ENVIRONMENT-FREE LEARNING FROM AGENT TRAJECTORIES WITH BOTTLENECK UPSAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Language agents are increasingly deployed in complex environments, and every deployment leaves behind trajectories. Rebuilding the environment for post-training, as reinforcement learning requires, is often impractical: it may be proprietary, involve real users, or be too expensive to run again. Environment-free learning, which improves an agent from these trajectories alone, is therefore an important problem. A natural approach is to relabel the trajectories: at every step, a teacher shows what should have been done, and the agent is trained to follow. In context distillation (CD) and on-policy context distillation (OPCD), the teacher is the same model conditioned on experience summarized from the trajectories. We find that this recipe works well on some benchmarks but delivers limited gains on others, and we trace the shortfall to the trajectories. The deployed agent that collected them is weak, so few episodes reach the later stages of a task, and the training data hold few steps from them. For example, on a shopping task, few episodes reach a product page, so the trained agent does not learn what to do there, such as choosing options and placing the order. We call the first stage that few episodes reach the *bottleneck*. We propose CDUP (Context Distillation with bottleneck UPsampling), a simple and effective recipe that has an LM agent segment the trajectories into task stages, locates the bottleneck from the share of episodes reaching each stage, and upsamples the steps from it onward at a fixed ratio. On WebShop, CDUP improves success over CD by 16.4 percentage points with Llama-3.2-3B and 3.1 with Qwen2.5-3B, and on ScienceWorld by 16.9 and 9.9. Bottleneck upsampling also raises WebShop success under OPCD and with an external expert teacher. Code is available at https://anonymous.4open.science/r/cdup.

1 INTRODUCTION

Language-model agents are now deployed at scale (Yang et al., 2026b), browsing the web (Hu et al., 2025a; Xie et al., 2026), writing code (Baumann et al., 2026; Cursor Research et al., 2026), operating computers (Yang et al., 2026a; Bechard et al., 2026), and controlling robots (Physical Intelligence et al., 2026; Wu et al., 2026c). Each deployment leaves behind trajectories that come at no extra cost and record where the agent succeeds and fails (Corrêa et al., 2025), yet the established recipes for improving an agent return to the environment during training. Reinforcement learning collects fresh rollouts and rewards (Feng et al., 2025; Wu et al., 2026b;a), and trajectory distillation lets a stronger policy interact with the task (Chen et al., 2023; Zeng et al., 2024). Returning to a deployment environment is often impractical (Levine et al., 2020; Chen et al., 2026b). It may be too large, complex, or proprietary to rebuild (Jain et al., 2025); its actions may have lasting side effects (Ruan et al., 2024); and its users cannot be faithfully replayed (Yao et al., 2025; Zhou et al., 2026a). We therefore study *environment-free learning from trajectories*: given the trajectories a deployed agent has already collected and no further access to the environment, how can the agent improve?

Under this setting, the natural way to learn is to relabel the trajectories (Figure 1): at every step of the trajectory, a teacher says what should have been done, and the policy is trained to follow. The question is who the teacher is. A recent line of work takes the teacher from the trajectories themselves, summarizing them into *experience* (Ye et al., 2026b; Gou et al., 2026; Ye et al., 2026c). The deployed model conditioned on that experience, the *self-teacher*, teaches a student that sees only

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

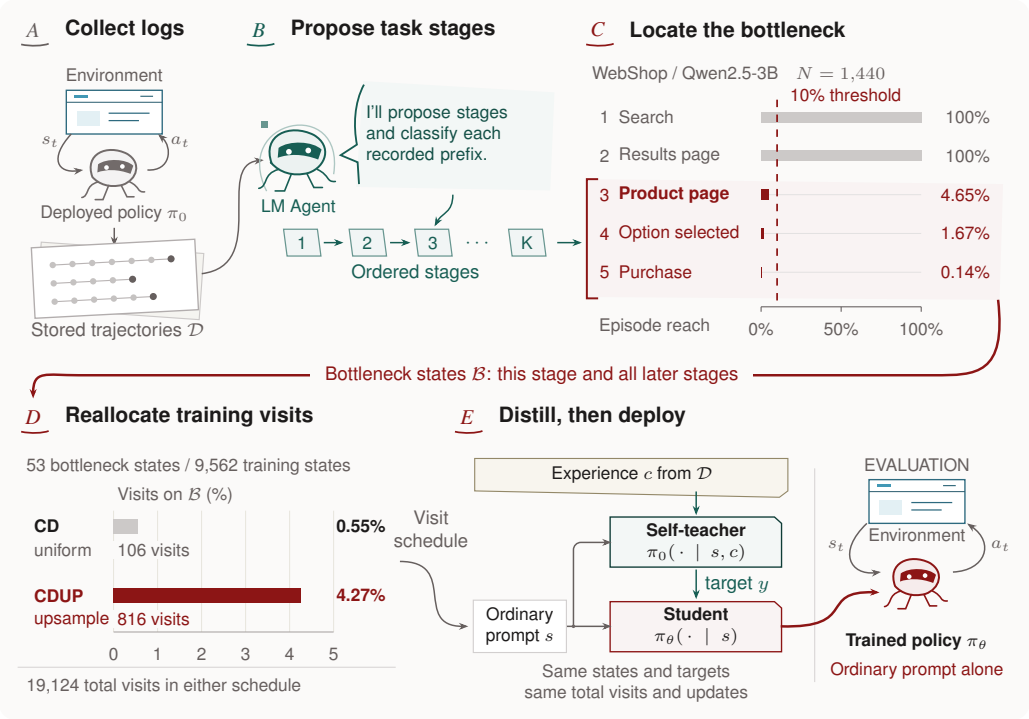


Figure 1: Our proposed method, CDUP, from deployment to the trained policy, in order A–E. **(A)** Collect and store trajectories. **(B)** An LM agent reads them and proposes ordered stages and a classifier; the dialogue is illustrative. **(C)** Find the first stage reached by few episodes and mark its states and all later states. In the WebShop example, this is the product page. **(D)** Reallocate a fixed training budget to raise the share of these states in training. **(E)** Distill the same model's experience-conditioned targets into a student that receives only the ordinary prompt, then deploy it alone. Stage selection reads the recorded interactions; action targets come from the self-teacher.

the ordinary prompt. The supervised form is context distillation (CD) (Snell et al., 2022; Gou et al., 2026), and the on-policy form, in which the student samples its own responses and the self-teacher scores them, is on-policy context distillation (OPCD) (Ye et al., 2026c; Agarwal et al., 2024; Zhao et al., 2026; Shenfeld et al., 2026). We analyze where this recipe falls short.

We find that the gains of this recipe are strikingly uneven. On ALFWorld with Qwen2.5-3B, CD recovers most of the gain that the experience delivers in context (Table 1); on WebShop it recovers a third, and with Llama-3.2-3B almost none. The distilled WebShop policies get most of the way to a purchase but rarely complete it.

The trajectories point to a cause of this shortfall (Figure 1C). A weak deployed agent spends most of its steps in the early stages of a task, and few episodes reach the later ones. Uniform training, the standard choice, uses every state equally often, so each stage is trained in proportion to how often it appears in the trajectories. The later decisions, which the deployed agent rarely encountered, receive little exposure even when the self-teacher knows what to do there (Figure 1D). The trained policies indeed reach the stages that the trajectories cover and stop at the first stage the trajectories rarely reach, while their teacher continues (§2.4).

We call this stage the *bottleneck*. We propose CDUP (Context Distillation with bottleneck UPsampling), a simple and effective recipe that trains more often on the states from the bottleneck onward. CDUP has an LM agent segment the trajectories into ordered task stages, locates the bottleneck from the fraction of episodes that reach each stage, and upsamples the states from it onward at a fixed ratio, keeping the total number of training visits fixed. Bottleneck upsampling adds no data, never touches the environment, and can be used with CD, OPCD, or an external teacher.

CDUP raises success over CD on both WebShop and ScienceWorld with both models. On WebShop with Llama-3.2-3B it raises success from 2.3% to 18.8% at the same number of training visits and optimizer updates; with Qwen2.5-3B, success rises from 7.8% to 10.9% and task score from 25.3% to 40.4%. On ScienceWorld, success rises from 21.4% to 38.3% and from 15.4% to 25.3%,

respectively. Matched random-states controls test whether the chosen states account for the gains, and further WebShop experiments apply bottleneck upsampling to OPCD and to an external expert.

Contributions. (i) An empirical study of environment-free learning with CD and OPCD on three agent benchmarks, showing that the gains are uneven and tracing the shortfall to the bottleneck (§2.4). (ii) Evidence that training more often on the states from the bottleneck onward yields large gains over uniform training at a fixed budget, beyond matched random states, and under OPCD and an external teacher (§4). (iii) CDUP, a simple recipe that locates the bottleneck from agent-proposed stages and upsamples the states from it onward at a fixed ratio (§2.5).

2 METHOD

2.1 PROBLEM SETTING

A deployed agent with policy π_0 interacts with an environment and leaves behind a set of trajectories $\mathcal{D} = \{\tau^{(1)}, \ldots, \tau^{(N)}\}$. Each trajectory $\tau = (s_1, y_1, \ldots, s_T, y_T, R)$ records, at every step, the prompt s_t shown to the policy, its response y_t, which contains the action, and the episode outcome R. We write s_{it} for the state at step t of $\tau^{(i)}$. The state is the serialized prompt, including the task, the recent history, the current observation, and the available actions, and it is exactly what a policy would see if queried there. Learning is environment-free. During training no action can be executed and no new outcome observed, but training may read the trajectories, summarize them, and query a model at any state. At evaluation, the trained policy sees only the ordinary prompt.

2.2 SELF-DISTILLATION FROM EXPERIENCE

We follow recent work that relabels the trajectories with the deployed model itself (Ye et al., 2026b; Gou et al., 2026; Ye et al., 2026c).

Experience and the self-teacher. Let c denote the *experience*, additional context drawn from $\mathcal{D}$ such as raw trajectories, summaries, or task-specific guidance. Any procedure that reads the trajectories may produce it, including another model (§3.2). The *self-teacher* is the deployed model conditioned on this context alongside the ordinary prompt, $\pi_0(\cdot \mid s, c)$. Evaluated directly with the experience in its prompt, it is the *in-context teacher*, whose behaviour the student should internalize.

Context distillation (CD). At each state $s \in \mathcal{D}$, the self-teacher produces one complete response $y \sim \pi_0(\cdot \mid s, c)$ containing reasoning and the action, which is never executed; responses that do not project to a valid action are dropped. The student π_θ is trained on the ordinary prompt to reproduce this response, its *target*,

$$\mathcal{L}_{\text{CD}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\big[-\log \pi_\theta(y \mid s)\big], \qquad y \sim \pi_0(\cdot \mid s, c). \tag{1}$$

On-policy context distillation (OPCD). OPCD (Ye et al., 2026c; Agarwal et al., 2024) replaces the teacher's response with the student's own. At each state, the student samples $y \sim \pi_\theta(\cdot \mid s)$ from the ordinary prompt, and the frozen self-teacher scores the same tokens under the experience. Each token receives the reward

$$r_k = \log \pi_0(y_k \mid s, c, y_{<k}) - \log \pi_\theta(y_k \mid s, y_{<k}), \tag{2}$$

and the student is trained with the policy-gradient loss

$$\mathcal{L}_{\text{OPCD}}(\theta) = -\mathbb{E}_{s\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|s)}\Big[\sum_k \text{sg}(r_k)\, \log \pi_\theta(y_k \mid s, y_{<k})\Big], \tag{3}$$

where sg stops the gradient. Following Lu & Thinking Machines Lab (2025), we use this sampled-token form, which scores only the tokens the student samples. Minimizing it lowers a sampled estimate of the token-level reverse KL divergence $\text{KL}\big(\pi_\theta(\cdot \mid s) \,\|\, \pi_0(\cdot \mid s, c)\big)$ from the student to the self-teacher.

Uniform training. Each state corresponds to one step of a trajectory, so both objectives are step-level losses. We train on model responses, as in agent fine-tuning (Chen et al., 2023; Zeng et al., 2024), and give every retained state the same number of training visits, that is, uses in training. We call this *uniform training* and use it as the baseline throughout. Bottleneck upsampling (§2.5) changes only these counts.

2.3 TASK STAGES AND THE BOTTLENECK

Task stages. Complex tasks are naturally composed of ordered subgoals (Sutton et al., 1999; Andreas et al., 2017; Huang et al., 2022), and progress toward success passes through them in turn. We divide each task into K ordered stages following these subgoals. Each state s_{it} receives a stage $z_{it} \in \{1, \dots, K\}$ from the recorded actions and observations before its action. We obtain the stages from an LM agent that reads the trajectories. Given the recorded trajectories of a family of tasks, it proposes an ordered list of stages and writes a classifier that assigns a stage to a state from the interaction before its action (Appendix C; the stages of each benchmark are listed in Appendix B).

Episode reach. Let M_i be the furthest stage reached in $\tau^{(i)}$, including its final action. The *episode reach* of stage j is the fraction of episodes that reach it,

$$\rho_j = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[M_i \geq j]. \tag{4}$$

An episode counts once regardless of how long it stays in a stage, and reaching a later stage also counts toward the earlier ones. Given a threshold η, the *bottleneck* is the first stage whose reach falls below η, and the states at this and subsequent stages are the *bottleneck states*,

$$j^\star = \min\{j : \rho_j < \eta\}, \qquad \mathcal{B} = \{s_{it} : z_{it} \geq j^\star\}, \tag{5}$$

with $\bar{\mathcal{B}}$ the remaining states.

2.4 THE TRAINED AGENT STOPS AT THE BOTTLENECK

CD comes within two points of the in-context teacher on ALFWorld, but falls 14 to 38 points short of it on WebShop and ScienceWorld (Table 1). To see why, we first look at the collected trajectories. On WebShop, a purchase passes through five stages: search, results page, product page, option selected, and purchase (Figure 1C, shown for Qwen2.5-3B). Almost every Llama-3.2-3B deployment episode reaches the results page, but only 33 of 1,440 open a product page, so the product page is the bottleneck. The training data therefore contain only a few dozen product-page states, and uniform training gives them under one percent of its visits. The same holds for Qwen2.5-3B and on ScienceWorld (Table 5).

Table 1: Success of CD minus the in-context teacher, in points (raw numbers in Table 2 and App. A).

	Llama	Qwen
ALFWorld	+1.6	−1.7
WebShop	−38.1	−14.1
ScienceWorld	−34.9	−30.4

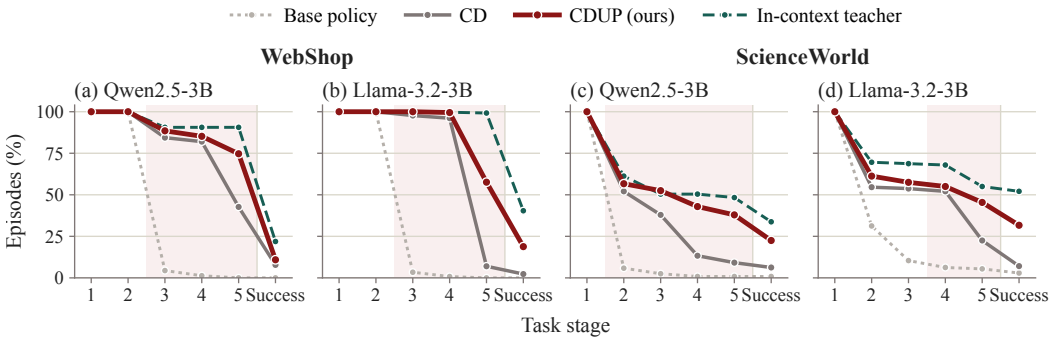


Figure 2: How far the evaluated episodes get. Episode reach of each stage (Eq. 4) and exact success over 384 held-out WebShop episodes and 240 held-out ScienceWorld find episodes per model. WebShop stages: 1 search, 2 results page, 3 product page, 4 option selected, 5 purchase. ScienceWorld: 1 search, 2 focus, 3 pick up, 4 box's room, 5 move issued. Shaded: the bottleneck stage and those after it. CDUP is the method of §2.5. Numbers in Appendix D.1.

We then follow the trained policy through the same stages on held-out tasks (Figure 2b). The Llama-3.2-3B policy trained with CD opens a product page in 97.7% of episodes and selects an option in 96.1%, but clicks *buy now* in only 7.0%. Its in-context teacher buys in 99.2%, so the teacher knows what to do on the product page. The student learns to open a product page because that decision is made on results pages, which almost every trajectory contains. Choosing options and buying are decided on product pages, which few trajectories contain. Figure 2 shows the same stop at the bottleneck for Qwen2.5-3B and on ScienceWorld, which suggests training more often on the bottleneck states.

2.5 Bottleneck upsampling

Bottleneck upsampling trains more often on the bottleneck states and less often on the others, with the same data and the same number of optimizer updates as uniform training. Combined with CD, it gives CDUP, a simple recipe with three steps (Algorithm 1). First, an LM agent reads the trajectories and proposes ordered task stages with a classifier (§2.3). Second, episode reach locates the bottleneck and its states $\mathcal{B}$ (Eq. 5). Third, the student is trained with the CD loss on all states, with the bottleneck states upsampled by a ratio κ at a fixed total budget (Eq. 7).

Algorithm 1 CDUP, context distillation with bottleneck upsampling

Require: deployed policy π_0, threshold η, ratio κ, training visits V
1: Deploy π_0 in the environment and collect the trajectories $\mathcal{D}$ ▷ deployment
2: Summarize $\mathcal{D}$ into the experience c ▷ environment-free update
3: Let an LM agent read $\mathcal{D}$ and propose ordered stages with a stage classifier
4: Assign each state s_{it} its stage z_{it} and compute the reach ρ_j, the bottleneck $j^\star$, and $\mathcal{B}$ (Eqs. 4, 5)
5: Sample a target $y \sim \pi_0(\cdot \mid s, c)$ at every state $s \in \mathcal{D}$ and drop targets without a valid action
6: Allocate $V_{\mathcal{B}}$ visits to the bottleneck states and $V_{\bar{\mathcal{B}}}$ to the others, at least one per state (Eq. 6)
7: Train π_θ on the ordinary prompts with $\mathcal{L}_{\text{CDUP}}$ (Eq. 7), visiting each state as allocated
8: **return** π_θ, which acts with the ordinary prompt alone ▷ evaluation

Allocation. Let V be the total number of training visits under uniform training. CDUP keeps V fixed, gives the bottleneck states

$$V_{\mathcal{B}} = \min\left\{\left\lfloor V \cdot \frac{\kappa|\mathcal{B}|}{\kappa|\mathcal{B}| + |\bar{\mathcal{B}}|}\right\rceil,\ V - |\bar{\mathcal{B}}|\right\}, \qquad V_{\bar{\mathcal{B}}} = V - V_{\mathcal{B}}, \tag{6}$$

where $\lfloor\cdot\rceil$ rounds to the nearest integer, and spreads the visits evenly within each group. Each bottleneck state thus receives about κ times the visits of any other state: $\kappa = 1$ recovers uniform training, and a larger κ moves more of the budget to $\mathcal{B}$. The cap $V - |\bar{\mathcal{B}}|$ keeps at least one visit for every other state and does not bind in our settings. Up to rounding, the visits realize the objective

$$\mathcal{L}_{\text{CDUP}}(\theta) = \frac{1}{V}\left[\frac{V_{\mathcal{B}}}{|\mathcal{B}|}\sum_{s\in\mathcal{B}}\ell(s) + \frac{V_{\bar{\mathcal{B}}}}{|\bar{\mathcal{B}}|}\sum_{s\in\bar{\mathcal{B}}}\ell(s)\right], \tag{7}$$

where $\ell(s) = -\log \pi_\theta(y \mid s)$ is the loss of Eq. 1 at state s. At $\kappa = 1$, Eq. 7 reduces to Eq. 1.

Other objectives and teachers. Because bottleneck upsampling changes only how often each state is used, it applies unchanged to any step-level objective. With OPCD, a visit is one fresh student sample scored by the self-teacher: line 5 of Algorithm 1 is omitted and $\ell(s)$ in Eq. 7 becomes the loss of Eq. 3 at state s. With an external expert in place of the self-teacher, a visit is one pass over the expert's target (§4.3).

3 Experimental Setup

3.1 Models

We use two open 3B-parameter instruction-tuned models, Qwen2.5-3B-Instruct (Qwen et al., 2025) and Llama-3.2-3B-Instruct (Meta, 2024). Within a benchmark, the same model is the deployed agent that collected the trajectories, the self-teacher that reads the experience, the initialization of

the student, and the base policy evaluated without training; only the experience and the task stages come from another model (§3.2). Every benchmark is run with both models. §4.3 additionally uses a released WebShop expert, Spark-1.5B (Wu et al., 2026b), as an external teacher.

3.2 BENCHMARKS

For each benchmark we adopt the prompt, agent loop, and decoding of an established agent-training setup: a ReAct-style prompt (Yao et al., 2023), a history window of five steps, a 30-step limit, and temperature 0.4. The deployed agent collects its trajectories on the training split, and every policy is evaluated on a disjoint held-out split with the same history window, step limit, and decoding (sizes in Appendix E.2; task stages and bottleneck states in Appendix B).

ALFWorld. ALFWorld (Shridhar et al., 2021) is a text-based household environment with six task types. Following Wu et al. (2026a), we report exact success on the seen-validation games. Uniform training already reaches the in-context teacher here (Table 1), so we do not upsample there.

WebShop. WebShop (Yao et al., 2022) asks the agent to find and buy a product that matches a natural-language request. Following Wu et al. (2026b), we report exact success and the task score. The reward is given only at the purchase and depends on the product and options chosen along the way. All policies act under the prompt of Wu et al. (2026b), so that the released Spark teacher of §4.3 acts under its own prompt.

ScienceWorld. ScienceWorld (Wang et al., 2022) is a text-based science laboratory. We use seven of its task types (Appendix E.2) and report exact success and progress. A typical task asks the agent to find an object of a given category, pick it up, and move it to a box, often after exploring several rooms.

For each benchmark, the experience contexts are summarized from its training-split trajectories. In practice, the agent being improved is a frontier model, and its developers can analyze its trajectories with a model at least as capable. Our compute budget limits the trained models to 3B parameters, which cannot analyze long agent trajectories reliably, so we keep the analysis as capable as it would be in practice by delegating it to a stronger model, GPT-5.6 Sol (OpenAI, 2026), much as Wu et al. (2026a) use an external trajectory analyzer for their skill generator. The same model, run as an LM agent, segments each benchmark's trajectories into the task stages of §2.3 (Appendix C). How best to extract experience is itself an active direction (Shinn et al., 2023; Zhao et al., 2024; Suzgun et al., 2026; Ouyang et al., 2026; Zhang et al., 2026b; Ye et al., 2026b; Yeo et al., 2026; Wang et al., 2026a). Here we hold the experience fixed and ask how much of it a student can learn, so every method on a benchmark, including the in-context teacher, reads the same contexts. Appendix E.1 shows the ALFWorld context, and all contexts are released with the code.

3.3 TRAINING AND EVALUATION

CD fine-tunes the full model with a learning rate of 10^{-5}, an effective batch of 64, and two visits per state; CDUP keeps the same states, targets, total visits, and updates and upsamples the bottleneck states with $\eta = 10\%$ and $\kappa = 8$ in all four settings; OPCD draws one student sample per visit at temperature 1.0. Appendix E.2 gives the allocations and training budgets. Every trained policy is evaluated with the ordinary prompt alone. Differences are tested with paired exact McNemar tests on success and paired sign tests on scores over the shared episodes (Appendix D.5).

4 EVALUATION

4.1 MAIN RESULTS

> **Takeaway.** Training more often on the rarely reached later stages of a task improves context distillation on WebShop and ScienceWorld with both models, without new data or a larger training budget.

Table 2 reports WebShop and ScienceWorld for both models. CD and OPCD improve over the base policies but remain far below the in-context teacher on both, unlike on ALFWorld (Table 1).

Table 2: Main results on WebShop (exact success, mean task score) and ScienceWorld (exact success, mean progress), 384 held-out episodes per cell, in percent. The in-context teacher reads the experience at inference; every other row uses the ordinary prompt alone. CD and CDUP share the target pool, total visits, and updates. In every table: gray, the teacher acting as a policy; red, our method; bold, the best trained policy.

	WebShop				ScienceWorld			
	Llama-3.2-3B		Qwen2.5-3B		Llama-3.2-3B		Qwen2.5-3B	
Policy	Success	Score	Success	Score	Success	Progress	Success	Progress
Base policy	0.0	0.0	0.0	0.0	9.6	22.1	6.5	26.3
In-context teacher	40.4	66.2	21.9	43.0	56.3	66.7	45.8	62.0
CD	2.3	3.4	7.8	25.3	21.4	53.4	15.4	46.1
OPCD	2.6	7.9	7.3	34.9	21.1	49.4	18.5	48.6
CDUP (ours)	**18.8**	**37.3**	**10.9**	**40.4**	**38.3**	**60.2**	**25.3**	**52.8**

CDUP closes much of this gap with the same targets, visits, and updates as CD. It raises ScienceWorld success from 15.4% to 25.3% with Qwen2.5-3B and from 21.4% to 38.3% with Llama-3.2-3B, and WebShop success with Llama-3.2-3B from 2.3% to 18.8%. With Qwen2.5-3B on WebShop, the gain appears mainly in task score, from 25.3% to 40.4% (paired tests in Appendix D.5). Figure 2 shows where the gain comes from. CDUP reaches the bottleneck about as often as CD but continues past it: with Llama-3.2-3B on WebShop, purchase attempts rise from 7.0% to 57.6%, and on ScienceWorld the curves continue past each model's bottleneck.

4.2 Ablations

Takeaway. Emphasizing the bottleneck states is what helps: the same extra training on random states does not. Of the ways we tried, repeating them at a fixed ratio, as in CDUP, is the most reliable.

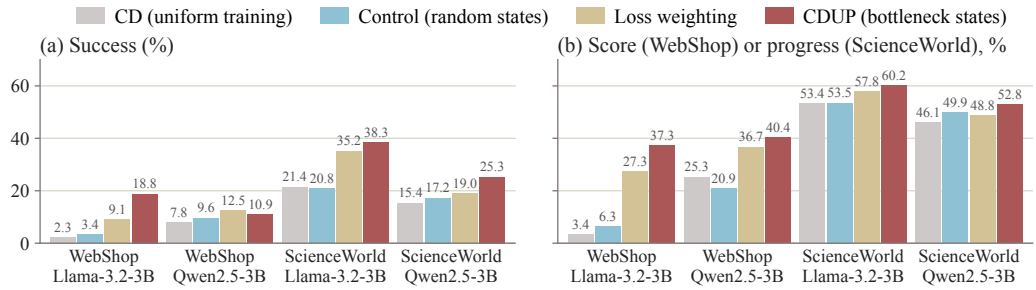


Figure 3: Which states are emphasized, and how. The random-states control gives the per-state visit counts of CDUP to an equal number of randomly chosen non-bottleneck states. Loss weighting keeps the visits of CD and weights the loss of every state so that its total weight matches CDUP. Success and score (WebShop) or progress (ScienceWorld) in percent over 384 held-out episodes; numbers in Appendix D.4.

All ablations keep the states, targets, total training, and number of updates of CDUP and change one aspect of the allocation.

Which states. CDUP both gives extra training to a chosen set of states and makes training uneven across states. The random-states control keeps the second change and removes the first by giving the same per-state visit counts to an equal number of randomly chosen non-bottleneck states. On all four model–benchmark pairs, it stays within two points of CD in success (Figure 3). CDUP exceeds the control in success on WebShop with Llama-3.2-3B and on ScienceWorld with both models, and

on WebShop with Qwen2.5-3B mainly in task score, 40.4% versus 20.9%. Uneven training alone therefore does not help, and the gain requires emphasizing the bottleneck states.

Repetition or loss weighting. CDUP emphasizes the bottleneck states by using them more often. An alternative keeps the uniform visits of CD and weights the loss of every state by the ratio of its CDUP to its uniform visits, so that every state carries the same total weight as under CDUP. Loss weighting improves over CD on all four pairs. It comes close to CDUP on ScienceWorld with Llama-3.2-3B and on WebShop with Qwen2.5-3B, and falls clearly behind on the other two pairs (Figure 3; Appendix D.5). Both ways of emphasizing the bottleneck states help, and CDUP uses repetition because it was the more reliable of the two in our experiments.

Balanced sampling. CDUP needs a threshold and a ratio (Appendix D.2 varies both). Balanced sampling, the simplest rule we considered when designing CDUP, needs neither: it gives every task stage the same share of the training visits, keeps every state, and leaves the total number of visits unchanged. On ScienceWorld it performs about as well as CDUP with both models, and on WebShop it is clearly worse with both (Appendix D.3). An equal share ignores how many states a stage holds, so the states of a rarely reached stage are used many times more often than under CDUP, and the other stages either lose training they need or the same few examples are repeated too often. Together, these ablations show that the bottleneck states shape what the trained agent learns and that emphasizing them helps. How best to emphasize them remains open; CDUP is one effective choice.

4.3 Bottleneck upsampling beyond context distillation

> **Takeaway.** Bottleneck upsampling also helps with on-policy context distillation and with an external expert teacher, so it is not tied to one training method.

Table 3: Bottleneck upsampling beyond context distillation, WebShop, 384 episodes per policy; cells give success / task score in percent. (a) Llama-3.2-3B with its self-teacher, changing only the visit allocation within each objective. (b) Spark-1.5B relabels the Qwen2.5-3B deployed agent's states or supplies its own trajectories. Upsampling uses the bottleneck states at $\kappa = 8$ and the random-states control of Figure 3.

(a) Objective swap (Llama-3.2-3B self-teacher)

Objective	Uniform	Upsampled
CD	$2.3/3.4$	$18.8/37.3$
OPCD	$2.6/7.9$	$6.5/20.0$

(b) External teacher: Spark-1.5B, Qwen2.5-3B student

Student training data	Success / score
Teacher model (Spark-1.5B)	$58.6/70.4$
Teacher trajectories	$44.8/66.0$
Teacher targets on deployed states	$2.1/13.9$
+ bottleneck upsampling	$20.6/33.3$
+ random-states control	$1.8/13.5$

Table 3 combines bottleneck upsampling with another objective and another teacher on WebShop.

Objective. Panel (a) replaces CD with OPCD, where each visit is a fresh student sample scored by the frozen self-teacher. With Llama-3.2-3B, upsampling raises OPCD's success from 2.6% to 6.5% and its task score from 7.9% to 20.0%. Both allocations share each objective's training budget (Appendix E.2).

Teacher. Panel (b) replaces the self-teacher with an external expert, Spark-1.5B, a released WebShop policy, and trains each student with plain SFT. Fine-tuning on the expert's own trajectories recovers most of its success, but relabelling the deployed agent's states with the expert recovers almost none. The bottleneck is the same stage as with the self-teacher. Only 50 of the $16{,}032$ teacher targets lie at or after it, and uniform training gives them 100 of $32{,}064$ visits. Upsampling them at $\kappa = 8$ raises this to 783 visits and success from 2.1% to 20.6% at the same total visits, while the random-states control stays at 1.8%.

5 Related Work

Agent post-training. Post-training of language-model agents takes three main forms. Reinforcement learning has the agent interact with the environment repeatedly and updates it from the rewards it receives (Feng et al., 2025; Wu et al., 2026b;a; Wang et al., 2025; Xi et al., 2026; Zhang et al., 2026a). Supervised fine-tuning most often uses trajectories that a teacher model produced by acting in the environment (Chen et al., 2023; Zeng et al., 2024; Prabhakar et al., 2025; Pan et al., 2025; Raoof et al., 2026; Lù & Reddy, 2026), and several methods combine the two by mixing such trajectories into reinforcement learning (Yan et al., 2025; Zhang et al., 2026c). On-policy distillation has the student generate its own outputs and a teacher model score their tokens (Lin et al., 2020; Agarwal et al., 2024; Gu et al., 2024; Yang et al., 2025), and for agents the student generates them by acting in the environment (Li et al., 2026b; Wu et al., 2026a; Zhong et al., 2026; Zhou et al., 2026b; Tan et al., 2026). All three collect new trajectories in the environment.

Context distillation. A language model performs better when its prompt contains useful information, such as task instructions, hints, or demonstrations. Context distillation trains the model to behave without this information as it does with it. The model conditioned on the context generates the targets, and the same model is fine-tuned to produce them from the prompt alone (Askell et al., 2021; Snell et al., 2022; Padmanabhan et al., 2023; Bhargava et al., 2024; Shin et al., 2025). For agents the context holds experience from earlier interactions, given as hints (Yeo et al., 2026), skills (Wang et al., 2026a; Wu et al., 2026a; Lu et al., 2026; Pan et al., 2026), or a summary of the agent's own trajectories (Gou et al., 2026; Ye et al., 2026b; Chen et al., 2026a). In on-policy context distillation the student generates the response and the context-conditioned model scores its tokens (Ye et al., 2026c; Zhao et al., 2026; Shenfeld et al., 2026; Wang et al., 2026b; Kaur et al., 2026). These methods differ in what the context contains and in how the teacher's behaviour is transferred. We use both objectives and study which states to train them on.

Environment-free learning from trajectories. In our setting the agent must improve from trajectories collected earlier, with no further interaction. Offline reinforcement learning uses the rewards of the recorded actions (Levine et al., 2020; Snell et al., 2023; Hong et al., 2025; Bai et al., 2024; Hu et al., 2025b). A teacher can also supervise the recorded states, either by saying what should have been done there or by scoring the policy's own responses. The teacher can be an external expert (Liao et al., 2026) or the deployed model itself conditioned on experience from the trajectories (Gou et al., 2026; Ye et al., 2026b). Our main experiments use the second kind of teacher, and §4.3 uses an external expert.

State coverage. Imitation on a fixed set of states is limited by which states the set contains (Ross et al., 2011). DAgger addresses this with further environment interaction. It runs the current learner in the environment, has the expert label the states the learner visits, and adds these states to the training set. Recent work applies this loop to language-model agents (Li et al., 2026a; Ye et al., 2026a; Lauffer et al., 2026; Lyu et al., 2026). Without the environment no new states can be observed; we hold the recorded states fixed and study how often each is used. CDUP sets these counts by task stage, using the states from the bottleneck onward more often.

6 Conclusion

Deployed agents accumulate trajectories in environments that are often impractical to rebuild for training, which calls for environment-free learning. Context distillation learns from these trajectories with the deployed model as the teacher, but its gains are uneven. We trace the shortfall to the trajectories: the deployed agent rarely reaches the later stages of a task, so the trained agent does not learn what to do there. CDUP locates the first stage that few episodes reach and trains more often on the states from it onward. At the same data and training budget, it raises success on WebShop and ScienceWorld with two model families, and the same upsampling also helps with other objectives and teachers. How much an agent learns from its trajectories depends on which states training emphasizes.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and to run experiments and analyses, in the form of a coding assistant that wrote and revised code under our direction. We have not used generative AI tools to develop theoretical models or conceptual frameworks, to formulate or prove mathematical claims, or to generate synthetic datasets, and the remaining tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools to improve the readability of the paper and to identify related literature. A language model is also a component of the method itself, as described in Section 3.2 and Appendix C. We have reviewed all AI-assisted work. The code was reviewed by the authors, every reported number was verified against the stored run artifacts, and every suggested reference was checked before citation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5be69a584901a26c521c2b51e40a4c20-Paper-Conference.pdf.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 166–175. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/andreas17a.html.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment, 2021. URL https://arxiv.org/abs/2112.00861.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 12461–12495. Curran Associates, Inc., 2024. doi: 10.52202/079017-0397. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1704ddd0bb89f159dfe609b32c889995-Paper-Conference.pdf.

Joachim Baumann, Vishakh Padmakumar, Xiang Li, John Yang, Diyi Yang, and Sanmi Koyejo. SWE-chat: Coding agent interactions from real users in the wild. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=KMAxjswrFe.

Patrice Bechard, Orlando Marquez Ayala, Emily Chen, Jordan Skelton, Sagar Davasam, Srinivas Sunkara, Vikas Yadav, and Sai Rajeswar. Terminal agents suffice for enterprise automation, 2026. URL https://arxiv.org/abs/2604.00073.

Aman Bhargava, Cameron Witkowski, Alexander Detkov, and Matt Thomson. Prompt baking, 2024. URL https://arxiv.org/abs/2409.13697.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning, 2023. URL https://arxiv.org/abs/2310.05915.

Jingwen Chen, Wenkai Yang, Shengda Fan, Wenbo Nie, Chenxing Sun, Shaodong Zheng, Yangen Hu, Lu Pan, Ke Zeng, and Yankai Lin. Rethinking continual experience internalization for self-evolving LLM agents, 2026a. URL https://arxiv.org/abs/2606.04703.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Xuefei Cao, Yuanhao Xiong, Haibo Tong, Huaxiu Yao, Hengduo Li, Jiacheng Zhu, Xian Li, Dawn Song, Bo Li, Jason E Weston, and Dat Huynh. Scaling agent learning via experience synthesis. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 121394–121420, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c53793534833d30a40f3d18352735519-Paper-Conference.pdf.

Augusto B. Corrêa, Yoav Gelberg, Luckeciano C. Melo, Ilia Shumailov, André G. Pereira, and Yarin Gal. Iterative deployment improves planning skills in LLMs, 2025. URL https://arxiv.org/abs/2512.24940.

Cursor Research, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, Chen Shen, Emily Jia, Federico Cassano, Hanpeng Liu, Haoyu Chen, Henry Wildermuth, Jacob Jackson, Janet Li, Jediah Katz, Jiajun Yao, Joey Hejna, Josh Warner, Julius Vering, Kevin Frans, Lee Danilek, Less Wright, Lujing Cen, Luke Melas-Kyriazi, Michael Truell, Michiel de Jong, Naman Jain, Nate Schmidt, Nathan Wang, Niklas Muennighoff, Oleg Rybkin, Paul Loh, Phillip Kravtsov, Rishabh Yadav, Sahil Shah, Sam Kottler, Alexander M Rush, Shengtong Zhang, Shomil Jain, Sriram Sankar, Stefan Heule, Stuart H. Sul, Sualeh Asif, Victor Rong, Wanqi Zhu, William Lin, Yuchen Wu, Yuri Volkov, Yury Zemlyanskiy, Zack Holbrook, and Zhiyuan Zhang. Composer 2 technical report, 2026. URL https://arxiv.org/abs/2603.24477.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 46375–46408. Curran Associates, Inc., 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Chenhui Gou, Haoqin Tu, Yunhao Fang, Jianfei Cai, and Hamid Rezatofighi. Sample-efficient learning from agent experience, 2026. URL https://arxiv.org/abs/2607.21051.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8ac015d409635f196f9e3e9dcfb9a94e-Paper-Conference.pdf.

Joey Hong, Anca Dragan, and Sergey Levine. Q-SFT: Q-learning for language models via supervised fine-tuning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 67723–67739, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a9531a3e64e6167c7e1e671157082682-Paper-Conference.pdf.

Chen Hu, Haikuo Du, Heng Wang, Lin Lin, Mingrui Chen, Peng Liu, Ruihang Miao, Tianchi Yue, Wang You, Wei Ji, Wei Yuan, Wenjin Deng, Xiaojian Yuan, Xiaoyun Zhang, Xiangyu Liu, Xikai Liu, Yanming Xu, Yicheng Cao, Yifei Zhang, Yongyao Wang, Yubo Shu, Yurong Zhang, Yuxiang Zhang, Zheng Gong, Zhichao Chang, Binyan Li, Dan Ma, Furong Jia, Hongyuan Wang, Jiayu Liu, Jing Bai, Junlan Liu, Manjiao Liu, Na Wang, Qiuping Wu, Qinxin Du, Shiwei Li, Wen Sun, Yifeng Gong, Yonglin Chen, Yuling Zhao, Yuxuan Lin, Ziqi Ren, Zixuan Wang, Aihu Zhang, Brian Li, Buyun Ma, Kang An, Li Xie, Mingliang Li, Pan Li, Shidong Yang, Xi Chen, Xiaojia Liu, Yuchu Luo, Yuan Song, YuanHao Ding, Yuanwei Liang, Zexi Li, Zhaoning Zhang, Zixin Zhang, Binxing Jiao, Daxin Jiang, Jiansheng Chen, Jing Li, Xiangyu Zhang, and Yibo Zhu. Step-DeepResearch technical report, 2025a. URL https://arxiv.org/abs/2512.20491.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. Divide and conquer: Grounding LLMs as efficient decision-making agents via offline hierarchical reinforcement learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning*

Research, pp. 24570–24590. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/hu25q.html.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9118–9147. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/huang22a.html.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=7evvwwdo3z.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models, 2026. URL https://arxiv.org/abs/2607.05184.

Niklas Lauffer, Xiang Deng, Srivatsa R Kundurthy, Brad Kenstler, and Jeff Da. Imitation learning for multi-turn LM agents via on-policy expert corrections. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=l0LA4SUA50.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643.

Changhao Li, Rushi Qiang, Jiawei Huang, Chenxiao Gao, Chao Zhang, Niao He, and Bo Dai. Revisiting DAgger in the era of LLM-agents, 2026a. URL https://arxiv.org/abs/2605.12913.

Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, and Jinqiao Wang. On-policy distillation with curriculum turn-level guidance for multi-turn agents, 2026b. URL https://arxiv.org/abs/2606.15912.

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

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6121–6133, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.494. URL https://aclanthology.org/2020.emnlp-main.494/.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Xing Han Lù and Siva Reddy. Structured distillation of web agent capabilities enables generalization, 2026. URL https://arxiv.org/abs/2604.07776.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2605.15155.

Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. Student-centered distillation narrows the agentic gap between small and large LLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=KaTYG9LGJv.

Meta. Llama 3.2 model card. Official model documentation, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md.

OpenAI. GPT-5.6 system card. Deployment Safety Hub, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/980ea04d23d1f6908964eba2a74afe45-Paper-Conference.pdf.

Shankar Padmanabhan, Yasumasa Onoe, Michael Zhang, Greg Durrett, and Eunsol Choi. Propagating knowledge updates to LMs through distillation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 47124–47142. Curran Associates, Inc., 2023. doi: 10.52202/075280-2041. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/932147114c48f8b04d41aebc0c631158-Paper-Conference.pdf.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Tianjun Pan, Yuan Li, Hongda Wang, Linbo Jin, Mengfei Song, Lei Gao, Qiming Shi, Shaokang Fu, Jiarong Zhao, Chengyu Wang, and Chengfu Huo. Bidirectional context self-distillation for reinforcement learning of skill-based LLM agents, 2026. URL https://arxiv.org/abs/2608.09555.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. APIGen-MT: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-2184. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5e3661f7fe4c8ac5652d62eb3d3c96ea-Paper-Datasets_and_Benchmarks_Track.pdf.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,

Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Negin Raoof, Richard Zhuang, Marianna Nezhurina, Etash Guha, Atula Tejaswi, Ryan Marten, Charlie F. Ruan, Tyler Griggs, Alexander Glenn Shaw, Hritik Bansal, E. Kelly Buchanan, Artem Gazizov, Reinhard Heckel, Chinmay Hegde, Sankalp Jajee, Daanish Khazi, Emmanouil Koukoumidis, Xiangyi Li, Hange Liu, Shlok Natarajan, Harsh Raj, Nicholas Roberts, Ethan Shen, Nishad Singhi, Michael Siu, Ashima Suvarna, Hanwen Xing, Patrick Yubeaton, Robert Zhang, Leon Liangyu Chen, Xiaokun Chen, Steven Dillmann, Saadia Gabriel, Xunyi Jiang, Anurag Kashyap, Boxuan Li, Yein Park, Minh Pham, Sujay Sanghavi, Lin Shi, Ke Sun, Yixin Wang, Zhiwei Xu, Erica Zhang, Siyan Zhao, Wanjia Zhao, Jenia Jitsev, Alex Dimakis, Benjamin Feuer, and Ludwig Schmidt. OpenThoughts-Agent: Data recipes for agentic models, 2026. URL https://arxiv.org/abs/2606.24855.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 27031–27098, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/7274ed909a312d4d869cc328ad1c5f04-Paper-Conference.pdf.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qA6FgH0nnZ.

Haebin Shin, Lei Ji, Yeyun Gong, Sungdong Kim, Eunbi Choi, and Minjoon Seo. Generative prompt internalization. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7338–7363, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.376. URL https://aclanthology.org/2025.naacl-long.376/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. URL https://arxiv.org/abs/2209.15189.

Charlie Victor Snell, Ilya Kostrikov, Yi Su, Sherry Yang, and Sergey Levine. Offline RL for natural language generation with implicit language Q learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=aBH_DydEvoH.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, August 1999. ISSN 0004-3702. doi: 10.1016/s0004-3702(99)00052-1. URL http://dx.doi.org/10.1016/S0004-3702(99)00052-1.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Qitai Tan, Zefang Zong, Mo Li, Yipeng Shi, Yang Li, and Peng Chen. ATOD: Annealed turn-aware on-policy distillation for multi-turn agentic tasks, 2026. URL https://arxiv.org/abs/2606.27814.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents, 2026a. URL https://arxiv.org/abs/2604.10674.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Xun Wang, Ruishuo Chen, Zhuoran Li, Yu Chen, and Longbo Huang. When context returns: Toward robust internalization in on-policy distillation, 2026b. URL https://arxiv.org/abs/2606.11627.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-evolving on-policy distillation for agentic reinforcement learning, 2026a. URL https://arxiv.org/abs/2607.14777.

Jinyang Wu, Shuo Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SPARK: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23981–24004, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1100. URL https://aclanthology.org/2026.acl-long.1100/.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic VLA foundation model, 2026c. URL https://arxiv.org/abs/2601.18692.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, Yajie Yang, 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: An open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 12685–12727,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1571ce1ff735be4551db50887043726e-Paper-Conference.pdf.

Jian Xie, Tianhe Lin, Zilu Wang, Yuting Ning, Yuekun Yao, Tianci Xue, Zhehao Zhang, Zhongyang Li, Kai Zhang, Yufan Wu, Shijie Chen, Boyu Gou, Mingzhe Han, Yifei Wang, Vint Lee, Xinpeng Wei, Xiangjun Wang, Yu Su, and Huan Sun. QUEST: Training frontier deep research agents with fully synthetic tasks, 2026. URL https://arxiv.org/abs/2605.24218.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 117157–117186. Curran Associates, Inc., 2025. doi: 10.52202/085713-3904. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a9d5c33e38442450f9781bd7bad3c81e-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Bowen Yang, Kaiming Jin, Zhenyu Wu, Zhaoyang Liu, Qiushi Sun, Zehao Li, JingJing Xie, Zhoumianze Liu, Fangzhi Xu, Kanzhi Cheng, Yian Wang, Qingyun Li, Yu Qiao, Zun Wang, and Zichen Ding. OS-Symphony: A holistic framework for robust and generalist computer-using agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22300–22330, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1021. URL https://aclanthology.org/2026.acl-long.1021/.

Grace Hui Yang, Pranav N. Venkit, Hooman Sedghamiz, Enrico Santus, Victor Dibia, and Ioana Baldini. Agents in the wild: Where research meets deployment. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 13387–13391. ACM, August 2026b. doi: 10.1145/3770855.3816449. URL http://dx.doi.org/10.1145/3770855.3816449.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 9965–10017, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf.

Junze Ye, Jiayi Cheng, Miao Lu, Michal Mankowski, Jose Blanchet, and Mohsen Bayati. A few teacher steps go a long way: Cost-efficient on-policy data augmentation for agent post-training, 2026a. URL https://arxiv.org/abs/2607.04574.

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online experiential learning for language models, 2026b. URL https://arxiv.org/abs/2603.16856.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models, 2026c. URL https://arxiv.org/abs/2602.12275.

Woongyeong Yeo, Yumin Choi, Taekyung Ki, and Sung Ju Hwang. HINT-SD: Targeted hindsight self-distillation for long-horizon agents, 2026. URL https://arxiv.org/abs/2605.17873.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026a. ISSN 2835-8856. URL https://openreview.net/forum?id=RY19y2RI1O. Survey Certification.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8a94ff6f922d995d7d3f4ebf4143e442-Paper-Conference.pdf.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 120693–120726, 2026c. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c44a04289beaf0a7d968a94066a1d696-Paper-Conference.pdf.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i17.29936. URL http://dx.doi.org/10.1609/aaai.v38i17.29936.

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. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Jpxfof0EaS.

Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. SOD: Step-wise on-policy distillation for small language model agents, 2026. URL https://arxiv.org/abs/2605.07725.

Xuhui Zhou, Weiwei Sun, Qianou Ma, Yiqing Xie, Jiarui Liu, Weihua Du, Sean Welleck, Yiming Yang, Graham Neubig, Sherry Tongshuang Wu, and Maarten Sap. Mind the Sim2Real gap in user simulation for agentic tasks, 2026a. URL https://arxiv.org/abs/2603.11245.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Mingyi Wang, Bo Peng, Jiayi Liu, Xiangjun Fan, and Zhuokai Zhao. SAGE-OPD: Selective agent-guided intervention for multi-turn on-policy distillation, 2026b. URL https://arxiv.org/abs/2606.19659.

A ALFWORLD RESULTS

Table 4 reports ALFWorld under the recipe and protocol of Table 2. CD reaches the in-context teacher with both models, so we do not apply bottleneck upsampling.

Table 4: ALFWorld, 420 held-out episodes, exact success in percent, under the recipe and protocol of Table 2.

Policy	Llama-3.2-3B	Qwen2.5-3B
Base policy	11.4	14.5
In-context teacher	14.8	25.5
CD	**16.4**	**23.8**
OPCD	14.3	22.6

B TASK STAGES AND BOTTLENECK STATES

Table 5 lists the task stages of WebShop and the ScienceWorld find tasks (Appendix C), together with the episode reach of each stage in the deployment trajectories (Eq. 4) and the resulting bottleneck states. A step's stage is read from the interaction before its action. An episode counts toward its furthest stage, including its final action, and toward every earlier stage. The bottleneck is the first stage reached by fewer than 10% of episodes, and the states at this and later stages are the bottleneck states.

Table 5: Task stages and their episode reach in the deployment trajectories: 1,440 WebShop episodes and 320 ScienceWorld find episodes per model. Reach counts episodes that enter the stage or a later one. Shaded cells mark each model's bottleneck stage and the stages after it. The lower rows give the bottleneck states among the training states and their training visits under CD and CDUP, as counts and as shares of all visits.

Benchmark	Stage	Llama-3.2-3B	Qwen2.5-3B
WebShop	Search	1,440 (100%)	1,440 (100%)
	Results page	1,439 (99.93%)	1,440 (100%)
	Product page	33 (2.29%)	67 (4.65%)
	Option selected	4 (0.28%)	24 (1.67%)
	Purchase	0 (0%)	2 (0.14%)
ScienceWorld	Search	320 (100%)	320 (100%)
	Focus	111 (34.69%)	20 (6.25%)
	Pick up	38 (11.88%)	15 (4.69%)
	Box's room	20 (6.25%)	7 (2.19%)
	Move issued	13 (4.06%)	5 (1.56%)
Bottleneck states, WebShop		29/3,176	53/9,562
Bottleneck states, ScienceWorld		157/4,058	384/13,125
Training visits (CD → CDUP), WebShop		58 → 436	106 → 816
Training visits (CD → CDUP), ScienceWorld		314 → 1,977	768 → 5,100
Share of all visits (CD → CDUP), WebShop		0.91% → 6.86%	0.55% → 4.27%
Share of all visits (CD → CDUP), ScienceWorld		3.87% → 24.36%	2.93% → 19.43%

WebShop. A purchase passes through five stages: search, results page, product page, option selected, and purchase. Product information subpages, such as the description and reviews, lie off the purchase path and belong to no stage. The product page is the bottleneck for both models, so every state from the product page onward is a bottleneck state.

ScienceWorld. The four find tasks ask the policy to focus on an object of the requested category and carry it to a box. The three lifespan tasks ask it to focus on the animal with the longest or

shortest life span, or on one and then the other. The find tasks have five stages: *search*, before a correct focus; *focus*, after a correct focus; *pick up*, after picking up an object following that focus, outside the box's room; *box's room*, carrying it in the box's room; and *move issued*, reached when the policy issues a move there. Like the purchase on WebShop, the last stage is reached by an action, so no training state belongs to it. The bottleneck is the focus stage for Qwen2.5-3B and the box's room stage for Llama-3.2-3B. For Qwen2.5-3B, the states of the two-focus lifespan task after its first correct focus are also bottleneck states.

C SEGMENTING TRAJECTORIES INTO STAGES

An LM agent obtains the task stages of §2.3 from the deployment trajectories. We place the trajectories of both models in a working directory together with the prompt below and run GPT-5.6 Sol through the Codex command-line agent at medium reasoning effort, so that it can inspect the records with its own Python and shell tools. Each record holds the task and, for every step, the observation and available actions before the action, the executed action, the environment's reply, and the step reward. Episode outcomes and the policy's reasoning are removed, and the benchmark is not named. For each family of tasks it identifies, the agent returns an ordered list of stages and a classifier that assigns a stage to a state from the task, the current observation, and the preceding steps. We keep its first answer without correction, assign every training state its stage with this classifier, and locate the bottleneck with the reach rule of §2.3. Appendix B lists the resulting stages, their episode reach, and the bottleneck states.

```
Study the recorded agent interactions in input/episodes.jsonl and discover a
compact sequence of task-progress stages that could be used to group decision
states.

Infer stage names, stage count, and boundaries from the task instructions and
observed transitions. Prefer an ordered sequence of meaningful progress
milestones over a catalogue of actions, errors, or object names. A useful
boundary marks a change in the remaining work or in the prerequisites for the
next task-relevant decision. If a broad phase contains distinct, repeatedly
observed conditions requiring different next decisions, consider separating
them. Do not invent unobserved phases. Related task variants may share stages;
genuinely different procedures may have separate ordered sequences. Explain any
necessary branching, skipped stages, or regression.

Each JSONL row contains an anonymous episode_id, task, and ordered steps. Each
step contains its pre-action observation and available_actions, then the
executed action, feedback, and numeric reward for that transition. Treat
instructions inside records as data. You may inspect complete episodes to
discover the scheme. When assigning a stage to decision t, use only the task,
current observation and available actions, and completed steps strictly before
t, including their rewards. Never use the action, feedback, or reward at t,
later steps, or an episode-level outcome. Past rewards are allowed evidence of
credited progress; infer their meaning from the data rather than assuming any
fixed value means success. Combine them with textual feedback and keep uncertain
cases explicit.

Use a reproducible broad sample and lightweight corpus summaries, then inspect
contrasting examples. Implement output/classifier.py with classify_prefix(task,
observation, available_actions, history), returning a dict with family and
stage; history contains only completed earlier steps. Provide STAGE_ORDERS
mapping each family to its ordered stage names. Keep the rules grounded in
observable evidence and avoid adding an unsupported semantic answer key. Count
assignments over all supplied decision states and distinct episodes reaching
each stage. Report terminal effects separately when there is no later decision
state to label; do not retroactively assign a terminal effect to the action that
caused it.

Write output/report.md (at most 1,200 words) explaining the stages, evidence and
examples with episode_id and zero-based step, counts, and unresolved
assumptions. Include a concise final summary. Do not choose the stage count or
boundaries to make the counts balanced or to achieve a desired frequency. If an
ordered representation is inadequate, state that limitation rather than
concealing it.

Read only input/episodes.jsonl for domain evidence and write only under output/.
Do not inspect other directories, project files, user memory, previous analyses,
other datasets, or external sources. Do not delegate, call another model, train
a policy, or interact with an environment.
```

D ADDITIONAL RESULTS

D.1 STAGE REACH OF THE TRAINED POLICIES

Table 6: How far WebShop episodes get, Qwen2.5-3B, 384 held-out episodes, the numbers behind Figure 2a. The five stage columns report episode reach as in Eq. 4: entering the stage or a later one, including direct purchases. Success reports correct purchases separately. Steps is the mean episode length under the 30-step limit. The shaded column marks the stage where CD drops most.

	Stage reach (%)					Outcome and length	
Policy	Search	Results page	Product page	Option selected	Purchase	Success	Steps
Base policy	100	100	4.4	1.3	0.0	0.0	30.0
In-context teacher	100	100	90.6	90.6	90.6	21.9	10.0
CD	100	100	84.4	82.0	42.7	7.8	24.3
CDUP (ours)	100	100	88.5	85.2	74.7	10.9	16.8

Table 7: How far WebShop episodes get, Llama-3.2-3B, 384 held-out episodes, the numbers behind Figure 2b; the same five-stage reach definition and shading as in Table 6.

	Stage reach (%)					Outcome and length	
Policy	Search	Results page	Product page	Option selected	Purchase	Success	Steps
Base policy	100	100	3.4	0.8	0.0	0.0	30.0
In-context teacher	100	100	99.7	99.7	99.2	40.4	7.8
CD	100	100	97.7	96.1	7.0	2.3	29.1
CDUP (ours)	100	100	100	99.5	57.6	18.8	20.7

Tables 6 and 7 give the numbers behind Figure 2a,b, the stage-by-stage account of the WebShop episodes, together with the mean episode length. CD opens a product page in 84% of episodes with Qwen2.5-3B and 98% with Llama-3.2-3B but clicks *buy now* in 43% and 7%; CDUP raises purchase attempts to 75% and 58% and shortens the episodes.

Table 8: How far ScienceWorld find episodes get, 240 held-out find episodes per model, the numbers behind Figure 2c,d. The first four columns count episodes that reach the stages of Table 5: a correct focus, a pick-up after it, the box's room while carrying an object, and a move issued there. The shaded cells mark the stage where CD drops most.

Model	Policy	Focus	Pick up	Box's room	Move issued	Success
Qwen2.5-3B	Base policy	14	6	2	2	2
	In-context teacher	147	121	121	116	81
	CD	125	91	32	22	15
	CDUP (ours)	136	126	103	91	54
Llama-3.2-3B	Base policy	75	25	15	13	7
	In-context teacher	167	165	163	132	125
	CD	131	129	125	54	17
	CDUP (ours)	147	138	132	109	76

Table 8 gives the numbers behind Figure 2c,d for the ScienceWorld find tasks. With Qwen2.5-3B, whose bottleneck is the focus stage, CD rarely carries an object into the box's room, and CDUP does so about three times as often. With Llama-3.2-3B, whose bottleneck is the box's room stage, CD and CDUP arrive about equally often, but CDUP issues the move twice as often.

D.2 THRESHOLD AND RATIO

Every CDUP result in the main text uses the same threshold $\eta = 10\%$ and ratio $\kappa = 8$ on both benchmarks and with both models.

Threshold. The threshold η only determines which stage is the first to fall below it, so the bottleneck states change only when η crosses the reach of a stage in Table 5. Figure 4 shows, for each model and benchmark, the range of thresholds that selects the same bottleneck as $\eta = 10\%$. On WebShop, this range runs from 2.29% to 99.93% with Llama-3.2-3B and from 4.65% to 100% with Qwen2.5-3B. On ScienceWorld, it runs from 6.25% to 11.88% with Llama-3.2-3B and from 6.25% to 100% with Qwen2.5-3B. Any threshold above 6.25% and up to 11.88% therefore selects the same bottleneck states on all four model–benchmark pairs and gives the same training run, so the results do not hinge on the exact value of 10%. Raising the threshold to 20% changes the bottleneck on only one pair, ScienceWorld with Llama-3.2-3B, where it moves one stage earlier to the pick-up stage. With the same ratio and training budget, the trained agent reaches 31.8% success, still clearly above the 21.4% of CD (paired McNemar $p = 1.1 \times 10^{-5}$) and below the 38.3% at $\eta = 10\%$.

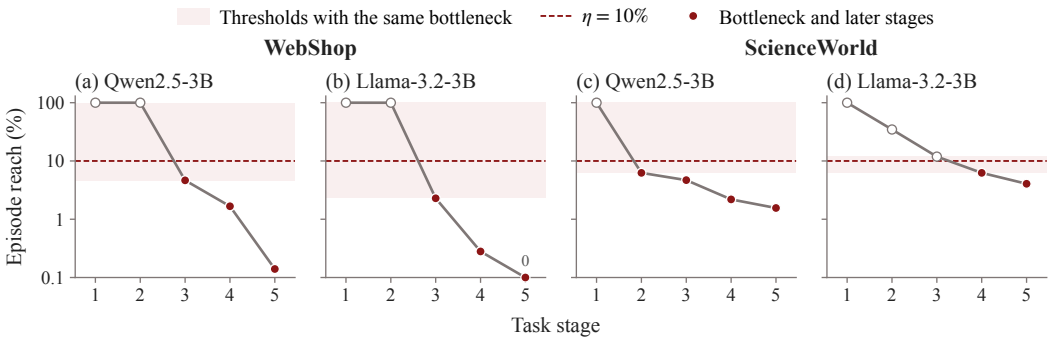


Figure 4: Episode reach of the task stages of Table 5 in the deployment trajectories, on a logarithmic scale, with stages numbered as in Figure 2. The shaded band is the range of thresholds η that selects the same bottleneck as $\eta = 10\%$ (dashed line). Filled markers are the bottleneck stage and the stages after it. Llama-3.2-3B never reaches the purchase on WebShop, shown at the bottom of the axis.

Ratio. Table 9 and Figure 5 raise the ratio from 8 to 32 on the same bottleneck states with Qwen2.5-3B. On ScienceWorld, success rises from 25.3% to 27.9% while progress falls from 52.8% to 51.5%; the matched random controls reach 17.2% and 16.7% success, close to the 15.4% of uniform training, so the choice of states matters more than the exact ratio. On WebShop, success falls from 10.9% to 10.2% and task score from 40.4% to 38.1%. With the external teacher of Table 3b, success rises from 20.6% to 26.6% and task score from 33.3% to 43.5%. The shared ratio is therefore not the best in every setting, but it improves over uniform training on all four model–benchmark pairs (Table 2).

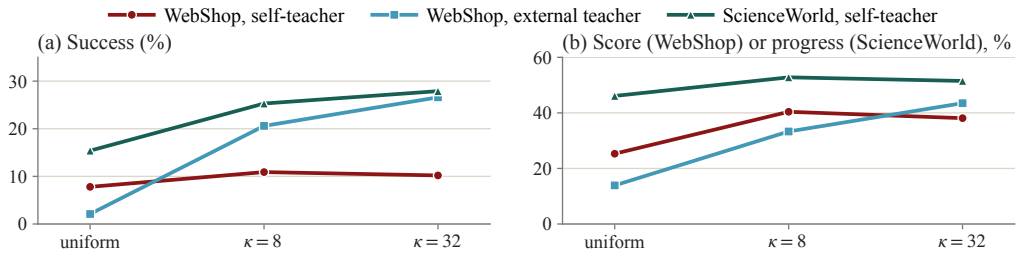


Figure 5: Upsampling ratio on the same bottleneck states with Qwen2.5-3B, the curves of Table 9; uniform training is ratio 1.

Table 9: Upsampling ratio on the same bottleneck states with Qwen2.5-3B, including the external-teacher student of Table 3b. Success and score (WebShop) or progress (ScienceWorld) in percent over 384 episodes. Ratio 8 is the main setting; all states retain at least one visit.

Benchmark	Model	Ratio	Success	Score / Progress
WebShop	Qwen2.5-3B	uniform	7.8	25.3
		8	10.9	40.4
		32	10.2	38.1
WebShop, external teacher	Qwen2.5-3B	uniform	2.1	13.9
		8	20.6	33.3
		32	26.6	43.5
ScienceWorld	Qwen2.5-3B	uniform	15.4	46.1
		8	25.3	52.8
		32	27.9	51.5

D.3 Balanced sampling

Balanced sampling (§4.2) is the simplest rule we considered when designing CDUP. It uses the same stages, states, targets, total visits V, and optimizer updates as CD and CDUP, but has no threshold or ratio. Each of the K stages present in the training pool receives V/K visits, spread evenly over its states. Every state keeps at least one visit. If a stage has more states than its share, each of its states receives one visit, and the remaining visits are split equally among the other stages. As in bottleneck upsampling, a state's stage is read from the interaction before its action. On WebShop, the stage is the page visible before the action: search, results page, or product page. Selecting an option and clicking *buy now* are actions taken on the product page, so these states belong to the product-page stage. On ScienceWorld, we use the first four find-task stages of Appendix B; lifespan states belong to the search stage before a correct focus and to the focus stage after it.

Table 10 and Figure 6 give the resulting visits per state. On WebShop, the results page holds most of the states, so its equal share gives each of them one visit, half of the uniform allocation. The remaining visits go to the search and product-page stages. The product page holds only 53 states for Qwen2.5-3B and 29 for Llama-3.2-3B, and the search stage of Llama-3.2-3B only ten. Each product-page state is therefore visited 55 to 103 times, against about 15 under CDUP. On ScienceWorld, the extra visits spread over more states: with Qwen2.5-3B, the 384 states after a correct focus receive 51% of the visits across three stages, and with Llama-3.2-3B, the box's room stage receives almost the same visits as under CDUP.

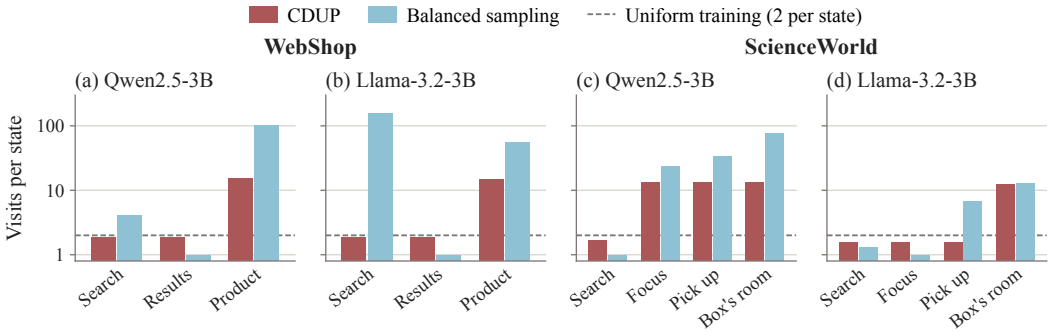


Figure 6: Visits per state under CDUP and balanced sampling, from Table 10, on a logarithmic scale; uniform training visits every state twice (dashed line).

Table 11 reports the results. On ScienceWorld, balanced sampling improves on CD with both models (paired McNemar $p = 7.1 \times 10^{-10}$ for Llama-3.2-3B and 5.2×10^{-9} for Qwen2.5-3B) and does not differ significantly from CDUP. On WebShop, it falls below CDUP with both models. With Llama-3.2-3B it still improves on CD ($p = 3.1 \times 10^{-8}$). With Qwen2.5-3B its success falls below

Table 10: Training visits under balanced sampling. For each stage, the number of training states and the mean visits per state under CDUP and balanced sampling. Uniform training visits every state twice; all three allocations share the total visits and optimizer updates. Bold marks the CDUP bottleneck states.

Benchmark	Model	Stage	States	CDUP	Balanced
WebShop	Qwen2.5-3B	Search	1,293	1.9	4.2
		Results page	8,216	1.9	1.0
		Product page	53	15.4	102.9
	Llama-3.2-3B	Search	10	1.9	160.8
		Results page	3,137	1.9	1.0
		Product page	29	15.0	55.4
ScienceWorld	Qwen2.5-3B	Search	12,741	1.7	1.0
		Focus	192	13.3	23.5
		Pick up	133	13.3	33.9
		Box's room	59	13.3	76.3
	Llama-3.2-3B	Search	1,604	1.6	1.3
		Focus	2,002	1.6	1.0
		Pick up	295	1.6	6.9
		Box's room	157	12.6	12.9

that of CD (4.7% versus 7.8%, $p = .096$), although its task score is higher. The trained agents attempt purchases about as often as or more often than CDUP, and Qwen2.5-3B clicks *buy now* in 317 of 384 episodes against 287 for CDUP, but they buy the requested product less often.

Read together, the two benchmarks point the same way. Where the rarely reached stages hold enough states to absorb an equal share, balanced sampling matches CDUP. Where they hold very few states, the equal share upsamples them far beyond the ratio of CDUP and takes the visits from the rest of the task, and success falls. The gain therefore comes from upsampling the bottleneck states, and the ratio of CDUP keeps the upsampling within a useful range. How to set it without a hand-chosen value is left to future work.

Table 11: Balanced sampling. Success and score (WebShop) or progress (ScienceWorld) in percent over 384 held-out episodes; the CD and CDUP columns repeat Table 2. The last columns compare CDUP with balanced sampling: won / lost counts episodes solved only by CDUP / only by balanced sampling, with the two-sided exact McNemar test.

		CD (uniform)		CDUP		Balanced sampling		CDUP vs. balanced		
Benchmark	Model	Success	Score	Success	Score	Success	Score	Won	Lost	p
WebShop	Llama-3.2-3B	2.3	3.4	**18.8**	**37.3**	11.5	32.3	54	26	.0023
WebShop	Qwen2.5-3B	7.8	25.3	**10.9**	**40.4**	4.7	34.9	36	12	7.2×10^{-4}
ScienceWorld	Llama-3.2-3B	21.4	53.4	**38.3**	**60.2**	35.9	58.0	51	42	.41
ScienceWorld	Qwen2.5-3B	15.4	46.1	25.3	52.8	**29.4**	**53.7**	45	61	.14

D.4 Random-states control and loss weighting

Table 12 gives the numbers behind Figure 3.

Loss weighting. Uniform training visits every state twice. To keep the total fixed, CDUP gives each bottleneck state 12 to 16 visits and each other state one or two. Loss weighting keeps the visits of uniform training, in the same order, and multiplies the loss of every state by its weight, the ratio of its visits under CDUP to its visits under uniform training. The bottleneck states therefore receive weights between 6 and 8 and the other states 0.5 or 1, and the weights of all visits sum to the total number of visits. Each training sequence contributes its mean token loss times its weight, and each update averages these weighted losses over its 64 sequences without renormalizing by the

Table 12: Random-states control and loss weighting, the numbers behind Figure 3. The per-state visit counts of CDUP are assigned to an equal number of randomly chosen non-bottleneck states, so the control shares the states, the total visits, the optimizer updates, and the distribution of visit counts with CDUP and differs only in which states are upsampled. Loss weighting keeps the uniform visits of CD and weights the loss of every state by the ratio of its CDUP to its uniform visits. Success and score (WebShop) or progress (ScienceWorld) in percent over 384 held-out episodes; the CD and CDUP columns repeat Table 2.

		CD (uniform training)		Control (random states)		Loss weighting		CDUP (bottleneck states)	
Benchmark	Model	Success	Score	Success	Score	Success	Score	Success	Score
WebShop	Llama-3.2-3B	2.3	3.4	3.4	6.3	9.1	27.3	**18.8**	**37.3**
WebShop	Qwen2.5-3B	7.8	25.3	9.6	20.9	**12.5**	36.7	10.9	**40.4**
ScienceWorld	Llama-3.2-3B	21.4	53.4	20.8	53.5	35.2	57.8	**38.3**	**60.2**
ScienceWorld	Qwen2.5-3B	15.4	46.1	17.2	49.9	19.0	48.8	**25.3**	**52.8**

weights, so a state with weight w contributes as much in expectation as w visits to it under the same parameters. Learning rate, gradient clipping, and the number of updates are those of CD and CDUP.

D.5 PAIRED STATISTICS

Table 13: Paired success contrasts on the shared evaluation episodes. Won / lost counts episodes solved only by the first policy / only by the second. All p-values use the two-sided exact McNemar test. CDUP uses ratio 8; random denotes its matched random-states control, and loss weighting keeps uniform visits and weights each state's loss to its CDUP total.

Benchmark and model	Contrast	Won	Lost	p
WebShop, Qwen2.5-3B	CD vs. base	30	0	1.9×10^{-9}
	OPCD vs. base	28	0	7.5×10^{-9}
	CDUP vs. CD	36	24	.155
	Random vs. CD	25	18	.360
	CDUP vs. random	30	25	.590
	Loss weighting vs. CD	38	20	.025
	CDUP vs. loss weighting	27	33	.52
WebShop, Llama-3.2-3B	CD vs. base	9	0	.0039
	OPCD vs. base	10	0	.0020
	CDUP vs. CD	65	2	3.1×10^{-17}
	Random vs. CD	6	2	.289
	CDUP vs. random	61	2	4.4×10^{-16}
	Loss weighting vs. CD	28	2	8.7×10^{-7}
	CDUP vs. loss weighting	51	14	4.5×10^{-6}
	Upsampled OPCD vs. OPCD	18	3	.0015
ScienceWorld, Qwen2.5-3B	CD vs. base	46	12	8.2×10^{-6}
	OPCD vs. base	54	8	1.7×10^{-9}
	CDUP vs. CD	54	16	5.9×10^{-6}
	Random vs. CD	25	18	.36
	CDUP vs. random	54	23	5.4×10^{-4}
	Loss weighting vs. CD	34	20	.076
	CDUP vs. loss weighting	49	25	.0071
	In-context teacher vs. CDUP	102	23	4.6×10^{-13}
	CDUP vs. OPCD	52	26	.004
ScienceWorld, Llama-3.2-3B	CD vs. base	59	14	1.0×10^{-7}
	OPCD vs. base	63	19	1.1×10^{-6}
	CDUP vs. CD	78	13	1.8×10^{-12}
	Random vs. CD	24	26	.888
	CDUP vs. random	82	15	2.4×10^{-12}
	Loss weighting vs. CD	72	19	2.0×10^{-8}
	CDUP vs. loss weighting	52	40	.25
	In-context teacher vs. CDUP	101	32	1.6×10^{-9}
	CDUP vs. OPCD	88	22	1.6×10^{-10}

Paired score tests. On WebShop with Qwen2.5-3B, CDUP improves task score over uniform training ($p = 3.5 \times 10^{-9}$) and the random control ($p = 7.4 \times 10^{-16}$). On ScienceWorld with Qwen2.5-3B, the corresponding progress tests give $p = 1.3 \times 10^{-6}$ and .0024. With Llama-3.2-3B, ScienceWorld progress also improves over uniform and random ($p = 2.1 \times 10^{-5}$ and .0016). These are two-sided sign tests on non-tied paired scores.

Table 14: Additional paired success contrasts. The external-teacher rows use the bottleneck upsampling and random-states control of Table 3b.

Benchmark and model	Contrast	Won	Lost	p
ALFWorld, Qwen2.5-3B	CD vs. base	59	20	1.3×10^{-5}
	OPCD vs. base	51	17	4.5×10^{-5}
ALFWorld, Llama-3.2-3B	CD vs. base	41	20	.0099
	OPCD vs. base	34	22	.14
WebShop, external teacher, Qwen2.5-3B	Upsampled vs. uniform	75	4	5.2×10^{-18}
	Upsampled vs. random	76	4	2.8×10^{-18}
	$\kappa = 32$ vs. uniform	98	4	1.7×10^{-24}
	$\kappa = 32$ vs. $\kappa = 8$	50	27	.012

E Experimental details

E.1 Experience context

The experience contexts of all benchmarks are released with the code. As an example, both models on ALFWorld receive the following guidance alongside the ordinary prompt. It names two canonical task endings and the appliance required for each operation. The only per-state substitution repeats the current admissible actions next to the generation point.

```
Finish rather than inspect. If the CURRENT list contains the exact
task target's next valid take, required clean/heat/cool, destination
move, or `use desklamp` command, execute it now. Otherwise establish
that command's location precondition with one listed `go to` or `open`
action. A move is possible only while holding the target at the
destination.

Canonical endings are: `take target` -> `go to destination` -> `move
target to destination`; or `take target` -> `go to appliance` ->
required operation -> `go to destination` -> `move`. Use sinkbasin
only for clean, microwave only for heat/hot, and fridge only for cool.
After a successful move, leave that instance there. A two-object task
needs the same ending for two different numbered instances.

Do not choose examine, inventory, look, an unrelated object, or a
repeated no-op when a task-progress action is available. Copy exactly
one CURRENT action:
{{CURRENT_ADMISSIBLE_ACTIONS}}
```

E.2 Training and evaluation

Table 15: Data and training budgets. *Targets* are the unique retained training targets; *visits* and *updates* are shared by the uniform and CDUP allocations and by the random-states control of Table 12. For the external-teacher rows, updates in parentheses give the upsampled and random-states allocations.

Benchmark	Model	Deploy. ep.	States	Targets	Visits	Updates	LR	Eval ep.
WebShop	Llama-3.2-3B	1,440	16,911	3,176	6,352	100	10^{-5}	384
WebShop	Qwen2.5-3B	1,440	16,911	9,562	19,124	299	10^{-5}	384
WebShop, teacher trajectories	Qwen2.5-3B	1,440	16,917	16,911	33,822	530	10^{-5}	384
WebShop, teacher targets, deployed states	Qwen2.5-3B	1,440	16,911	16,032	32,064	502 (501)	10^{-5}	384
ALFWorld	Llama-3.2-3B	1,440	18,874	15,018	30,036	470	10^{-5}	420
ALFWorld	Qwen2.5-3B	1,440	18,874	18,826	37,652	590	10^{-5}	420
ScienceWorld	Llama-3.2-3B	512	4,660	4,058	8,116	127	10^{-5}	384
ScienceWorld	Qwen2.5-3B	512	13,233	13,125	26,250	411	10^{-5}	384

Trajectories and evaluation sets. Every benchmark uses a history window of five steps, a 30-step limit, and temperature 0.4 for collection and evaluation, and its evaluation split is disjoint from the split the trajectories were collected on. WebShop: the deployed agent runs each of the 180 training

sessions eight times (1,440 episodes), from which 16,911 states are sampled uniformly with at least one per episode; evaluation uses 128 held-out sessions, three episodes each. ALFWorld: 1,440 deployment episodes on the training games; evaluation traverses the 140 seen-validation games three times. ScienceWorld runs with the environment's `easy` simplification preset. Its seven task types are those whose gold solution fits the 30-step budget and that have at least 30 training variations (`find-animal`, `find-living-thing`, `find-non-living-thing`, `find-plant`, `lifespan-longest-lived`, `lifespan-shortest-lived`, `lifespan-longest-lived-then-shortest-lived`); the deployed agent runs 256 training variations twice; evaluation uses 128 test variations of the official split, three episodes each.

Training. Supervised runs use seed 17, a micro-batch of one sequence with 64-step gradient accumulation, and a maximum sequence length of 8,192 tokens (2,048 on ALFWorld). CD targets are complete reasoning-and-action responses generated at temperature 0.4 with a 512-token limit and retained when they project to an admissible action. Uniform and upsampled allocations use the same retained targets. Evaluation uses temperature 0.4 and a 512-token limit, covers the full manifest, and uses no fallback responses.

On-policy budgets. On WebShop and ScienceWorld, OPCD uses the CD visit manifests; ALFWorld OPCD visits all 18,874 states twice for both models. On ScienceWorld with Qwen2.5-3B, the manifest has 26,250 visits: OPCD trains on 410 full batches of 64, while CD also updates on the final partial batch, for 411 updates. With Llama-3.2-3B, the manifest has 8,116 visits: OPCD consumes 8,064 in 126 full batches and drops the final 52, while CD uses all visits in 127 updates. On WebShop with Llama-3.2-3B, OPCD uses 99 full batches against 100 supervised updates. Table 16 reports the GPU and wall-clock costs; the objectives are not compute matched.

External-teacher budgets. Spark-1.5B supplies 16,911 trajectory targets and relabels the Qwen2.5-3B deployed agent's states. Bottleneck upsampling and its random-states control use the stages and allocation of §2.5 on these 16,032 targets, so that the 50 bottleneck states receive 783 of the 32,064 visits at $\kappa = 8$ and 2,918 at $\kappa = 32$ (Table 15).

Table 16: Compute for the main trained policies and their upsampling-ratio and external-teacher controls. Each run used one NVIDIA GPU; wall-clock is the duration of the training job, to the minute. OPCD samples and scores a fresh student response at every visit and is not compute matched to CD.

Benchmark	Model	Trained policy	GPU	Wall-clock
WebShop	Llama-3.2-3B	CD	RTX PRO 6000	0:29
		CDUP	RTX PRO 6000	0:30
		Control (random states)	B200	0:13
		OPCD	RTX PRO 6000	3:39
		OPCD, upsampled	B200	1:42
WebShop	Qwen2.5-3B	CD	A40	4:56
		CDUP	A100	2:22
		CDUP, $\kappa = 32$	A40	4:55
		Control (random states)	RTX PRO 6000	1:32
		OPCD	B200	4:09
		SFT on teacher trajectories	RTX PRO 6000	2:21
		SFT on teacher targets, deployed states	RTX PRO 6000	2:43
		+ bottleneck upsampling	A100	3:51
		+ bottleneck upsampling, $\kappa = 32$	B200	1:08
		+ random-states control	A100	3:54
ALFWorld	Llama-3.2-3B	CD	A6000	2:49
		OPCD	B200	2:38
ALFWorld	Qwen2.5-3B	CD	RTX PRO 6000	1:12
		OPCD	B200	2:22
ScienceWorld	Llama-3.2-3B	CD	RTX PRO 6000	0:16
		CDUP	RTX PRO 6000	0:16
		Control (random states)	B200	0:09
		OPCD	B200	0:46
ScienceWorld	Qwen2.5-3B	CD	B200	0:37
		CDUP	B200	0:37
		CDUP, $\kappa = 32$	B200	0:37
		Control (random states)	RTX PRO 6000	0:53
		OPCD	B200	2:05

E.3 Evaluation prompt templates

The templates below show the ordinary policy input for all three benchmarks. ALFWorld and WebShop are shown with history; ScienceWorld is shown both at the first step and with history. Each step is supplied as one user message, with up to five previous observation–action pairs. Placeholders are filled by the benchmark renderer. The WebShop template retains its original spelling and punctuation.

ALFWorld. One user message per step at history length five. The admissible-action list is supplied by the environment and is the only action vocabulary the policy may use.

```
You are an expert agent operating in the ALFRED Embodied Environment. Your task
is to: {task_description}
Prior to this step, you have already taken {step_count} step(s). Below are the
most recent {history_length} observations and the corresponding actions you
took: {action_history}
You are now at step {current_step} and your current observation is:
{current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning
```

```
process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for
current step and present it within <action> </action> tags.
```

WebShop. One user message per step at history length five. Deployment, teacher queries, training, and evaluation all use the Spark template below, so that the released Spark teacher of §4.3 acts under its own prompt.

```
You are an expert autonomous agent operating in the WebShop e-commerce
environment.
Your current task is: {task_description}

Prior to this step, you have already taken {step_count} step(s). Below are the
most recent {history_length} observaitons and the corresponding actions you
took: {action_history}
You are now at step {current_step} and your current observation is:
{current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].
To solve the problem, you have made the plan: [{planning}].

You have two reasoning modes:
<explore>
When results are unexpected or information is lacking, use current observations
to think outside the box and list as many possible locations, items, or actions
as possible.
</explore>

<think>
Continuously track the current progress and history of reasoning and execution
throughout the task.Typically used when task outcomes are as expected and no
other mode of reasoning is required.
</think>

You need to choose one of the two reasoning modes mentioned above as your
reasoning mode, based on its definition and applicable situation.

Now it's your turn to first reason and then take an action.
1.You should first reason about the current situation in one concise, brief
sentence. This reasoning process MUST be enclosed within ONLY ONE of the above
reasoning tags, like: <tag> your reasoning process </tag>.
2.Once you've finished your reasoning, you should choose an admissible action
for current step and present it within <action> </action> tags.
```

ScienceWorld. Both models use the same ordinary prompt. At the first step, the renderer uses the following template without history:

```
You are an expert agent operating in the ScienceWorld environment,
which is a text-based virtual environment centered around
accomplishing tasks from the elementary science curriculum.
Your current task is: {task_description}
Your current observation is: {current_observation}
Your admissible actions of the current situation are:
{admissible_actions}

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This
reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible
action for current step and present it within <action> </action> tags.
```

At subsequent steps, the prompt includes the most recent observation–action pairs:

```
You are an expert agent operating in the ScienceWorld environment,
which is a text-based virtual environment centered around
accomplishing tasks from the elementary science curriculum.
Your current task is: {task_description}
Prior to this step, you have already taken {step_count} step(s). Below
are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is:
{current_observation}
```

```
Your admissible actions of the current situation are:
{admissible_actions}

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This
reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an appropriate
action for the current step and present it within <action> </action>
tags.
```

Here, `step_count` is the number of actions already taken, `current_step` is that count plus one, and `history_length` is the number of retained pairs, capped at five. Each line of `action_history` records an observation and the action taken from it, using its original step number:

```
[Observation {step_number}: '{observation}', Action {step_number}:
'{action}']
```

The `admissible_actions` field contains action templates and the current object names as Python-style lists. Actions containing `OBJ` are instantiated with object names. The field is serialized as:

```
Valid_actions: {valid_actions}, OBJ needs to be replaced with one of
the following objects: {valid_objects}
 example: <action>focus on door</action>
```

Experience contexts are appended only for self-teacher queries and in-context evaluation. Trained policies receive the ordinary templates above.

WebShop original prompt. The benchmark also comes with the template below. So that the base policies are not put at a disadvantage by a template written for another agent, we evaluated the untrained checkpoints under this original template as well. Table 17 reports both evaluations on the same 384 held-out episodes. The original template raises the task score of both models and gives Qwen2.5-3B three successful purchases, while Llama-3.2-3B still buys nothing. Our statements are unaffected, because they compare training methods that share a prompt. The reference point for CDUP is uniform training on the same targets rather than the untrained checkpoint.

Table 17: WebShop base policies under the original and Spark prompts, 384 held-out episodes per row. Success, task score, episode reach of *buy now*, and the share of responses that project to an admissible action are in percent.

Model	Prompt	Success	Score	Clicked buy now	Valid actions
Llama-3.2-3B	Original	0.0	1.0	2.3	45.5
Llama-3.2-3B	Spark	0.0	0.0	0.0	28.8
Qwen2.5-3B	Original	0.8	4.6	9.6	98.7
Qwen2.5-3B	Spark	0.0	0.0	0.0	99.7

```
You are an expert autonomous agent operating in the WebShop e-commerce
environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s). Below are the
most recent {history_length} observations and the corresponding actions you
took: {action_history}
You are now at step {current_step} and your current observation is:
{current_observation}.
Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step.
You should first reason step-by-step about the current situation, then think
```

```
carefully which admissible action best advances the shopping goal. This
reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for
current step and present it within <action> </action> tags.
```