ENVIRONMENTAL FEEDBACK MODELING MATTERS: RETHINKING FEEDBACK TREATMENT IN AGENTIC HINDSIGHT SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning is commonly used to train language agents in interactive environments, but cannot be directly applied when rewards are unavailable. Recent methods use environmental feedback as privileged context for hindsight self-distillation, but our analysis suggests that simply conditioning the teacher on feedback is insufficient, motivating us to rethink how environmental feedback is used in agentic self-distillation. Given that environmental feedback contains rich supervision for modeling how the environment responds to agent actions, we introduce *agentic SElf-distilLation with environmental Feedback modeling* (SELF), a framework that jointly optimizes environmental feedback modeling and hindsight self-distillation. SELF learns to predict environmental responses while distilling guidance from a feedback-conditioned self-teacher into the policy. Our analysis reveals a mutually reinforcing mechanism: environmental feedback modeling strengthens hindsight supervision and policy learning, while self-distillation enhances the model's ability to model environmental feedback. With Qwen3-8B, SELF outperforms SDPO and GRPO by 6.4 and 4.1 percentage points in τ-bench success rate, and by 10.71 and 3.57 percentage points in AppWorld task goal completion, respectively. These results show that SELF uses environmental feedback more effectively within agentic self-distillation, improving agent capabilities.

1 INTRODUCTION

In recent years, training language agents to solve complex tasks in interactive environments involving tool use and code execution has become increasingly important (Yao et al., 2023; Trivedi et al., 2024; Choudhury & Sodhi, 2025). A common training approach is Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2025), where algorithms such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024) use trajectory-level rewards to optimize the policy. In some settings, however, reliable rewards are unavailable, and environmental feedback becomes the primary source of external information for agent training. This feedback arises naturally from interactions and includes tool outputs, code execution results, and environmental state changes caused by the agent's actions. Since such feedback typically describes action consequences rather than providing direct supervision for policy learning, using it effectively in agent training remains a challenge.

Recent methods tackle this challenge through hindsight self-distillation. SDPO (Hübotter et al., 2026) conditions a self-teacher on environmental feedback, while OEL (Ye et al., 2026) further extracts transferable knowledge from interaction experience. Our analysis shows that, despite containing sufficient information for policy improvement, such feedback yields only modest gains when used in these ways, suggesting that the key limitation lies in how it is exploited during training. Although environmental feedback does not directly supervise action generation, it provides direct targets for learning how the environment responds to agent actions. We therefore ask: *Can learning to predict environmental feedback enable more effective use of it in agentic self-distillation?*

Our analysis reveals a mutually reinforcing relationship: environmental feedback modeling improves hindsight self-distillation, while self-distillation in turn strengthens environmental feedback prediction. Building on this finding, we introduce *agentic SElf-distilLation with environmental Feedback* (**SELF**), a framework that jointly optimizes environmental feedback modeling and hindsight

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.

self-distillation. For each interaction transition, SELF uses the realized feedback in two complementary ways: as a prediction target for learning how the environment responds to the agent's action, and as privileged context for a self-teacher whose action distribution is distilled into the student policy. By jointly learning from these two signals within a shared model, SELF enables more effective use of environmental feedback for policy improvement.

We evaluate SELF on τ-bench Retail (Yao et al., 2025) for multi-turn customer-service interactions with domain-specific tools, and AppWorld (Trivedi et al., 2024) for code-based task completion across applications. Across Qwen3-4B-Instruct and Qwen3-8B (non-thinking) (Yang et al., 2025), SELF outperforms feedback-learning baselines and reward-trained GRPO (Shao et al., 2024) in task success and goal completion. With Qwen3-8B, SELF surpasses SDPO (Hübotter et al., 2026) and GRPO by 6.4 and 4.1 percentage points in τ-bench success rate, and by 10.71 and 3.57 percentage points in AppWorld task goal completion, respectively, which demonstrates SELF's effectiveness in turning environmental feedback into useful supervision for policy improvement. Further analyses show that SELF also retains strong generalization capabilities beyond the training environments.

Our key contributions can be summarized as follows:

- We identify a mutually reinforcing mechanism: environmental feedback modeling improves hindsight self-distillation, while self-distillation strengthens environmental feedback prediction.
- We propose SELF, an agent training framework that learns from environmental feedback by jointly optimizing environmental feedback modeling and hindsight self-distillation.
- We empirically show that SELF improves performance on τ-bench Retail and AppWorld, outperforming both existing self-distillation approaches and reward-trained GRPO, which demonstrates its effectiveness in enabling agentic self-distillation to make better use of environmental feedback.

2 Related Work

Agentic Self-Distillation. Agentic self-distillation provides dense policy supervision without requiring a separate stronger teacher. OPSD (Zhao et al., 2026) constructs student and teacher policies from the same model, conditioning the teacher on privileged reference information. SDPO (Hübotter et al., 2026) instead conditions the self-teacher on textual feedback and distills its feedback-informed predictions into the policy. SDFT (Shenfeld et al., 2026) uses demonstration-conditioned self-distillation to acquire new knowledge and skills while mitigating forgetting. RLSD (Yang et al., 2026a) further combines self-distillation with RLVR, using privileged teachers for token-level credit while retaining verifier rewards to determine the policy-update direction.

Learning from Environmental Feedback. Environmental feedback contains rich information about the realized consequences of agent actions, and recent methods exploit this information in different forms. OCSD (Yang et al., 2026b) contrasts replay contexts with and without future observations to calibrate policy updates, while HERO (Liu et al., 2026) converts subsequent observations into turn-level hindsight diagnoses. OEL (Ye et al., 2026) extracts transferable experiential knowledge from interaction trajectories before distilling it into the policy. Kleine Buening et al. (Kleine Buening et al., 2026) use raw user follow-up messages as hindsight information for self-distillation. AHEAD (Jin et al., 2026) conditions the teacher on environment feedback and supplements critical error steps with generated corrective hints, whereas OpenClaw-RL (Wang et al., 2026) converts next-state signals into process rewards or hindsight guidance for online agent learning. These approaches use realized feedback to construct or refine supervision for action generation.

Environmental Feedback Modeling for Policy Learning. A complementary line of work directly learns the predictive structure contained in interaction feedback. ECHO (Shrivastava et al., 2026) and PaW (Lu et al., 2026) incorporate environment prediction into policy optimization, while AAWM (Cai et al., 2026) studies decision-oriented modeling targets rather than directly reconstructing future observations. Early Experience (Zhang et al., 2026) uses next states produced by agent actions as supervision for implicit world modeling and self-reflection. Most closely related, RLTF (Song et al., 2026) studies both feedback-conditioned self-distillation and feedback modeling, where textual critiques are predicted as an auxiliary objective. In contrast, SELF models environmental responses directly from interactive trajectories and uses the same realized feedback both as a prediction target and as privileged context for hindsight self-distillation. Our analysis further shows that these objectives mutually reinforce each other.

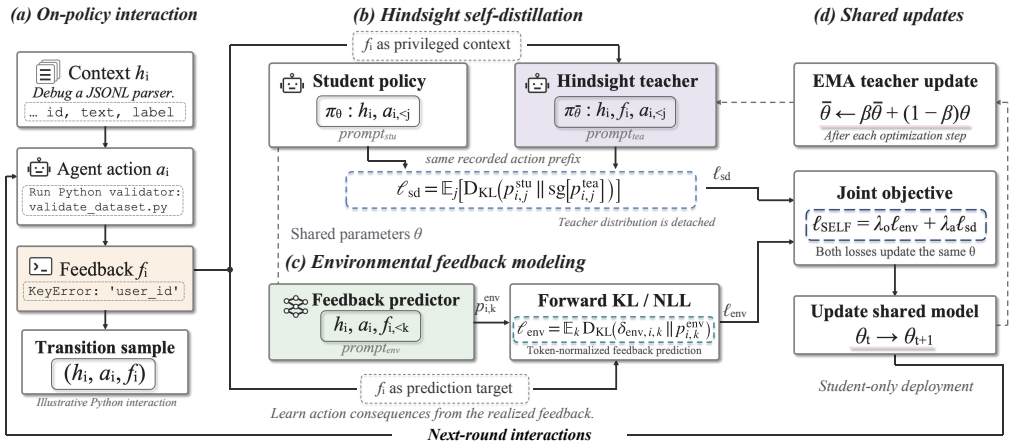


Figure 1: Overview of SELF. The current policy interacts with the environment to collect transitions (h_i, a_i, f_i). Each transition is used for two complementary objectives: hindsight self-distillation, where a feedback-conditioned EMA teacher supervises the student policy, and environmental feedback modeling, where the model predicts the realized feedback from the interaction history and action. The two objectives are jointly optimized to obtain the next-round policy.

3 METHODOLOGY

We present SELF, a framework that improves agentic self-distillation by explicitly modeling environmental feedback. We first formalize agent–environment interaction and the problem of learning from environmental feedback. We then describe on-policy hindsight self-distillation, which uses realized feedback as privileged information to construct policy supervision. Finally, we introduce the environmental feedback modeling objective and show how it is jointly optimized with hindsight self-distillation within the framework.

3.1 PROBLEM FORMULATION

Agent–Environment Interaction. We consider an agent that interacts with an environment through actions and observable feedback, including tool outputs, code execution results, and state changes. Let q denote a task instruction, o_0 the initial observation, and f_i the feedback returned after action a_i. Before taking a_i, the agent has access to the interaction history

$$h_i = (q, o_0, a_1, f_1, \ldots, a_{i-1}, f_{i-1}). \tag{1}$$

At training round t, the current policy and environment generate

$$a_i \sim \pi_{\theta_t}(\cdot \mid h_i, \mathrm{prompt}_{\mathrm{stu}}), \;\; f_i \sim \mathcal{P}_{\mathrm{env}}(\cdot \mid h_i, a_i), \tag{2}$$

where π_{θ_t} is the policy with parameters θ_t, $\mathrm{prompt}_{\mathrm{stu}}$ specifies the action-generation interface, and $\mathcal{P}_{\mathrm{env}}$ is the conditional distribution of environmental feedback. Each transition (h_i, a_i, f_i) associates an action with its observed consequence, without requiring access to latent environment states.

A trajectory is $\tau = (q, o_0, a_1, f_1, \ldots, a_{n_\tau}, f_{n_\tau})$, where n_τ is the number of interaction turns, and $\mathcal{T}_t$ denotes the trajectories collected at round t. Training uses only the resulting actions and environmental feedback, without task rewards, outcome labels, or external teachers. For feedback modeling, the observed feedback f_i is represented by the point-mass target $\delta_{\mathrm{env},i}(f) = \mathbf{1}\{f = f_i\}$.

3.2 HINDSIGHT SELF-DISTILLATION

Hindsight self-distillation provides a general approach for converting environmental feedback into supervision for policy learning. We consider its on-policy form, where trajectories are collected by the current policy and distillation is performed on the action prefixes encountered along these trajectories (Agarwal et al., 2024). For each transition (h_i, a_i, f_i), the student predicts the action

using only the information available before the environmental feedback is observed, while a self-teacher additionally conditions on the realized feedback f_i as privileged hindsight information. The student and teacher use distinct prompting interfaces, $\mathrm{prompt}_{\mathrm{stu}}$ and $\mathrm{prompt}_{\mathrm{tea}}$, respectively.

For an action a_i containing L_i tokens, we define the student and teacher next-token distributions as

$$p^{\mathrm{stu}}_{i,j}(\cdot) = \pi_\theta\big(\cdot \mid h_i, a_{i,<j}, \mathrm{prompt}_{\mathrm{stu}}\big),\ p^{\mathrm{tea}}_{i,j}(\cdot) = \pi_{\bar\theta}\big(\cdot \mid h_i, f_i, a_{i,<j}, \mathrm{prompt}_{\mathrm{tea}}\big), \tag{3}$$

where both distributions are evaluated on the same on-policy action prefix $a_{i,<j}$, but only the teacher has access to the realized environmental feedback. The teacher parameters $\bar\theta$ are maintained as an exponential moving average (EMA) (Tarvainen & Valpola, 2017) of the student parameters, $\bar\theta \leftarrow \beta\bar\theta + (1-\beta)\theta$, where β is the EMA decay coefficient.

The feedback-conditioned teacher is distilled into the student through the reverse-KL objective (Gu et al., 2024)

$$\ell_{\mathrm{sd}}(h_i, a_i, f_i; \theta, \bar\theta) = \mathbb{E}_{j\sim\mathrm{Unif}([L_i])}\left[D_{\mathrm{KL}}\big(p^{\mathrm{stu}}_{i,j} \,\big\|\, \mathrm{sg}[p^{\mathrm{tea}}_{i,j}]\big)\right], \tag{4}$$

where $[L_i] = \{1, \ldots, L_i\}$ and sg stops gradients through the teacher. The collected action therefore determines the contexts at which the two distributions are compared, rather than serving as a ground-truth action target. This formulation allows information available only after an action is executed to supervise a policy that must act without access to that information at inference time (Choudhury & Sodhi, 2025). In our analysis, however, hindsight self-distillation in this form yields only modest policy improvements, suggesting that conditioning the teacher on feedback alone does not fully exploit the information contained in environmental feedback.

Why Environmental Feedback Modeling? Hindsight self-distillation uses environmental feedback as privileged context for constructing policy supervision, but does not explicitly learn the information contained in the feedback itself. Our analysis suggests that this limits how effectively the self-teacher can exploit environmental feedback.
Environmental feedback, however, provides rich supervision about how the environment responds to an agent's actions. Each interaction therefore naturally provides a supervised action–feedback relation,

$$(h_i, a_i) \longrightarrow f_i.$$

This motivates explicitly modeling environmental feedback alongside hindsight self-distillation, so that the model learns how the environment responds while using the same feedback to improve its policy.

3.3 Environmental Feedback Modeling

Motivated by the observation in the previous section, we treat each transition (h_i, a_i, f_i) not only as privileged context for hindsight self-distillation, but also as direct supervision for modeling how the environment responds to an action. We therefore jointly train environmental feedback modeling and agentic self-distillation.

Given (h_i, a_i, f_i), we approximate the conditional feedback distribution $\mathcal{P}_{\mathrm{env}}(\cdot \mid h_i, a_i)$ with $p_{\mathrm{env},\theta}(\cdot \mid h_i, a_i, \mathrm{prompt}_{\mathrm{env}})$, where $\mathrm{prompt}_{\mathrm{env}}$ is a dedicated prompt for environmental feedback prediction. The conditioning context contains the complete interaction history up to and including a_i, while the realized feedback f_i serves as the prediction target. This prediction interface shares parameters θ with the student policy, rather than introducing a separate environment model.

For a feedback sequence f_i containing M_i tokens, let $\delta_{\mathrm{env},i,k}$ denote the point-mass target on the observed token $f_{i,k}$. We define the autoregressive next-token distribution as

$$p^{\mathrm{env}}_{i,k}(\cdot) = p_{\mathrm{env},\theta}\big(\cdot \mid h_i, a_i, f_{i,<k}, \mathrm{prompt}_{\mathrm{env}}\big), \tag{5}$$

where $f_{i,<k}$ denotes the observed feedback prefix. The environmental feedback modeling objective is

$$\ell_{\mathrm{env}}(h_i, a_i, f_i; \theta) = \mathbb{E}_{k\sim\mathrm{Unif}([M_i])}\left[D_{\mathrm{KL}}\big(\delta_{\mathrm{env},i,k} \,\big\|\, p^{\mathrm{env}}_{i,k}\big)\right]. \tag{6}$$

Because each target is a point mass, this forward-KL objective is equivalent to the token-normalized negative log-likelihood of the realized feedback. It therefore trains the shared model to capture observable action–feedback relationships directly from interaction, without requiring latent environment states or treating the sampled action as a correct action target.

Table 1: Comparison of learning paradigms under the configurations considered in this work. Supervision describes the granularity of the learning signal. Teacher denotes the source of teacher supervision. Env. Modeling indicates whether an explicit objective predicts environmental responses from interaction histories and actions. Dashes indicate inapplicable attributes.

Method	Sampling	Supervision	Feedback Source	Teacher	Env. Modeling
SFT	Off-policy	Dense	Teacher	External	No
GRPO	On-policy	Sparse	Verifier	None	No
Mem0	–	–	Environment	None	No
SDPO	On-policy	Dense	Environment	Self	No
SELF (Ours)	On-policy	Dense	Environment	Self	Yes

3.4 Overall SELF Training Objective

The proposed **SELF** framework jointly optimizes environmental feedback modeling and hindsight self-distillation within a shared model. The same realized feedback f_i serves two complementary roles: it provides a prediction target for learning how the environment responds to an action, while also acting as privileged hindsight information for constructing policy supervision. By optimizing both objectives through the same underlying model, SELF jointly improves environmental feedback modeling and agent action generation. For each transition (h_i, a_i, f_i), the SELF objective is

$$\ell_{\mathrm{SELF}} = \lambda_o \ell_{\mathrm{env}} + \lambda_a \ell_{\mathrm{sd}}, \tag{7}$$

where λ_o and λ_a control the contributions of environmental feedback modeling and hindsight self-distillation, respectively.

At training round t, the current policy collects a set of trajectories $\mathcal{T}_t$. Averaging the token-normalized objectives over trajectories and interaction turns yields

$$\begin{aligned}\mathcal{L}_{\mathrm{SELF},t}(\theta) = \mathbb{E}_{\tau\sim\mathrm{Unif}(\mathcal{T}_t)}\mathbb{E}_{i\sim\mathrm{Unif}([n_\tau])}\Bigg[&\lambda_o \mathbb{E}_{k\sim\mathrm{Unif}([M_i])} D_{\mathrm{KL}}\big(\delta_{\mathrm{env},i,k} \,\big\|\, p^{\mathrm{env}}_{i,k}\big) \\ &+ \lambda_a \mathbb{E}_{j\sim\mathrm{Unif}([L_i])} D_{\mathrm{KL}}\big(p^{\mathrm{stu}}_{i,j} \,\big\|\, \mathrm{sg}[p^{\mathrm{tea}}_{i,j}]\big)\Bigg],\end{aligned} \tag{8}$$

where n_τ denotes the number of interaction turns in trajectory τ. Optimizing $\mathcal{L}_{\mathrm{SELF},t}$ updates the shared parameters from θ_t to θ_{t+1}. Environmental feedback modeling therefore updates the same model from which subsequent hindsight supervision is derived, while self-distillation directly improves the deployable policy and simultaneously updates the representation used for feedback prediction. The updated policy then collects new interactions for the next training round. At deployment, only the student policy $\pi_\theta(\cdot \mid h_i, \mathrm{prompt}_{\mathrm{stu}})$ is required. The complete training procedure is summarized in Algorithm 1 in the appendix, and illustrative examples of $\mathrm{prompt}_{\mathrm{stu}}$, $\mathrm{prompt}_{\mathrm{tea}}$, and $\mathrm{prompt}_{\mathrm{env}}$ are provided in the appendix.

4 Experiments

In this section, we evaluate SELF on τ-bench Retail and AppWorld and address the following three research questions (RQs). **RQ1:** Can SELF improve interactive agent performance using environmental feedback without verifiable rewards? **RQ2:** Do environmental feedback modeling and hindsight self-distillation mutually improve each other? **RQ3:** Does SELF retain general language-model capabilities after agent training?

4.1 Experimental Setup

Benchmark Datasets. We evaluate SELF on τ-bench Retail (Yao et al., 2025) and AppWorld (Trivedi et al., 2024), covering multi-turn customer-service interactions with domain-specific

Table 2: Performance on τ-bench and AppWorld. SR denotes the pass@4 success rate on τ-bench, and Turns denotes the average number of interaction turns. TGC and SGC denote task goal completion and scenario goal completion on AppWorld, respectively. Observation and Full denote baseline feedback settings, while Reward denotes reward-based training. SELF learns from agent-generated actions and environmental observations, and its rows are shaded light blue. Higher SR, TGC, and SGC and lower Turns are better. Within each backbone, excluding external model baselines, the best means are **bold** with pale gold shading, and the second-best means are underlined with pale pink shading. Missing results and inapplicable settings are denoted by –.

Method	Setting	τ-bench		AppWorld	
		SR ↑	Turns ↓	TGC ↑	SGC ↑
External Model Baselines					
Qwen3-30B-A3B-Instruct	–	33.6	11.9	31.55	12.50
Qwen3-4B-Instruct					
Base	–	27.2 ± 1.4	10.5 ± 0.3	16.67 ± 1.19	5.36 ± 0.62
OEL	Observation	30.6 ± 1.5	10.9 ± 0.4	19.05 ± 1.43	6.25 ± 0.74
SDPO	Observation	30.4 ± 1.7	11.5 ± 0.5	20.24 ± 1.56	7.14 ± 0.83
Mem0	Observation	19.2 ± 1.8	12.6 ± 0.6	11.90 ± 1.31	3.57 ± 0.58
SDPO	Full	22.5 ± 1.6	12.4 ± 0.5	14.29 ± 1.47	4.46 ± 0.69
Mem0	Full	27.4 ± 1.5	11.7 ± 0.4	17.86 ± 1.38	5.95 ± 0.77
GRPO	Reward	33.3 ± 1.9	13.4 ± 0.7	28.57 ± 1.72	10.71 ± 1.04
SELF (Ours)	Observation	**38.7** ± 1.5	**10.4** ± 0.4	**32.14** ± 1.61	**12.50** ± 0.96
Qwen3-8B (Non-Thinking)					
Base	–	35.7 ± 1.3	**10.0** ± 0.3	22.62 ± 1.28	7.14 ± 0.71
OEL	Observation	37.9 ± 1.4	10.4 ± 0.4	26.19 ± 1.46	8.04 ± 0.82
SDPO	Observation	36.8 ± 1.6	11.0 ± 0.4	25.00 ± 1.53	8.93 ± 0.89
Mem0	Observation	29.5 ± 1.7	12.0 ± 0.6	16.67 ± 1.35	4.36 ± 0.63
SDPO	Full	30.6 ± 1.5	11.8 ± 0.5	19.05 ± 1.42	6.25 ± 0.76
Mem0	Full	35.8 ± 1.4	11.1 ± 0.4	22.62 ± 1.39	7.14 ± 0.84
GRPO	Reward	39.1 ± 1.8	12.7 ± 0.6	32.14 ± 1.68	11.61 ± 1.02
SELF (Ours)	Observation	**43.2** ± 1.4	10.8 ± 0.4	**35.71** ± 1.57	**13.39** ± 0.93

tools and code-based task completion across applications, respectively. We use the official training splits for learning and held-out test splits for evaluation. On τ-bench Retail, we report success rate (SR; pass@4) and the average number of interaction turns (Turns) to assess task performance and interaction efficiency. On AppWorld, we report Task Goal Completion (TGC) and Scenario Goal Completion (SGC), measuring task-level success and consistency across task variants, respectively. We additionally evaluate MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024), and IFEval (Zhou et al., 2023) to assess the retention of general capabilities after agent training.

Comparing Baselines. We compare SELF with hindsight-distillation methods OEL (Ye et al., 2026) and SDPO (Hübotter et al., 2026), the memory-based method Mem0 (Chhikara et al., 2025), and the reinforcement learning method GRPO (Shao et al., 2024). We distinguish three supervision settings: *Observation* supplies only environmental observations, without ground-truth task outcomes or success/failure labels; *Full* additionally supplies ground-truth information and explicit success/failure feedback; and *Reward* provides verifiable task rewards for reinforcement learning. OEL and SELF use Observation, SDPO and Mem0 are evaluated under both Observation and Full, and GRPO uses Reward. Table 1 summarizes the methods' sampling strategies, supervision sources, and use of environment modeling. We also report unadapted base models and a strong external model to contextualize trained-agent performance: Qwen3-30B-A3B-Instruct (Yang et al., 2025).

Implementation Details. Our main experiments use Qwen3-4B-Instruct and Qwen3-8B (non-thinking) as backbone models (Yang et al., 2025). All ablation studies, mechanism analyses, and general-capability evaluations use Qwen3-4B-Instruct. Models evaluated in our framework oper-

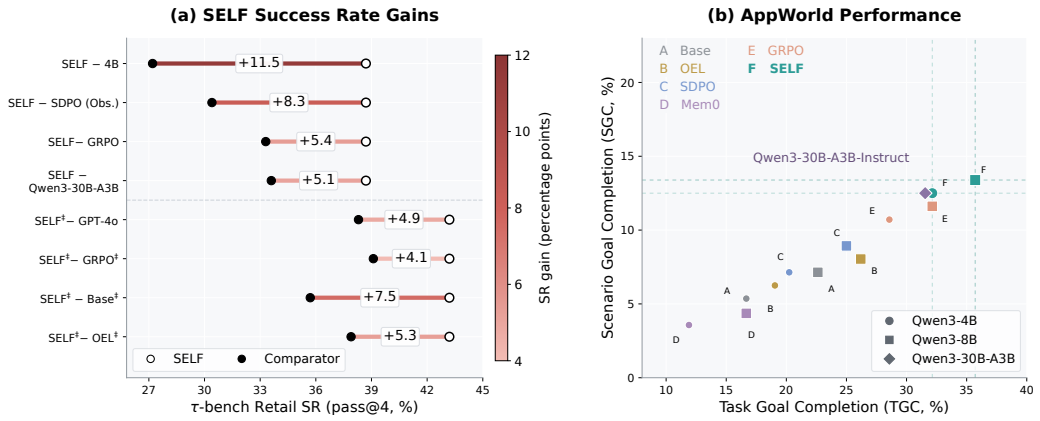


Figure 2: Performance comparison on τ-bench Retail and AppWorld. (a) Success rate (SR, pass@4) gains of SELF over selected comparators on τ-bench Retail. Open and filled circles denote SELF and comparator scores, respectively; annotations and line colors indicate gains in percentage points. The superscript ‡ denotes Qwen3-8B; unmarked methods use Qwen3-4B-Instruct unless otherwise specified. (b) Task goal completion (TGC) and scenario goal completion (SGC) on AppWorld. Colors distinguish methods, and marker shapes distinguish model backbones.

ate under the same ReAct-style tool-use interface (Yao et al., 2023). For SELF, the self-distillation teacher is maintained as an exponential moving average (EMA) (Tarvainen & Valpola, 2017) of the student, with decay coefficient $\beta = 0.99$. Unless otherwise specified, we use $\lambda_o = 0.2$ and $\lambda_a = 1.0$ as the default loss weights. We repeat each experiment over three independent runs and report the mean together with the standard deviation. Complete training hyperparameters and implementation details are provided in the appendix.

4.2 Main Results (RQ1)

We first evaluate whether SELF improves interactive agent performance using environmental feedback alone, comparing it with self-distillation, memory-based, and reward-trained baselines under different supervision settings. Table 2 and Figure 2 report task performance and interaction efficiency on τ-bench Retail and AppWorld using Qwen3-4B-Instruct and Qwen3-8B (non-thinking). Under Observation, SELF consistently outperforms SDPO and OEL on SR, TGC, and SGC. With Qwen3-4B-Instruct, SELF achieves 38.7% SR, exceeding SDPO and OEL by 8.3 and 8.1 percentage points; the corresponding gains with Qwen3-8B are 6.4 and 5.3 points. These results indicate more effective use of environmental feedback than existing self-distillation methods. SELF also surpasses Mem0 under both Observation and Full: with Qwen3-4B-Instruct, Mem0 achieves 19.2% and 27.4% SR, respectively, versus SELF's 38.7%. SELF further outperforms SDPO under Full feedback. Indeed, Full feedback reduces SDPO's SR from 30.4% to 22.5% with Qwen3-4B-Instruct and from 36.8% to 30.6% with Qwen3-8B, alongside lower AppWorld TGC and SGC. Additional ground-truth information and success/failure labels therefore do not necessarily improve feedback-based learning. Finally, SELF exceeds reward-trained GRPO by 5.4 and 4.1 SR points on the two backbones, respectively, while also achieving higher TGC and SGC. Together, these comparisons demonstrate effective policy learning from environmental feedback without verifiable rewards.

4.3 Mutual Benefits of Feedback Modeling and Self-Distillation (RQ2)

We examine the interaction between the two objectives in SELF. Specifically, we test whether environmental feedback modeling improves feedback-derived hindsight supervision, and whether hindsight self-distillation improves the shared model's ability to predict environmental feedback.

Environmental feedback modeling strengthens the hindsight teacher and improves feedback-derived supervision. We vary the observation-modeling weight $\lambda_o \in \{0, 0.1, 0.2, 0.5, 1.0\}$ while fixing the self-distillation weight, using Qwen3-4B-Instruct on τ-bench Retail. Table 3 and Figure 3(a) report student and privileged-teacher performance; the teacher evaluation protocol is de-

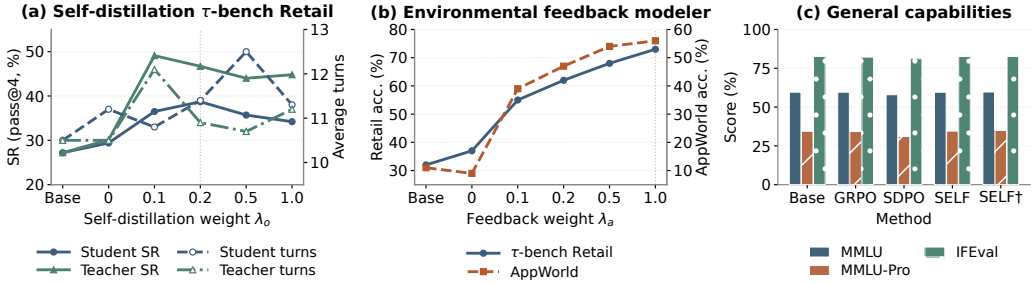


Figure 3: Ablation studies and general capability evaluation with Qwen3-4B-Instruct. (a) Student and privileged teacher success rates (SR, pass@4) and average interaction turns on τ-bench Retail across different values of λ_o. (b) Environmental feedback prediction accuracy on τ-bench Retail and AppWorld across different values of λ_a. (c) Performance on MMLU, MMLU-Pro, and IFEval; SELF† denotes the privileged teacher. Vertical dashed lines in (a) and (b) indicate default settings.

Table 3: Effect of the environmental feedback modeling weight λ_o on τ-bench Retail with Qwen3-4B-Instruct. SR is reported as pass@4 (%), and Turns denotes the average number of interaction turns. **Bold** indicates the best result in each column.

Setting	Student		Privileged teacher	
	SR ↑	Turns ↓	SR ↑	Turns ↓
Base	27.2 ± 1.4	10.5 ± 0.3	–	–
$\lambda_o = 0$	29.4 ± 2.3	11.2 ± 0.5	30.1 ± 1.1	$\mathbf{10.5} \pm 0.2$
$\lambda_o = 0.1$	36.5 ± 0.9	10.8 ± 0.4	$\mathbf{49.1} \pm 1.8$	12.1 ± 0.6
$\lambda_o = 0.2$	$\mathbf{38.7} \pm 1.5$	$\mathbf{10.4} \pm 0.4$	46.7 ± 2.4	10.9 ± 0.4
$\lambda_o = 0.5$	35.7 ± 2.1	12.5 ± 0.6	44.0 ± 1.3	10.7 ± 0.3
$\lambda_o = 1.0$	34.2 ± 1.2	11.3 ± 0.3	44.8 ± 2.0	11.2 ± 0.5

tailed in the appendix. At the default $\lambda_o = 0.2$, the student achieves its highest SR of 38.7%, exceeding Base by 11.5 percentage points, while the teacher reaches 46.7%. The teacher peaks at 49.1% with $\lambda_o = 0.1$, but the corresponding student achieves only 36.5%, showing that higher privileged-teacher performance does not necessarily produce the best distilled student. Compared with no observation modeling ($\lambda_o = 0$), the default improves teacher SR from 30.1% to 46.7% and student SR from 29.4% to 38.7%, gains of 16.6 and 9.3 points, respectively. These results support the interpretation that feedback modeling strengthens the feedback-conditioned teacher and its ability to provide effective policy supervision, with the student gains demonstrating benefits for hindsight self-distillation, even when learning relies on observed environmental responses.

Self-distillation strengthens environmental feedback modeling, supporting the reverse direction of their mutually reinforcing relationship. We vary the self-distillation weight $\lambda_a \in \{0, 0.1, 0.2, 0.5, 1.0\}$ while fixing λ_o, using Qwen3-4B-Instruct on τ-bench Retail and AppWorld. We measure feedback prediction accuracy (Acc.) on a shared, fixed set of held-out transitions collected by a reference policy in each environment. All variants receive identical histories and actions and are evaluated against the same observed feedback under identical evaluation settings (see appendix for details). As shown in Table 4 and Figure 3(b), accuracy increases monotonically across the tested λ_a values, peaking at $\lambda_a = 1.0$: 73% on Retail and 56% on AppWorld, compared with 37% and 9% without self-distillation. Policy performance improves alongside prediction accuracy: increasing λ_a from 0 to 1.0 raises Retail SR from 20.1% to 38.7% and AppWorld TGC from 5.52% to 32.14%. Without self-distillation, task performance falls below Base on both benchmarks, highlighting the importance of the policy-learning objective. Together with the λ_o ablation, results support a mutually reinforcing relationship: feedback modeling improves feedback-derived policy supervision, while hindsight self-distillation enhances the shared model's ability to predict action consequences. Feedback modeling benefits from policy learning while supporting it.

Table 4: Effect of the self-distillation weight λ_a on τ-bench Retail and AppWorld with Qwen3-4B-Instruct. SR is reported as pass@4, and Turns denotes the average number of interaction turns. All metrics except Turns are reported as percentages.

Setting	τ-bench Retail			AppWorld		
	SR ↑	Turns ↓	Acc. ↑	TGC ↑	SGC ↑	Acc. ↑
Base	27.2 ± 1.4	10.5 ± 0.3	32 ± 1.8	16.67 ± 1.19	5.36 ± 0.62	11 ± 1.3
$\lambda_a = 0$	20.1 ± 2.1	11.2 ± 0.5	37 ± 2.6	5.52 ± 0.83	1.21 ± 0.34	9 ± 1.7
$\lambda_a = 0.1$	31.5 ± 1.3	12.0 ± 0.6	55 ± 1.5	26.67 ± 2.07	8.62 ± 0.91	39 ± 2.4
$\lambda_a = 0.2$	33.2 ± 1.8	10.7 ± 0.3	62 ± 2.2	29.91 ± 1.43	8.45 ± 1.16	47 ± 1.6
$\lambda_a = 0.5$	36.5 ± 0.9	$\mathbf{10.2} \pm 0.2$	68 ± 1.3	31.10 ± 1.86	9.73 ± 0.78	54 ± 2.1
$\lambda_a = 1.0$	$\mathbf{38.7} \pm 1.5$	10.4 ± 0.4	$\mathbf{73} \pm 1.7$	$\mathbf{32.14} \pm 1.61$	$\mathbf{12.50} \pm 0.96$	$\mathbf{56} \pm 1.2$

Table 5: General capability evaluation on MMLU, MMLU-Pro, and IFEval using Qwen3-4B-Instruct. Results are mean ± standard deviation. Light and darker blue shading identify the SELF student and teacher, respectively. Higher is better; **bold** entries indicate the highest mean per column.

Method	**Setting**	**MMLU ↑**	**MMLU-Pro ↑**	**IFEval ↑**
Base	–	59.60 ± 0.18	34.38 ± 0.27	$\mathbf{82.62} \pm 0.35$
GRPO	Reward	59.61 ± 0.24	34.25 ± 0.36	82.19 ± 0.48
SDPO	Observation	57.96 ± 0.31	31.02 ± 0.42	81.17 ± 0.53
SELF	Observation	59.57 ± 0.22	34.49 ± 0.33	82.61 ± 0.41
SELF (teacher)	Observation	$\mathbf{59.71} \pm 0.20$	$\mathbf{34.92} \pm 0.29$	82.58 ± 0.38

4.4 General Capability Retention (RQ3)

Finally, we examine whether SELF's gains in interactive environments come at the expense of broader capabilities by evaluating the Qwen3-4B-Instruct student and EMA teacher on MMLU, MMLU-Pro, and IFEval, covering general knowledge, reasoning, and instruction following (Table 5 and Figure 3). The SELF student closely matches the base model: MMLU changes from 59.60 to 59.57, MMLU-Pro from 34.38 to 34.49, and IFEval from 82.62 to 82.61, with absolute differences of at most 0.11 points. In contrast, observation-based SDPO reduces these scores to 57.96, 31.02, and 81.17, respectively. The EMA teacher similarly retains general performance. These results support capability retention rather than general capability improvements, showing that SELF's interactive-task gains accompany minimal changes on the evaluated general benchmarks.

5 Conclusion

We introduced SELF, a framework that jointly optimizes environmental feedback modeling and hindsight self-distillation, using realized action consequences both as prediction targets for learning action–feedback relationships and as privileged context for self-teacher supervision. Experiments on τ-bench Retail and AppWorld show that SELF consistently outperforms existing feedback-learning methods and reward-trained GRPO while largely preserving the general capabilities of the base model. Our analysis further reveals a mutually reinforcing mechanism: environmental feedback modeling strengthens hindsight self-distillation, while self-distillation improves feedback prediction. These findings suggest that effective learning from environmental feedback requires not only conditioning on the observed feedback, but also explicitly learning the action–feedback regularities encoded in interaction. SELF therefore provides a simple framework for jointly learning how the environment responds and how the agent should act.

AI USE STATEMENT

Generative AI tools were used to assist with language drafting and editing based on author-provided research materials and experimental results. The authors reviewed the technical content, verified the reported results against the underlying experiments, and take responsibility for the final text and claims. We did not use generative AI tools to formulate or prove mathematical claims, nor did we use them to generate mathematical proofs.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5be69a584901a26c521c2b51e40a4c20-Paper-Conference.pdf.

Guangfeng Cai, Kaibing Yang, Shuo He, Yu Li, Shengtian Yang, Jiaqi Lv, and Lei Feng. Beyond next-observation prediction: Agent-authored world modeling for sequential decision making. *arXiv preprint arXiv:2606.25421*, 2026. doi: 10.48550/arXiv.2606.25421. URL https://arxiv.org/abs/2606.25421.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI 2025*, pp. 2993–3000, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: LLM agents that learn from privileged AI feedback. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1c60ed2b01120d383eebf12dc7a0e138-Abstract-Conference.html.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=QkfkxyRizZ.

Xiaolong Jin, Dingmin Wang, Vijay Lingam, and Varun Kumar. Ahead: Adaptive hindsight with environment-augmented distillation for agentic rl. *arXiv preprint arXiv:2608.24114*, 2026. doi: 10.48550/arXiv.2608.24114.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions. *arXiv preprint arXiv:2603.12273*, 2026. doi: 10.48550/arXiv.2603.12273.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi.

Tülu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Haoran Liu, Yuwei Zhang, Xiyao Li, Bohan Lyu, and Jingbo Shang. HERO: Hindsight-Enhanced Reflection from Environment Observations for Agentic Self-Distillation. *arXiv preprint arXiv:2606.11559*, 2026. URL https://arxiv.org/abs/2606.11559.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Ning Lu, Baijiong Lin, Shengcai Liu, Jiahao Wu, Haoze Lv, Yanbin Wei, Lingting Zhu, Shengju Qian, Xin Wang, Ying-Cong Chen, Qi Wang, and Ke Tang. Policy and world modeling co-training for language agents. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026. URL https://arxiv.org/abs/2606.02388.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

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

Vaishnavi Shrivastava, Piero Kauffmann, Ahmed Awadallah, and Dimitris Papailiopoulos. Echo: Terminal agents learn world models for free. *arXiv preprint arXiv:2605.24517*, 2026. URL https://arxiv.org/abs/2605.24517.

Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J. Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*, 2026. doi: 10.48550/arXiv.2602.02482.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/68053af2923e00204c3ca7c6a3150cf7-Abstract.html.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026. doi: 10.48550/arXiv.2603.10165.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://openreview.net/forum?id=y10DM6R2r3#discussion.

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. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026a. doi: 10.48550/arXiv.2604.03128.

Yi Yang, Cong Qin, Xiaodan Liu, Chishui Chen, Qing Dong, Yan Zhang, Cao Liu, Zhao Yang, Lu Pan, Jiaye Lin, and Yi Feng. Agentic reinforcement learning with observation-calibrated self-distillation. *arXiv preprint arXiv:2608.04788*, 2026b. URL https://arxiv.org/abs/2608.04788.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik 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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

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

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Xuefei Cao, Lawrence Keunho Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason E. Weston, Yu Su, and Yifan Wu. Agent learning via early experience. In *Forty-third International Conference on Machine Learning*, 2026.

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 *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Jpxfof0EaS&noteId=qRewJdmNTV.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

A Datasets, Benchmarks, and Models

This section introduces the interactive environments, general capability benchmarks, and models used to evaluate SELF. Interactive environments assess learning from action consequences, while general capability benchmarks measure retention of knowledge, reasoning, and instruction following after agent training.

A.1 Datasets and Benchmarks

Interactive Benchmarks. We use τ-bench Retail and AppWorld to evaluate learning from agent–environment interactions. Learning uses the official training splits, and task performance is measured on held-out test splits. Training trajectories are collected by interacting with the environments, rather than provided as expert demonstrations. For SELF, the resulting observations serve as feedback-prediction targets and as hindsight context for self-distillation. Task-success annotations are reserved for evaluation.

- **τ-bench Retail.** The retail domain of τ-bench evaluates agents through multi-turn conversations with simulated users and calls to domain-specific tools (Yao et al., 2025). Agents must interpret user requests, obtain relevant information, and perform actions consistent with retail policies. Tool responses reveal the consequences of actions and inform subsequent decisions, but successful execution of a tool call does not necessarily establish that the user's overall request has been fulfilled. We use Retail to assess learning from tool feedback in policy-constrained interactions.
- **AppWorld.** AppWorld evaluates agents that generate and execute code to complete tasks across multiple applications (Trivedi et al., 2024). Tasks combine application APIs with program logic, and execution outputs guide subsequent actions. A single code-based action can compose multiple operations, creating dependencies both within actions and across interaction steps. Programmatic tests assess the resulting application states, including task completion and unintended changes. We use AppWorld to assess learning from code-execution feedback, complementing Retail's tool-based interactions.

General Capability Benchmarks. MMLU, MMLU-Pro, and IFEval are used exclusively for evaluation. We compare the initial Qwen3-4B-Instruct checkpoint, the trained SELF student, and its EMA teacher to assess changes in general capabilities after agent training. Each checkpoint receives only the inputs provided by the corresponding benchmark. These evaluations measure the teacher's general capabilities separately from its use of privileged hindsight information during interactive learning.

- **MMLU.** MMLU evaluates knowledge and problem solving through multiple-choice questions across 57 subjects, including mathematics, history, computer science, and law (Hendrycks et al., 2021). Its broad subject coverage extends beyond the retail and application domains used for agent training. We use it to assess whether interactive adaptation preserves general knowledge and question-answering ability.
- **MMLU-Pro.** MMLU-Pro emphasizes reasoning-intensive questions and increases the number of answer options to as many as ten (Wang et al., 2024). It complements MMLU with a more demanding assessment of reasoning and answer selection. We use it to examine whether these capabilities are retained after interactive training.
- **IFEval.** IFEval evaluates instruction following through programmatically verifiable requirements, such as constraints on response length and keyword usage (Zhou et al., 2023). It assesses whether generated responses satisfy explicit instructions without relying on a subjective model judge. We use it to measure retention of instruction-following behavior beyond tool use and code execution.

A.2 Models

We use Qwen3-4B-Instruct and Qwen3-8B as trainable backbones to evaluate SELF across model scales, and include Qwen3-30B-A3B-Instruct as external comparisons.

- **Qwen3-4B-Instruct.** Qwen3-4B-Instruct is an instruction-tuned language model in the Qwen3 family (Yang et al., 2025). We use it as a trainable backbone for the main interactive-task comparisons, coefficient ablations, mechanism analyses, and general capability evaluations.
- **Qwen3-8B.** Qwen3-8B is an open-weight language model in the Qwen3 family (Yang et al., 2025). We use its non-thinking configuration as a second trainable backbone to evaluate SELF on Retail and AppWorld at a larger model scale.
- **Qwen3-30B-A3B-Instruct.** Qwen3-30B-A3B-Instruct is an instruction-tuned mixture-of-experts model in the Qwen3 family (Yang et al., 2025). We report its performance on Retail and AppWorld to compare SELF with a larger model from the same family.

B Implementation Details

This section specifies the training hyperparameters, dataset partitions, and evaluation protocols used for SELF. Benchmark and model descriptions are provided in Appendix A, and the training objectives are derived in Appendix D.3.

B.1 Hyperparameters

Optimization Settings. We train Qwen3-4B-Instruct and Qwen3-8B using LoRA (Hu et al., 2022) with AdamW (Loshchilov & Hutter, 2019), a learning rate of 3×10^{-4}, and a training batch size of 16. Qwen3-8B uses its non-thinking configuration. The feedback-modeling and hindsight-distillation weights are $\lambda_o = 0.2$ and $\lambda_a = 1.0$, respectively. Both objectives update the same student adapters. Table 6 summarizes the default configuration, including the backbone-specific LoRA settings.

Interaction Collection and Optimization. Agents use a ReAct-style interface (Yao et al., 2023), selecting actions from the task instruction and accumulated interaction history. At the start of each training round, the current student collects trajectories containing transitions (h_i, a_i, f_i), where h_i is the history preceding an action, a_i is the executed action, and f_i is the resulting environmental feedback. The collected trajectories remain fixed during optimization within that round. The updated student then collects the next round of interactions. Optimization samples a trajectory uniformly and then a turn uniformly from that trajectory. Within each transition, feedback-prediction and action-distillation losses are normalized separately by their target lengths. This corresponds to averaging over trajectories, turns, and target tokens, rather than giving longer trajectories greater outer weight. Recorded actions and feedback are treated as fixed data; gradients do not propagate through sampling or environment execution.

Training Targets. Feedback prediction conditions on the history and executed action, and uses the realized environmental response as its target. During training, the response is scored through teacher forcing with token-averaged negative log-likelihood. For hindsight distillation, the student and teacher score the same recorded action prefixes. The student receives the available interaction history, while the teacher additionally receives the realized feedback. Their token distributions are aligned by position within the action, accounting for the different conditioning-context lengths. Distillation uses reverse KL from the student distribution to the detached teacher distribution over the full vocabulary (Gu et al., 2024). Feedback and action losses are evaluated only at their respective target positions; conditioning and padding tokens are excluded. At deployment, action generation uses only the student policy, without an additional feedback-prediction or teacher-scoring step.

B.2 Dataset Setup

Training and Test Partitions. We use the official training and test partitions of τ-bench Retail and AppWorld. Retail is evaluated on its official test split, and AppWorld is evaluated on `test-normal`. Training trajectories are collected through interactions with training tasks, rather than provided as expert solutions. Held-out test interactions do not contribute to parameter updates.

Supervision Available During Learning. SELF learns from environmental observations, including ordinary execution outputs, error messages, and status information. It does not add separate task-success labels to either training objective. In the comparison settings, Observation supplies environ-

Table 6: Default SELF training configuration. Backbone-specific LoRA settings are listed separately; other settings are shared.

Hyperparameter	**Qwen3-4B-Instruct**	**Qwen3-8B**
Optimizer	AdamW	
Learning rate	3×10^{-4}	
Training batch size	16	
LoRA rank r	16	96
LoRA alpha	32	128
LoRA dropout	0.05	0.05
Gradient checkpointing	Enabled	
Feedback-modeling weight λ_o	0.2	
Hindsight-distillation weight λ_a	1.0	
Interaction interface	ReAct-style	
Collection policy	Current student at the start of each round	
Within-round data	Fixed trajectory collection $\mathcal{T}_t$	
Action-prefix alignment	Same recorded prefixes for student and teacher	
Teacher gradients	Stopped during distillation	
Feedback-prediction loss	Token-averaged negative log-likelihood	
Action-distillation loss	Full-vocabulary reverse KL	
Loss averaging order	Trajectories, turns, then target tokens	
Deployment	Student action policy only	

mental responses; Full additionally supplies ground-truth information and explicit success/failure feedback; Reward supplies verifiable task rewards. OEL (Ye et al., 2026) and SELF use Observation, SDPO Hübotter et al. (2026) and Mem0 (Chhikara et al., 2025) are evaluated under both Observation and Full, and GRPO (Shao et al., 2024) uses Reward. Task-success annotations remain available to the evaluator for scoring held-out performance.

Feedback-Prediction Evaluation Data. We construct a fixed transition collection for each interactive benchmark using the initial Qwen3-4B-Instruct model. This model runs over the benchmark's evaluation tasks three times, producing recorded histories, actions, and actual environmental responses. The resulting transitions are shared across the feedback predictors being compared. These three collection passes define the diagnostic evaluation data; they are distinct from the four attempts per task used to calculate Retail $\mathrm{pass@4}$.

B.3 Evaluation Protocols

Interactive Task Performance. The deployable student is evaluated on held-out tasks using only the information available before each action. Task completion is determined by the benchmark evaluator. Retail reports success rate (SR), calculated as $\mathrm{pass@4}$, and average interaction turns (Turns). AppWorld reports its official Task Goal Completion (TGC) and Scenario Goal Completion (SGC) metrics (Trivedi et al., 2024). SR, TGC, and SGC are expressed as percentages.

Retail Success Rate and Interaction Length. For each of the N test tasks, we conduct four independent attempts. Let $s_{q,r} \in \{0, 1\}$ indicate whether attempt r on task q succeeds. A task contributes to $\mathrm{pass@4}$ if at least one attempt succeeds:

$$\mathrm{SR} = \mathrm{pass@4} = \frac{100}{N} \sum_{q=1}^{N} \mathbf{1}\left[\sum_{r=1}^{4} s_{q,r} \geq 1\right]. \tag{9}$$

Thus, each task receives equal weight, regardless of how many of its attempts succeed. This measures success within four attempts, rather than the proportion of tasks successful in all four attempts.

Let $L_{q,r}$ be the interaction length of attempt r on task q. We average over all attempts, including failures:

$$\mathrm{Turns} = \frac{1}{4N} \sum_{q=1}^{N} \sum_{r=1}^{4} L_{q,r}. \tag{10}$$

Turns is interpreted together with SR, since a short failed interaction does not indicate efficient task completion.

Feedback-Prediction Accuracy. Let $\mathcal{E}_b$ denote the fixed transition collection for benchmark b obtained from the three collection passes above. For each recorded transition (h_i, a_i, f_i), the evaluated model generates a feedback prediction $\hat{f}_i$ conditioned on h_i and a_i. The realized response f_i is withheld from the predictor and used only as the reference for scoring. This evaluation measures generated feedback, rather than token likelihood under teacher forcing.

GPT-5.6-Luna, with reasoning effort set to `low`, compares $\hat{f}_i$ with f_i and returns a binary judgment:

$$c_i = J(\hat{f}_i, f_i) \in \{0, 1\}, \ \text{Acc.}_b = \frac{100}{|\mathcal{E}_b|} \sum_{i \in \mathcal{E}_b} c_i. \tag{11}$$

Here, $c_i = 1$ means that the judge considers the predicted feedback consistent with the actual feedback, and $c_i = 0$ means that it does not. The judge receives only the two feedback texts, without the interaction history or action. Consequently, Acc. measures agreement with observed feedback, not whether the action itself is appropriate or completes the task. All recorded steps receive equal weight, so longer trajectories contribute more evaluation examples. The same transition collection is used for every predictor to separate prediction quality from differences in the trajectories generated by their own policies.

Privileged-Teacher Task Evaluation. The privileged teacher is evaluated on complete tasks, using the same task-success criteria as the student. At each decision step, we save the current environment state. The student generates a trial action and executes it to obtain environmental feedback. The teacher then receives the student's pre-action context augmented with that feedback. The student's trial action is not included in the teacher prompt; the feedback is its only additional information relative to the student's context. The environment is then restored to the saved pre-action state, and the teacher generates and executes its action from that state. The resulting teacher-action transition is added to the ongoing interaction history and determines the state for the next step. The student trial is used only to obtain the additional feedback; its state changes are rolled back. This process is repeated until the task terminates, so task success is evaluated on the trajectory executed by the teacher rather than by editing a completed student trajectory.

Teacher task scores assess performance with privileged feedback unavailable to the student at its original decision. They are reported separately from deployable student scores and do not represent ordinary inference without access to additional environmental information.

General Capability Retention. We evaluate the initial Qwen3-4B-Instruct model, the adapted student, and the teacher checkpoint on MMLU, MMLU-Pro, and IFEval. Each checkpoint receives the benchmark inputs without privileged environmental feedback. These evaluations assess knowledge, reasoning, and instruction following after agent training and are excluded from the training objectives. The teacher's general capability scores are therefore distinct from its privileged interactive-task performance.

C Additional Results

C.1 Effect of Cross-Rollout Feedback Modeling

Experimental Design. We examine whether the benefit of environmental feedback modeling depends on being coupled to the same interaction experience used for hindsight self-distillation. In standard SELF, the two objectives are computed from the same recorded transitions: the realized feedback of a transition is used both as privileged context for hindsight self-distillation and as the prediction target for environmental feedback modeling. We compare this setting with a cross-rollout variant, in which the two objectives are trained on transitions from different rollouts.

Let

$$\mathcal{B} = \{(h_i, a_i, f_i)\}_{i=1}^{B}$$

Table 7: Effect of coupling environmental feedback modeling and hindsight self-distillation on the same rollout with Qwen3-4B-Instruct. In the cross-rollout condition, feedback modeling uses intact transitions from independently collected rollouts, while hindsight self-distillation uses the original rollout. Results are from one run per condition. SR, TGC, and SGC are percentages, with higher values indicating better performance. Turns is interpreted alongside SR. Bold values indicate the best completion scores.

Condition	Retail		AppWorld	
	SR ↑	Turns ↓	TGC ↑	SGC ↑
Base	27.1	10.7	17.00	5.56
GRPO	33.0	13.4	28.67	10.50
SELF (cross-rollout)	36.6	11.7	31.21	8.67
SELF (same-rollout)	**38.6**	**10.3**	**32.16**	**12.44**

denote a batch of transitions used for hindsight self-distillation. Standard SELF also computes environmental feedback modeling on the same batch,

$$\mathcal{L}_{\mathrm{SELF}} = \frac{\lambda_a}{B}\sum_{i=1}^{B}\ell_{\mathrm{sd}}(h_i, a_i, f_i) + \frac{\lambda_o}{B}\sum_{i=1}^{B}\ell_{\mathrm{env}}(h_i, a_i, f_i). \tag{12}$$

For the cross-rollout variant, we independently sample another batch

$$\mathcal{B}' = \{(h'_i, a'_i, f'_i)\}_{i=1}^{B}$$

from different collected rollouts and optimize

$$\mathcal{L}_{\mathrm{cross}} = \frac{\lambda_a}{B}\sum_{i=1}^{B}\ell_{\mathrm{sd}}(h_i, a_i, f_i) + \frac{\lambda_o}{B}\sum_{i=1}^{B}\ell_{\mathrm{env}}(h'_i, a'_i, f'_i). \tag{13}$$

Importantly, each transition in $\mathcal{B}'$ preserves its original history, action, and realized feedback. Thus, the feedback-prediction task remains well defined: $(h'_i, a'_i) \rightarrow f'_i$ is still an actual environment transition. The intervention only removes the requirement that feedback modeling and hindsight self-distillation operate on the same interaction experience.

This comparison allows us to distinguish the general benefit of learning environmental feedback from the additional benefit of coupling feedback modeling and hindsight supervision on the same transitions. All other training settings are kept fixed. Each condition is evaluated in a single run. We report Retail success rate (SR; $\mathrm{pass@4}$) and average interaction turns, together with AppWorld Task Goal Completion (TGC) and Scenario Goal Completion (SGC).

Results and Analysis. Table 7 shows that environmental feedback modeling remains effective even when its training transitions are decoupled from those used for hindsight self-distillation. The cross-rollout variant achieves 36.6% SR on Retail and 31.21% TGC on AppWorld, substantially outperforming the unadapted backbone and remaining competitive with the full SELF objective. This result suggests that the benefit of feedback modeling does not arise solely from using the same transition for both objectives: learning to predict environmental responses from valid interaction experience can improve the shared model even when that experience comes from a different rollout.

Using the same rollout for both objectives provides an additional benefit. Compared with cross-rollout training, standard SELF improves Retail SR from 36.6% to 38.6% while reducing average interaction length from 11.7 to 10.3 turns. On AppWorld, TGC increases from 31.21% to 32.16%, while SGC increases from 8.67% to 12.44%. These results indicate that environmental feedback modeling provides a broadly useful learning signal on its own, while coupling it with hindsight self-distillation on the same interaction experience yields further gains.

C.2 Training Cost

Measurement Setup. We compare SELF with OEL (Ye et al., 2026), SDPO (Hübotter et al., 2026) under the Observation setting, and GRPO (Shao et al., 2024) using four NVIDIA A100 GPUs 80 GB.

Table 8: Training costs for the main experiments in Table 1 using four NVIDIA A100 80 GB GPUs. Values are aggregated across the reported training runs and exclude final evaluation and additional ablations. Lower values indicate lower cost.

Method	GPU-hours	Wall time(h)
OEL	20.3	6.12
SDPO (Observation)	11.8	3.67
GRPO	41.2	12.46
SELF	16.4	4.93

For each method, we aggregate the training costs of its runs in Table 1 across the reported backbones and interactive benchmarks. The accounting includes interaction collection and optimization, but excludes final evaluation and additional ablation experiments. Table 8 reports local GPU-hours and wall time. GPU-hours sum the device time used across runs, while wall time sums the elapsed duration of the individual training runs.

Results and Analysis. SELF requires 16.4 GPU-hours and 4.93 hours of wall time. Compared with GRPO, it reduces GPU usage by 60.2% and wall time by 60.4%. It also uses 19.2% fewer GPU-hours and 19.4% less wall time than OEL. Relative to SDPO (Observation), SELF incurs an additional 4.6 GPU-hours and 1.26 hours, corresponding to increases of 39.0% and 34.3%, respectively. SELF therefore has a higher training cost than SDPO (Observation), while remaining less expensive than OEL and GRPO in these experiments. Its feedback-prediction and hindsight-distillation objectives reuse collected transitions without requiring additional environment execution to construct their targets. The additional computation is confined to training: deployment uses only the student action policy.

D SELF DETAILS

This section presents the complete SELF training procedure, then explains the construction of hindsight supervision and the two learning objectives. We distinguish the data held fixed within a collection round from the model quantities recomputed at each optimization step, and specify how losses are aligned, normalized, and differentiated.

D.1 TRAINING WORKFLOW

Interaction Collection and Recorded Transitions. SELF alternates between collecting interactions and optimizing the student. At the start of round t, the current student interacts with the training environment through its deployable action policy. A recorded transition (h_i, a_i, f_i) contains the task instruction and pre-action history h_i, the executed action a_i, and the resulting environmental feedback f_i. Earlier observations are included in h_i; the response to a_i becomes available only after that action is executed.

The resulting trajectory collection $\mathcal{T}_t$ remains fixed throughout the round's optimization steps. SELF does not resample actions or rerun the environment to construct targets from these transitions. Instead, feedback prediction uses the recorded response as a target, and hindsight distillation uses that response as additional teacher context. After optimization, the updated student collects the next round.

Student and Teacher Roles. The student uses shared trainable parameters θ for action generation and feedback prediction, with separate instructions for the two uses. In the LoRA implementation, both objectives update the same student adapters while the backbone weights remain frozen. The teacher, denoted by parameters $\bar{\theta}$, is initialized from the student and maintained through EMA updates. Teacher outputs are detached during distillation. Algorithm 1 treats teacher maintenance as a separate step rather than a gradient-based update through the teacher.

Sampling and Optimization. Each minibatch contains 16 transitions. To give trajectories equal outer weight, each transition is sampled by first selecting a trajectory uniformly from $\mathcal{T}_t$ and then

Algorithm 1 SELF: training with environmental feedback and hindsight distillation

Require: Initial student parameters θ_0; training environment; rounds T; updates per round K; batch size $B = 16$
Require: Loss weights $\lambda_o = 0.2$, $\lambda_a = 1.0$; learning rate $\eta = 3 \times 10^{-4}$; student, teacher, and feedback prompts
Initialize student $\theta \leftarrow \theta_0$ and teacher $\bar{\theta} \leftarrow \theta_0$
Initialize the AdamW optimizer for the student trainable parameters
for $t = 0, \ldots, T-1$ **do**
  Collect $\mathcal{T}_t$ using the current student action policy
  Store each transition (h_i, a_i, f_i); hold $\mathcal{T}_t$ fixed in this round
  for $s = 1, \ldots, K$ **do**
    Form $\mathcal{B}$ by sampling B times: a uniform trajectory, then a uniform turn
    for all $(h_i, a_i, f_i) \in \mathcal{B}$ **do**
      Compute student action distributions $p^{\mathrm{stu}}_{i,j}$ at recorded prefixes
      Compute detached teacher distributions $p^{\mathrm{tea}}_{i,j}$ using $h_i, f_i, a_{i,<j}$
      Align student and teacher outputs by action-token position j
      $\ell^{(i)}_{\mathrm{sd}} \leftarrow \frac{1}{L_i} \sum_{j=1}^{L_i} D_{\mathrm{KL}}\left(p^{\mathrm{stu}}_{i,j} \,\|\, \mathrm{sg}[p^{\mathrm{tea}}_{i,j}]\right)$
      Compute $p^{\mathrm{env}}_{i,k}$ using $h_i, a_i, f_{i,<k}$
      $\ell^{(i)}_{\mathrm{env}} \leftarrow -\frac{1}{M_i} \sum_{k=1}^{M_i} \log p^{\mathrm{env}}_{i,k}(f_{i,k})$
    end for
    $\widehat{\mathcal{L}} \leftarrow \frac{1}{B} \sum_{i \in \mathcal{B}} \left(\lambda_o \ell^{(i)}_{\mathrm{env}} + \lambda_a \ell^{(i)}_{\mathrm{sd}}\right)$
    Clear student gradients and backpropagate $\widehat{\mathcal{L}}$
    Update student trainable parameters with AdamW at learning rate η
    Update the teacher using the maintained EMA rule, without backpropagation
  end for
end for
return Student action policy conditioned on the task and available history

selecting a turn uniformly from that trajectory. Both losses are computed for each sampled transition and normalized by their respective target lengths before averaging across the minibatch. The resulting estimator matches the objective in Equation 28.

Algorithm 1 uses K optimization steps per collection round. AdamW updates only the student parameters; teacher outputs, recorded prefixes, and environmental feedback are treated as constants for that update. The teacher distributions are recomputed using the current teacher checkpoint at subsequent steps, even though the underlying transitions remain unchanged. Default optimizer and LoRA settings are given in appendix.

D.2 Hindsight Self-Distillation

Constructing Hindsight Supervision. Consider a collected transition (h_i, a_i, f_i), where h_i contains the task instruction and the interaction history before action a_i, and f_i is the feedback returned after its execution. The feedback records the realized consequence of the action, but does not directly specify a corrected action or a target policy. SELF constructs such supervision through a hindsight teacher that evaluates the recorded action with access to both the original task context and its observed consequence.

Let $a_i = (a_{i,1}, \ldots, a_{i,L_i})$. At action position j, the student and teacher distributions are

$$p^{\mathrm{stu}}_{i,j}(v) = \pi_\theta(v \mid h_i, a_{i,<j}, \mathrm{prompt}_{\mathrm{stu}}), \tag{14}$$

$$p^{\mathrm{tea}}_{i,j}(v) = \pi_{\bar{\theta}}(v \mid h_i, f_i, a_{i,<j}, \mathrm{prompt}_{\mathrm{tea}}), \tag{15}$$

for vocabulary token $v \in \mathcal{V}$. The teacher parameters $\bar{\theta}$ are maintained as an exponential moving average of the student. The task instruction remains available through h_i, so hindsight reassessment is conditioned on what the agent was asked to accomplish, not merely on the contents of the feedback.

The teacher and student evaluate the same recorded action prefix $a_{i,<j}$. The complete action is not provided as additional teacher context; at position j, only its prefix is used to define the next-token prediction being compared. The teacher additionally receives f_i, while the task and pre-action history remain available through h_i. This places both distributions at the same action-token decision, with hindsight feedback as the teacher's privileged information. The target is a learned distribution, not an externally supplied corrected action.

Because feedback adds tokens to the teacher's input, the student and teacher logits must be matched by position within the recorded action, not by absolute position in their full input sequences. Conditioning and padding positions are excluded from the comparison. In the formulas below, L_i and M_i denote the numbers of scored action and feedback tokens, respectively, and are assumed to be positive.

Action-Prefix Alignment and Distillation. SELF transfers hindsight supervision through the length-normalized reverse KL

$$\ell_{\mathrm{sd}}(h_i, a_i, f_i) = \frac{1}{L_i} \sum_{j=1}^{L_i} D_{\mathrm{KL}}\big(p_{i,j}^{\mathrm{stu}} \,\big\|\, \mathrm{sg}[p_{i,j}^{\mathrm{tea}}]\big)\,, \tag{16}$$

where sg denotes stop-gradient. The comparison uses distributions over candidate next tokens, rather than treating the recorded token $a_{i,j}$ as a correct action label. Consequently, the sampled action determines the prefixes at which learning occurs without requiring that the sampled action itself be successful.

For a fixed position, abbreviate $p = p_{i,j}^{\mathrm{stu}}$ and $q = \mathrm{sg}[p_{i,j}^{\mathrm{tea}}]$. The local objective is

$$D_{\mathrm{KL}}(p\|q) = \sum_{v \in \mathcal{V}} p(v)\big(\log p(v) - \log q(v)\big). \tag{17}$$

This objective allows the teacher to redistribute support across the vocabulary. Unlike hard-label imitation, it need not reinforce the recorded next token; the update depends on the teacher's distribution relative to the student's current prediction. The exact form of this update is derived below.

Gradient Boundaries and Deployment. The observation f_i enters the teacher context during training, but not the deployable student's context for action a_i. Distillation therefore transfers information through parameter updates rather than through access to future observations at inference time. Earlier observations remain available in h_i, as in the original interaction process.

Within each gradient update, the recorded transition and detached teacher prediction are fixed supervision. Gradients do not pass through the teacher, action sampling, or environment execution. When trajectories are reused across optimization steps, the recorded prefixes remain fixed even as the student parameters change. Equation 16 therefore defines distillation at behavior-policy prefixes, with the hindsight context supplied by each recorded transition.

This training computation differs from the privileged-teacher full-task evaluation in appendix. Training scores recorded prefixes without executing teacher actions. In that diagnostic evaluation, a student trial provides feedback, the environment is restored to its pre-action state, and the teacher generates and executes its own action. Ordinary deployment uses neither this trial-and-restore procedure nor future feedback.

D.3 SELF Objective

We derive the feedback-prediction and hindsight-distillation gradients, then express their contributions to the shared student under the trajectory–turn–token weighting.

Feedback Prediction as Forward KL. Hindsight supervision depends on the teacher's ability to interpret the feedback it receives. SELF complements this supervision with a predictive objective that directly trains the shared model on action–consequence relationships.

Write $f_i = (f_{i,1}, \ldots, f_{i,M_i})$ and define

$$p_{i,k}^{\mathrm{env}}(v) = p_{\mathrm{env},\theta}(v \mid h_i, a_i, f_{i,<k}, \mathrm{prompt}_{\mathrm{env}})\,. \tag{18}$$

These token distributions define the autoregressive feedback model

$$p_{\mathrm{env},\theta}(f_i \mid h_i, a_i, \mathrm{prompt}_{\mathrm{env}}) = \prod_{k=1}^{M_i} p_{i,k}^{\mathrm{env}}(f_{i,k}). \tag{19}$$

Let $\delta_{\mathrm{env},i,k}$ be a point mass on the observed token $f_{i,k}$. This is an empirical token target from a recorded response, rather than the full distribution of possible environmental responses. Its entropy is zero, giving

$$\begin{aligned} D_{\mathrm{KL}}\big(\delta_{\mathrm{env},i,k} \,\big\|\, p_{i,k}^{\mathrm{env}}\big) &= \sum_{v\in\mathcal{V}} \delta_{\mathrm{env},i,k}(v) \log \frac{\delta_{\mathrm{env},i,k}(v)}{p_{i,k}^{\mathrm{env}}(v)} \\ &= -\log p_{i,k}^{\mathrm{env}}(f_{i,k}). \end{aligned} \tag{20}$$

Here, zero-probability terms use the standard convention $0 \log 0 = 0$. Averaging over the feedback tokens yields

$$\begin{aligned} \ell_{\mathrm{env}}(h_i, a_i, f_i) &= \frac{1}{M_i} \sum_{k=1}^{M_i} D_{\mathrm{KL}}\big(\delta_{\mathrm{env},i,k} \,\big\|\, p_{i,k}^{\mathrm{env}}\big) \\ &= -\frac{1}{M_i} \sum_{k=1}^{M_i} \log p_{i,k}^{\mathrm{env}}(f_{i,k}) \\ &= -\frac{1}{M_i} \log p_{\mathrm{env},\theta}(f_i \mid h_i, a_i, \mathrm{prompt}_{\mathrm{env}}). \end{aligned} \tag{21}$$

Thus, forward-KL feedback modeling is exactly length-normalized maximum-likelihood learning from the realized environment response.

For a feedback-prediction logit $u_{i,k,v}$, softmax differentiation gives

$$\frac{\partial \ell_{\mathrm{env}}}{\partial u_{i,k,v}} = \frac{1}{M_i} \left(p_{i,k}^{\mathrm{env}}(v) - \mathbf{1}\{v = f_{i,k}\}\right). \tag{22}$$

For the response recorded in a transition, the target is available regardless of whether the action ultimately contributes to task success. For example, an execution error remains a valid observation to predict. The objective learns what follows an action, rather than labeling the action as desirable. This distinction explains why feedback modeling complements, rather than replaces, policy learning.

Deriving the Reverse-KL Gradient. At a fixed action position, write $p = p_{i,j}^{\mathrm{stu}}$ and $q = \mathrm{sg}[p_{i,j}^{\mathrm{tea}}]$. Let z_v denote the student logit, with $p(v) = \exp(z_v)/\sum_u \exp(z_u)$. Assuming positive probabilities, define

$$A(v) = \log q(v) - \log p(v), \qquad \bar{A} = \sum_{u\in\mathcal{V}} p(u) A(u). \tag{23}$$

The quantity $A(v)$ measures relative teacher support, and $\bar{A} = -D_{\mathrm{KL}}(p\|q)$. Since q is detached,

$$\frac{\partial D_{\mathrm{KL}}(p\|q)}{\partial p(u)} = \log p(u) - \log q(u) + 1. \tag{24}$$

Combining this expression with the softmax Jacobian,

$$\frac{\partial p(u)}{\partial z_v} = p(u)\big(\mathbf{1}\{u = v\} - p(v)\big), \tag{25}$$

gives

$$\begin{aligned} \frac{\partial D_{\mathrm{KL}}(p\|q)}{\partial z_v} &= \sum_{u\in\mathcal{V}} p(u)\big(\mathbf{1}\{u = v\} - p(v)\big)\big(\log p(u) - \log q(u) + 1\big) \\ &= p(v)\left[\log p(v) - \log q(v) - \sum_{u\in\mathcal{V}} p(u)\big(\log p(u) - \log q(u)\big)\right] \\ &= -p(v)\big(A(v) - \bar{A}\big). \end{aligned} \tag{26}$$

The constant terms cancel because $\sum_u p(u) = 1$. For the transition-level distillation loss, the corresponding derivative is

$$\frac{\partial \ell_{\mathrm{sd}}}{\partial z_{i,j,v}} = -\frac{1}{L_i} p^{\mathrm{stu}}_{i,j}(v) \left(A_{i,j}(v) - \bar{A}_{i,j} \right). \tag{27}$$

If logits were optimized independently, gradient descent would increase a token's logit when its relative teacher support exceeds the student-weighted average, and decrease it when that support is below average. In the actual model, these logit derivatives propagate through shared trainable parameters, so their effects across tokens and examples are coupled. The centering arises from probability normalization, not from a reward-based advantage estimator. Although A resembles an advantage notation, it is computed entirely from the student and teacher distributions.

This derivation explains how an observation can produce dense policy supervision without specifying a reference action. The usefulness of the correction depends on whether the feedback-conditioned teacher provides informative relative support for candidate tokens. The gradient expression characterizes how that support is transferred; it does not establish that every teacher reassessment improves task performance.

Joint Objective and Effective Sample Weights. For a fixed collection $\mathcal{T}_t$, let n_τ be the number of turns in trajectory τ. Transition indices below are local to each trajectory; the trajectory subscript on (h_i, a_i, f_i) is omitted for readability. SELF combines the two transition-level objectives as follows:

$$\mathcal{L}_{\mathrm{SELF},t} = \frac{1}{|\mathcal{T}_t|} \sum_{\tau \in \mathcal{T}_t} \frac{1}{n_\tau} \sum_{i=1}^{n_\tau} \left[\lambda_o \ell_{\mathrm{env}}(h_i, a_i, f_i) + \lambda_a \ell_{\mathrm{sd}}(h_i, a_i, f_i) \right]. \tag{28}$$

For nonempty trajectories and scored sequences, the effective coefficients on individual feedback and action token losses are, respectively,

$$w^{\mathrm{env}}_{\tau,i,k} = \frac{\lambda_o}{|\mathcal{T}_t|\, n_\tau M_i}, \qquad w^{\mathrm{sd}}_{\tau,i,j} = \frac{\lambda_a}{|\mathcal{T}_t|\, n_\tau L_i}. \tag{29}$$

Each trajectory receives equal outer weight, turns are averaged within their trajectory, and each objective is normalized by its own target length. Long feedback sequences therefore do not receive greater total weight merely because they contain more tokens. The coefficients λ_o and λ_a control the explicit weighting of the two normalized objectives, although their gradient magnitudes also depend on the model predictions.

This formulation corresponds to the trajectory-then-turn sampling procedure in Algorithm 1. Uniformly sampling a flattened pool of transitions would instead favor trajectories containing more turns, unless equivalent weighting corrections were applied. Similarly, a single average over all tokens in a batch generally differs from averaging the length-normalized transition losses.

Shared-Parameter Learning across Updates. Let

$$\begin{aligned} g_{\mathrm{env}} &= \frac{1}{|\mathcal{T}_t|} \sum_{\tau \in \mathcal{T}_t} \frac{1}{n_\tau} \sum_{i=1}^{n_\tau} \nabla_\theta \ell^{(i)}_{\mathrm{env}}, \\ g_{\mathrm{sd}} &= \frac{1}{|\mathcal{T}_t|} \sum_{\tau \in \mathcal{T}_t} \frac{1}{n_\tau} \sum_{i=1}^{n_\tau} \nabla_\theta \ell^{(i)}_{\mathrm{sd}}. \end{aligned} \tag{30}$$

With the collected data and teacher outputs fixed within the update, linearity gives

$$\nabla_\theta \mathcal{L}_{\mathrm{SELF},t} = \lambda_o g_{\mathrm{env}} + \lambda_a g_{\mathrm{sd}}. \tag{31}$$

The two gradients act on the same student parameters, but provide different supervision: g_{env} learns the observed consequence of the recorded action, whereas g_{sd} transfers the teacher's hindsight action distribution into the deployable policy.

Their interaction also extends beyond the current optimization step. Feedback-modeling updates alter the student and, through subsequent EMA updates (Tarvainen & Valpola, 2017), the teacher used for future hindsight supervision. Policy distillation also changes these shared parameters and can therefore affect feedback prediction, although the direction of that effect is an empirical question.

It can additionally change the actions and observations collected in later rounds. Because the teacher is detached, this interaction is not a same-step differentiable cycle.

This decomposition describes how the two objectives update the shared model, while the coefficient studies provide empirical evidence for their complementary benefits. These benefits are not guaranteed by gradient addition alone; the component gradients may reinforce or oppose one another at different optimization steps.

E Illustrative Prompts and Examples

[q] Task instruction

```
Fix the Python program analyze_sales.py.  The script should read
data/orders.csv, consider only orders whose status is completed,
compute the average revenue of those orders, and print the result
with exactly two decimal places.
You may inspect files, execute Python or shell commands, and edit
the implementation.  Do not modify the tests.
```

[o_0] Initial environment observation

```
Working directory:  /workspace/sales_analysis
Repository contents:
 analyze_sales.py
data/orders.csv
tests/test_analyze_sales.py
README.md
```

[a_1] Agent action

```
sed -n '1,220p' analyze_sales.py
```

[f_1] Environmental feedback

```
 import csv

def main():
  total = 0.0
  count = 0
  with open("data/orders.csv") as f:
    reader = csv.DictReader(f)
    for row in reader:
      total += float(row["revenue"])
      count += 1
  print(f"Average revenue:  {total/count:.1f}")

if __name__ == "__main__":
  main()
```

[a_2] Agent action

```
head -5 data/orders.csv && pytest -q
```

[f_2] Environmental feedback

```
 order_id,status,total
1001,completed,32.50
1002,cancelled,80.00
1003,completed,62.00
1004,pending,44.00
 F
============================ FAILURES ============================
KeyError:  'revenue'
analyze_sales.py:10
```

[a_3] Agent action

```
sed -n '1,220p' tests/test_analyze_sales.py
```

[f_3] Environmental feedback

```
 def test_completed_orders_only(capsys):
  main()
```

```
    out = capsys.readouterr().out.strip()
    assert out == "Average completed-order revenue:  47.25"
```

[a_4] Agent action

```
Inspect all rows to verify which records should contribute:
cat data/orders.csv

...
```

[f_6] Environmental feedback

```
 F
============================== FAILURES ==============================
Expected:  Average completed-order revenue:  47.25
Received:  Average completed-order revenue:  47.2
1 failed in 0.08s
```

[a_7] Agent action

```
 python - <<'PY'
from pathlib import Path
p = Path("analyze_sales.py")
s = p.read_text().replace(':.1f', ':.2f')
p.write_text(s)
PY
```

[f_7] Environmental feedback

```
Command completed successfully.
```

[a_8] Agent action

```
pytest -q
```

[f_8] Environmental feedback

```
 .
1 passed in 0.07s
```

[a_9] Agent action

```
python analyze_sales.py
```

[f_9] Environmental feedback

```
 Average completed-order revenue:  47.25
```

[a_{10}] Agent action

```
git diff -- analyze_sales.py
```

[f_{10}] Environmental feedback

```
 - total += float(row["revenue"])
- count += 1
+ if row["status"] != "completed":
+  continue
+ total += float(row["total"])
+ count += 1
- print(f"Average revenue:  {total/count:.1f}")
+ print(f"Average completed-order revenue:  {total/count:.2f}")
```

[a_n] Final agent response

```
Implemented the fix in analyze_sales.py.  The program now filters
for completed orders, uses the total field, and prints the average
with two decimal places.  The test suite passes successfully.
```

Example 1 --- Python execution **Correct prediction**

Task q: Debug a Python script that parses a JSONL dataset.

History h_i: ... The agent inspected the schema and found that each record contains id, text, and label.

Current action a_i: python validate_dataset.py data/train.jsonl

Realized feedback f_i: KeyError: 'user_id' at validate_dataset.py:27

Predicted feedback $\hat{f}_i$: KeyError: 'user_id' at validate_dataset.py:27

Result: The prediction correctly identifies both the exception type and the missing field.

Example 2 --- Retail tool interaction **Incorrect prediction**

Task q: Determine whether the user's wireless-headphone order is eligible for return.

History h_i: ... The user account has been verified and Order 1042 was identified as the relevant purchase.

Current action a_i: get_order_details(order_id="1042")

Realized feedback f_i: Order 1042 was delivered 12 days ago and remains within the return window.

Predicted feedback $\hat{f}_i$: Order 1042 was cancelled before shipment and is eligible for an automatic refund.

Result: The model predicts the wrong order state and therefore misinterprets the consequence of the action.

Example 3 --- SQL database interaction **Incorrect prediction**

Task q: Count the number of pending orders in the database.

History h_i: ... The agent connected successfully to orders.db and verified the table schema.

Current action a_i: SELECT COUNT(*) FROM orders WHERE status='pending';

Realized feedback f_i: 17

Predicted feedback $\hat{f}_i$: 0

Result: The predicted database response is incorrect despite the action itself being syntactically valid.

Example 4 --- Filesystem interaction **Correct prediction**

Task q: Locate the configuration used by the training script.

History h_i: ... The repository contains train.py, configs/, and several experiment directories.

Current action a_i: cat configs/base.yaml

Realized feedback f_i: model: qwen3-4b
batch_size: 16
learning_rate: 1e-6

Predicted feedback $\hat{f}_i$: model: qwen3-4b
batch_size: 16
learning_rate: 1e-6

Result: The predicted environmental feedback matches the observed file contents.

E.1 PROMPTS

Prompt for Student Action Generation

Task: Help the user identify an order that is eligible for return and complete the return according to the store policy.

Interaction history h_i:

. . .

User: I bought a pair of wireless headphones recently and would like to return them. Could you help me check whether the order is still eligible?

Agent: `get_user_details(user_id="U2481")`

Environment: The user account is verified and active.

Agent: `get_order_history(user_id="U2481")`

Environment: Recent orders include Order 1042, containing one pair of wireless headphones, and Order 1031, containing a phone case.

prompt$_{\text{stu}}$: Based on the task and the interaction history available so far, determine the next action that should be taken in the environment. Use the available tools when necessary and generate only the next agent action.

Prompt for Feedback-Conditioned Teacher Action Generation

Task: Help the user identify an order that is eligible for return and complete the return according to the store policy.

Interaction history h_i:

. . .

User: I bought a pair of wireless headphones recently and would like to return them. Could you help me check whether the order is still eligible?

Agent: `get_user_details(user_id="U2481")`

Environment: The user account is verified and active.

Agent: `get_order_history(user_id="U2481")`

Environment: Recent orders include Order 1042, containing one pair of wireless headphones, and Order 1031, containing a phone case.

Environmental feedback f_i: Order 1042 was delivered 12 days ago, contains one pair of wireless headphones, and remains within the allowed return window. The order has not previously been returned or refunded.

prompt$_{\text{tea}}$: Use the task, interaction history, and realized environmental feedback as privileged hindsight information. Determine the action that should be taken at the current decision point and generate only the corresponding agent action.

Prompt for Environmental Feedback Prediction

Task: Help the user identify an order that is eligible for return and complete the return according to the store policy.

Interaction history h_i:

. . .

User: I bought a pair of wireless headphones recently and would like to return them. Could you help me check whether the order is still eligible?

Agent: `get_user_details(user_id="U2481")`

Environment: The user account is verified and active.

Agent: `get_order_history(user_id="U2481")`

Environment: Recent orders include Order 1042, containing one pair of wireless headphones, and Order 1031, containing a phone case.

Current action a_i: `get_order_details(order_id="1042")`

prompt$_{\text{env}}$: Based on the task, interaction history, and current agent action, predict the environmental feedback that will be returned after the action is executed. Generate only the expected environmental response, including relevant tool outputs, execution results, or observable state changes.

Figure 4: Illustrative prompting interfaces used in SELF. The student prompt $\mathrm{prompt}_{\mathrm{stu}}$ generates the next action a_i from the interaction history h_i; the teacher prompt $\mathrm{prompt}_{\mathrm{tea}}$ additionally conditions on the realized environmental feedback f_i as privileged hindsight information; and the environmental-feedback modeling prompt $\mathrm{prompt}_{\mathrm{env}}$ predicts f_i from h_i and the current action a_i.