before probability scoring, review fixes the direction $d_j \in \{-1, +1\}$ and a nonempty binary mask $m_{j,t}$. The span can cover a tool name, an argument, or a complete call, as supported by the evidence. It is neither expanded to neighboring calls nor shortened to increase the probability margin. In the critical-token condition, reasoning remains in context with zero direct loss; complete-turn supervision includes reasoning (Section 5).

B.3 Distill Experience into Model Weights

The frozen parent is the probability teacher, distinct from the annotation model. With reviewed experience serialized as c_j, the token weight is

$$\delta_{j,t}(\phi) = \log \pi_{\text{teacher}}(y_{j,t} \mid c_j, x_j, y_{j,<t}) - \log \pi_\phi(y_{j,t} \mid x_j, y_{j,<t}). \tag{4}$$

Primary OPD uses the signed mean gate in Equation 2 with a threshold of 0.02 nat per supervised token. Both correction and positive recovery targets use the experience-conditioned teacher. CR correction instead uses the summed-span gate

$$d_j \sum_{t=1}^{L_j} m_{j,t}\, \delta_{j,t}(\theta_0) \geq \kappa_{\text{CR}}, \qquad \kappa_{\text{CR}} = 0.1 \text{ nat}. \tag{5}$$

Admission requires a probability signal consistent with the reviewed direction; it does not certify an eventual task improvement.

Teacher probabilities and masks remain frozen within a round. Student scores are recomputed at θ^- before each optimizer batch, giving

$$\widehat{\mathcal{L}}_j(\theta) = -\frac{1}{\sum_t m_{j,t}} \sum_t m_{j,t}\, \text{sg}\big(\delta_{j,t}(\theta^-)\big) \log \pi_\theta(y_{j,t} \mid x_j, y_{j,<t}). \tag{6}$$

The stop-gradient operator fixes the signed weight. Positive and negative weights respectively increase and decrease the selected token probability; token signs can differ within an admitted operation. Both variants use a unit distillation coefficient and the same batch reduction. This fixed-response, masked objective follows OPCD's teacher–student information pattern (Ye et al., 2026c), but is neither a full-vocabulary KL divergence nor an unbiased continuously on-policy estimate. The RL update is specified in Appendix E.1.

Continual correction and retention. CR correction targets are reviewed negative operations admitted by Equation 5. Retention targets are witnessed successful recoveries, with the completed-call boundary included in the operation mask. They require no positive experience-uptake gate and

use

$$\delta^{\text{ret}}_{j,t}(\theta^-) = \log \pi_{\theta_{r-1}}(y_{j,t} \mid x_j, y_{j,<t}) - \log \pi_{\theta^-}(y_{j,t} \mid x_j, y_{j,<t}), \tag{7}$$

where the preceding deployed model is frozen and receives no privileged experience. This weight replaces $\delta_{j,t}$ in Equation 6 for retention targets. There is no separate retention multiplier: relative exposure depends on target counts and masks. Each round resets optimizer and random-number state and trains for one epoch without duplicating targets.

Variant	Targets	Teacher or reward	Loss span
Primary OPD	Reviewed original operations	Frozen M_0 with experience	Admitted operation tokens
CR	Mistakes and observed recoveries	Parent with experience for correction; bare parent for retention	Reviewed operation tokens
Offline RL	Original responses with valid rewards	Three annotation-model votes; M_0 ratio/KL reference	Whole response, including reasoning

Table 6: **Training variants.** All optimize on recorded interactions without environment access. CR collects new deployment trajectories between rounds. Annotation, probability scoring, and benchmark evaluation have distinct roles.

B.4 A Source Event, Experience, and Critical Tokens

In stationary round three, an M_2 response calls `write` with a `path` but omits required `content`. Tool feedback identifies the malformed call; a later response supplies the content. The extracted lesson states:

> **Trigger:** "A file must be created or overwritten using the write tool."
> **Procedure:** "Provide both required arguments in the same tool call: path with the relative or absolute target path, and content with the full file contents to write. If the content is large, still include it in the content argument rather than narrating that it will be supplied."

Review assigns the malformed operation a negative direction and the observed recovery a retention target. Canonical serialization maps the former to 35 correction tokens and the latter to two spans of 8 and 13 tokens. Other tokens remain in the causal context with zero direct loss. Correction uses the experience-conditioned frozen M_2; retention uses the bare frozen M_2. The example illustrates local target construction, without attributing a whole-task outcome or evaluation gain to this one operation.

B.5 Redeploy and Collect the Next Round

For learning method m and deployment round r, the process is

$$\theta^{(m)}_{r-1} \xrightarrow[\mathcal{S}_r, H_r]{\text{deploy}} \mathcal{T}^{(m)}_r \xrightarrow{\text{review}} \mathcal{D}^{(m)}_r \xrightarrow{\text{update}} \theta^{(m)}_r. \tag{8}$$

Here $\mathcal{S}_r$ is the source-task batch and H_r is the deployed harness. Each branch collects trajectories from its preceding model, which also initializes the update and supplies the frozen reference for that round.

Both the single-round distillation update and its CR variant deploy without experience prompts. Evaluation tasks and outcomes remain outside collection, target construction, and checkpoint selection.

C Experimental Configuration

The experience-distillation experiments use DeepSeek V4 Flash for experience extraction and review and a frozen Qwen parent for target probabilities. HermesAgent 0.15.1 and OpenClaw 2026.5.27 retain their native tool interfaces. Each condition starts independently from its original Qwen3.5 checkpoint. The frozen original checkpoint supplies experience-conditioned teacher log-probabilities;

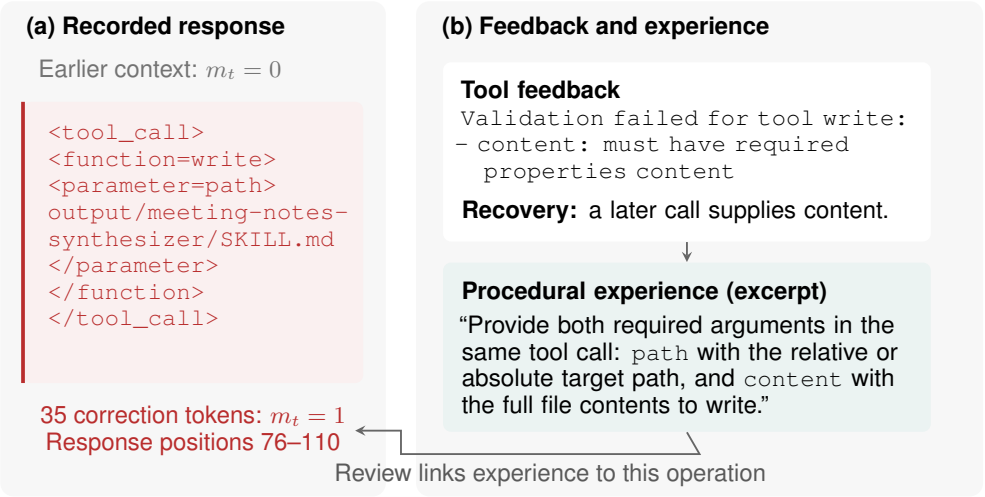


Figure 6: **From a recorded operation to supervised tokens.** A stationary round-three correction example: the call omits required `content`, and review links the observed rejection and recovery to a file-writing procedure. Red marks the 35 tokens receiving correction loss; earlier reasoning and tool feedback receive none. The path wraps for display.

training applies Equation 3 only at admitted critical-token positions. The Hermes 4B and OpenClaw settings train for one epoch in the recorded data order, with global batch size 16, learning rate 5×10^{-7}, and training seed 42. The Hermes 9B model uses 105 sources, 128 targets, and eight optimizer steps. We evaluate the final training checkpoint in each setting. Base, OPD, and RL scores are means over three evaluations in each setting. Task selection and evaluation budgets are specified in Appendix E.

D Continual-Adaptation Configuration

Each measured corrective-and-retention (CR) track starts independently from Qwen3.5-4B, acquires 64 new source-task trajectories per round, and evaluates on a 64-task panel after each of three updates. Tables 3 and 4 report current-setting and old-setting results, respectively; scores average three evaluations.

E Evaluation Protocol and Controls

PinchBench task selection. We evaluate a common panel of 137 text tasks from the 147-task inventory. We exclude six tasks involving image or media processing, browser-based end-to-end testing, or a greeting-only check, and four tasks that depend on local mock services for GitHub and Google Workspace. All methods are compared on the same 137 tasks.

Runtime and scoring. Each PinchBench task allows at most 150 model requests and a 3,600-second safety limit. The request budget accommodates extended agent execution and reduces premature termination. Requests are counted before forwarding to the model, including actor requests, retries, continuations, and context-compression requests; tool calls are not counted as model requests. GPT-5.5 serves as the evaluation judge. Scores are native task grades multiplied by 100 and averaged over three evaluations for every setting and method. Failures remain in the denominator, and benchmarks are reported separately.

E.1 Offline Turn-Level Reinforcement Learning

Each training example is one deduplicated original response from a recorded agent turn. DeepSeek V4 Flash judges the response using its causal prefix and already recorded feedback, without benchmark

grades. Three independently requested votes use seeds 42, 43, and 44. A vote labels the observed response $+1$ for supported correct behavior, -1 for an avoidable error, or abstains when the evidence does not support either judgment. At least two valid, non-abstaining votes must agree on the same sign; otherwise the turn abstains and receives a zero loss mask. Invalid API, schema, or evidence-citation responses are failed calls rather than votes. The voting procedure permits at most three attempts per required seed; a turn with an unresolved invalid seed is excluded from training and recorded separately from abstentions.

For an admitted response y_j, let $r_j \in \{-1, +1\}$ be its turn reward and $\rho_{j,t}(\theta) = \exp[\log \pi_\theta(y_{j,t} \mid x_j, y_{j,<t}) - \log \pi_{M_0}(y_{j,t} \mid x_j, y_{j,<t})]$ the sampled-token ratio to the original frozen checkpoint. With $\epsilon_- = 0.20$ and $\epsilon_+ = 0.28$, the implemented offline policy term is the per-response mean of

$$\ell_{j,t}^{\text{clip}}(\theta) = \max\{-r_j \rho_{j,t}(\theta), -r_j \,\text{clip}(\rho_{j,t}(\theta), 0.80, 1.28)\}\,. \tag{9}$$

The native loss adds 0.02 times its sampled-token fixed-reference approximate KL term; entropy coefficient is zero. These are canonically re-encoded logged responses, so the reference probabilities are not the unrecorded behavior-policy probabilities and Equation 9 is not an unbiased on-policy proximal policy optimization (PPO) estimator. The turn reward is broadcast to all tokens of that original response, including reasoning; neither advantage whitening nor new environment rollouts are used during optimization.

All four RL models start from the corresponding original M_0, with a fresh optimizer, learning rate 5×10^{-7}, global batch size 16, one epoch, training seed 42, and the final epoch checkpoint.

Table 2 reports Base, OPD, and RL scores on the same task panels.

Supervision-region evaluation. The ablation uses the same 137 PinchBench task IDs across methods, with a maximum of 150 model requests per task. Scores average three evaluations. ClawGym uses the Base references from Table 2; PinchBench uses separately evaluated Base references. For Hermes 4B on the 100-task ClawGym panel, one task required a scorer-output format correction; the other 99 task scores were retained.

F Task-Level Changes in ClawGym dev

We compare the same task IDs within each model–harness setting. A task is *improved*, *tied*, or *regressed* according to the sign of its final-minus-Base mean native grade, averaging three evaluations per task for each model. All scheduled tasks, including zero grades, enter these summaries.

Panel	Setting	Improved	Tied	Regressed	Mean Δ
ClawGym dev	4B / Hermes	48	25	27	+9.30
	9B / Hermes	44	24	32	+3.14
	4B / OpenClaw	30	31	39	+4.89
	9B / OpenClaw	32	46	22	+4.82

Table 7: **Signs of per-task grade changes.** Counts partition the reported panel in each setting. Mean Δ is in grade points, matching Table 2; a positive mean does not imply that every task improved.

Where the changes occur. Table 8 partitions ClawGym tasks into improved/tied/regressed counts. For example, 11 of 18 planning tasks improve for Hermes 4B, while seven of 16 software tasks regress. These descriptive slices do not isolate a causal effect of task type.

Illustrative score transitions. One ClawGym software task requires tests for an HTML press-item extractor and a comparison with supplied expected records. Hermes 4B changes from 0 to 100 on its native grade. The workflow task asks for three addressed chat updates and a coordination plan; Hermes 9B changes from 0 to 91.67. On a task requiring a TIFF preprocessing queue, routing edits, and documentation, OpenClaw 4B changes from 0 to 100. These are evaluation tasks, not the source trajectories used for the update.

Family	Tasks	Hermes 4B	Hermes 9B	OpenClaw 4B	OpenClaw 9B
ClawGym					
Data	16	8/4/4	6/2/8	5/5/6	4/7/5
Planning	18	11/3/4	11/3/4	9/5/4	5/9/4
Software	16	4/5/7	6/6/4	4/4/8	4/10/2
Systems	17	10/3/4	5/7/5	5/9/3	6/8/3
Workflow	16	6/6/4	6/3/7	4/6/6	6/6/4
Writing	17	9/4/4	10/3/4	3/2/12	7/6/4

Table 8: **Improved/tied/regressed task counts by family.** Each cell shows these three counts in that order; they sum to the Tasks column. ClawGym families are the analysis categories defined here.

An example ClawGym transition. For the Hermes 9B coordination task above, the Base trajectory reads the chat and brief, then ends without writing either required output file. The OPD checkpoint reads the same inputs and writes `output/memory/kiro-chat.md` and `output/coordination/plan.md`. The native grade changes from 0 to 91.67. This is an observed artifact-delivery difference, not attribution to one training experience.

G Trajectory Review and Qualitative Cases

Same-weight comparisons examine the harness; before/after comparisons examine learning within a harness. The cases below link grades to visible operations.

G.1 Review Record

Each pair records task and fixture hashes, model and harness versions, budgets, event identifiers, the first supported divergence, failure labels, feedback, recovery, termination, grades, and uncertainty. Attribution requires an avoidable decision and an observed consequence supported by the complete trajectory and workspace. Later evidence cannot be treated as information available before the action; unsupported or missing evidence leaves the case unresolved.

G.2 Same-Weight Harness Cases

We compare the original Qwen3.5-4B under both harnesses. Among 12 pairs whose complete transcripts fit the review budget and whose workspace views are unchanged, we select the largest score difference in each direction, breaking ties by task ID. All four transcripts, prompts, graders, and workspaces were reviewed. These are illustrative evaluation cases, not training targets or estimates of failure frequency.

Correct retrieval, wrong delivery path. In `task_memory`, the agent must read `notes.md` and save the beta-release deadline to `answer.txt`. Both agents retrieve the correct date, June 1, 2024, from `/workspace/notes.md`. Hermes writes to `/answer.txt`; its tool acknowledges 51 bytes, but the grading workspace contains no answer file. OpenClaw writes to `/workspace/answer.txt`, which is present in the retained workspace. The grader assigns zero on all five criteria when the required file is absent: Hermes scores 0 and OpenClaw 100. This supports an output-path diagnosis. The external file's final state was not retained, and the evidence does not isolate the prompt or tool feature responsible for the path choice.

Incomplete recovery leaves a required file missing. In `task_files`, the requested outputs are `src/main.py`, a README, and a root-level `.gitignore`. OpenClaw tries to create `src` using `write` with empty content, receives a parameter error, and then writes an unintended `hello world` file. Recovery via `exec` creates `src/.gitignore`, leaving the root file absent. Hermes creates the root file but omits the required literal `__pycache__` entry. Hermes therefore meets six of seven criteria (85.71), versus five of seven (71.43) for OpenClaw. The tool feedback and file

locations support incomplete recovery; the opposite score direction also rules out a universal harness ranking from these cases.