# Prospective Hindsight: Self-Calibrating Reinforcement Learning via Prediction–Reality Gaps

**Jiaxin Zhang, Xiangyu Peng, Qinglin Chen, Yu Li, Hiroaki Hayashi, Chien-Sheng Wu***

Salesforce AI Research

## Abstract

Reinforcement learning for long-horizon agents relies on *purely retrospective* training signals: credit is assigned only after observing environmental consequences, leaving the agent's belief at action time invisible to the gradient. We introduce **Prospective Hindsight (PH)**, a self-calibrating training principle that augments any retrospective base method with a signal derived from the gap between the agent's *prospective prediction* (before feedback) and the *retrospective evaluation* (after feedback). This per-rollout *surprise* identifies samples where the agent's self-model is most inaccurate and amplifies their gradient contribution through a stop-gradient surprise-weighted advantage. Since the prospective predictor shares parameters with the policy, the two co-evolve, progressively shifting focus to the agent's remaining blind spots. We connect this principle to a privileged-information gap and show that minimizing the surprise residual provides a descent pathway on the agent's miscalibration rate; calibration thus emerges as a byproduct of optimization rather than from an added objective. On single-turn verifiable tasks and a multi-turn personal-agent task (under GRPO, on-policy distillation, and their combination), PH improves both task performance and calibration, with consistent gains across model scales. Notably, the dominant miscalibration mode shifts structurally between regimes, overconfident failures in single-turn, underconfident successes in multi-turn, yet the same training principle addresses both successfully.

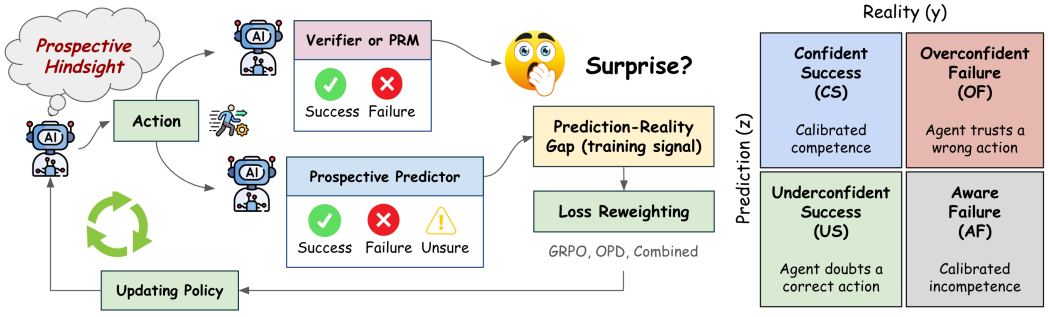



Figure 1: Prospective Hindsight (PH) converts hindsight feedback into a self-calibrating training signal. PH compares an agent's action-time prediction with the realized verifier/PRM outcome, partitions rollouts into calibrated and miscalibrated cases, and upweights surprising prediction–reality mismatches during policy updating within GRPO, on-policy distillation and their combination.

*Correspondence to {jiaxin.zhang, wu.jason}@salesforce.com

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

# 1 Introduction

LLMs increasingly act as *agents*: decomposing tasks, calling tools, writing code, conducting extended tutoring conversations, and making intermediate commitments whose consequences unfold over many steps [42, 3]. In these settings, the most damaging failures are often not simple mistakes, but mistakes made with confidence [52, 46, 55, 51]. A confidently wrong agent continues down a flawed trajectory, ignores opportunities for recovery, and triggers incorrect downstream actions; an agent that fails while expressing uncertainty [47, 28], in contrast, invites a fallback, a retry, or a request for human input. Reliable agentic learning therefore requires more than improving final accuracy — it requires aligning the agent's prospective belief about its own success with the reality of its outcomes.

**The hindsight trap.** Modern agentic RL has converged on increasingly rich *retrospective* feedback [45, 7]. Process reward models score each step by inspecting the next environmental state [42, 3]; on-policy distillation (OPD) methods [25, 37] construct a hindsight teacher conditioned on post-action context unavailable to the agent at decision time [1, 44, 14, 34]. These methods differ in mechanism but share a structural property: every learning signal is computed from quantities observed *after* the agent has acted, and the agent's prospective belief at action time, i.e., what it expected would happen, never enters the gradient. Two rollouts that produced the same outcome therefore receive the same update, regardless of whether one was a confident success and the other a lucky guess, or whether one was a failure the agent could have anticipated and the other a failure that took the agent by surprise. We call this structural obstruction the *hindsight trap*, and it is the reason why retrospective training reliably improves task accuracy without resolving overconfidence [50, 21, 5, 16, 11].

**Calibration as a training signal, not a diagnostic.** A substantial literature on LLM calibration has measured this overconfidence and proposed inference-time fixes: verbalized confidence prompts, semantic uncertainty estimators, post-hoc temperature scaling [12, 15, 39, 18]. Recent work extends these ideas into RL by adding explicit confidence-target losses or rewriting confidence tokens in the teacher distribution [50, 21, 5], but in doing so introduces a separate calibration objective alongside the task objective; the two are reported to trade against each other. We approach the problem from a different angle. The gap between the agent's prospective belief and the eventual outcome is not just a diagnostic of how well-calibrated the agent is; instead, it is a *training signal* that points to exactly the rollouts where action-time information was insufficient.

**Why prediction–reality gaps are informative.** The same property that makes the prospective belief invisible to retrospective losses also makes the gap between belief and outcome informative. By a law-of-total-variance argument, the residual variance in the eventual outcome given action-time information decomposes into aleatoric noise (irreducible even with privileged context) plus a non-negative *privileged-information gap*: the share of outcome variance that no action-time predictor can resolve. This places agent RL inside Learning Under Privileged Information (LUPI) framework [40, 30, 24]. The rollouts that materialize this gap, overconfident failures and underconfident successes, are exactly where action-time information falls short of post-action information. Standard hindsight-only methods see these rollouts only through their outcome label and treat them indistinguishably from calibrated ones; we ask what changes if training explicitly uses the disagreement.

We introduce **Prospective Hindsight (PH)**, a self-calibrating training principle that augments any retrospective base method with a signal derived from the prediction–reality gap. Before each rollout is scored by the verifier or PRM, we re-query the same policy with a different prompt to elicit a *prospective prediction* of the eventual outcome; comparing this prediction to the retrospective evaluation yields a per-rollout *surprise* that identifies the training samples where the agent's self-model is most inaccurate. PH amplifies the gradient contribution of these samples through a stop-gradient reweighting of the base loss, preserving the rollout sampling distribution and the internal structure of GRPO, OPD, or their combination. Because the prospective predictor shares parameters with the policy rather than being an external head bolted onto the model, the predictor co-evolves with the policy as training progresses: the agent's self-model improves implicitly alongside the agent's behavior, the surprise signal progressively shifts focus to the agent's remaining blind spots, and the reweighting diminishes as the prediction–reality gap closes. PH therefore reduces miscalibration as a byproduct of optimization rather than through an added calibration loss, sidestepping the capability–calibration trade-off [21, 5]. Our main contributions are summarized as follows:

- We identify the *hindsight trap* as a structural blind spot of retrospective RL and connect the resulting prediction–reality gap to a privileged-information quantity within the LUPI framework.

- We propose *Prospective Hindsight*, a principle method that converts this gap into a self-calibrating training signal compatible with binary RL (GRPO), On-policy distillation, and their combination.
- We define a calibration taxonomy by pairing the prospective belief with the verifier outcome yields four rollout cells that decompose calibration along two interpretable axes during training.
- PH improves both task performance and calibration on single-turn verifiable tasks and on a multi-turn personal-agent task, with consistent gains across model scales. The dominant miscalibration mode shifts structurally between regimes and PH addresses both successfully.

## 2 Related work

**Retrospective training signals in Agentic RL.** Recent RL/agentic RL frameworks have steadily enriched the post-action signal delivered to the policy [45, 4]. OpenClaw-RL [42] unifies binary RL (GRPO [33], PPO [32]), on-policy distillation (OPD [1, 44, 14, 34, 31, 17]), and their combination under a single asynchronous training stack [54, 6]; Buening et al. [3] further show that raw multi-turn user interactions are themselves sufficient supervision for self-distillation without an explicit reward model. These methods differ in mechanism but share a structural property: every learning signal is computed entirely from quantities observed *after* the agent has acted, so the agent's prospective belief at action time never enters the gradient [1, 44, 41]. Prospective Hindsight is orthogonal to this axis of progress: it leaves the base method's rollout, advantage, and regularization structure intact, and instead converts the gap between an action-time prospective prediction and the eventual retrospective evaluation into a per-rollout reweighting that plugs into any of the above base methods.

**Calibration of language-model agents.** Most prior work treats calibration as a property of the trained model to be measured or controlled at inference time, through diagnostic studies of confidence–accuracy alignment [12, 15, 10], elicitation of verbalized confidence or sampling-based uncertainty estimates [39, 22, 19], trajectory-level uncertainty quantification in agentic settings [18, 8, 53, 23, 13, 47, 51], and recent analyses linking hallucination to training procedures that reward confident guesses over abstention [16]. A separate line moves calibration into RL by adding an explicit confidence-target objective: CaOPD [50] attributes overconfidence in hindsight distillation to teacher–student information asymmetry and proposes confidence-token replacement; Leng et al. [21] and Damani et al. [5] report similar calibration-augmented RL recipes. These approaches consistently report a *capability–calibration trade-off* [43]: improved calibration comes at the cost of a regression in task accuracy. PH proceeds differently: rather than adding a calibration term to compete with the task objective, we use the per-rollout disagreement between the agent's prospective belief and the verifier outcome to *reweight* which base-method gradients deserve more mass. Calibration improves as a byproduct of the same gradient steps that drive task success, and the trade-off observed in calibration-augmented baselines does not arise because calibration and capability share a single descent direction.

## 3 The Hindsight Trap in Agent RL

### 3.1 Agentic Reinforcement Learning

We consider an agent represented by a policy $\pi_\theta$ acting in environments with verifiable outcomes. At each step $t \in \{1, \dots, T\}$, the agent observes a state $s_t$ (a query, a conversation history, or a partial task description), produces an action $a_t \sim \pi_\theta(\cdot \mid s_t)$, and receives a step-level outcome

$$r_t \;=\; R(s_t, a_t, c_t) \;\in\; \{-1, 0, +1\}, \qquad y_t \;=\; \mathbf{1}[r_t > 0] \;\in\; \{0, 1\}, \tag{1}$$

where $c_t$ denotes *privileged context* that the evaluator $R$ observes but the agent does not at action time. This covers single-step tasks ($T{=}1$, with $c_t$ a ground-truth reference) and multi-turn tasks ($T{>}1$, with $c_t{=}s_{t+1}$ the next environmental state). A trajectory is $\tau = \{(s_t, a_t, r_t)\}_{t=1}^{T}$; the base methods of §3.2 differ in how $r_t$ and $c_t$ shape the policy update.

### 3.2 Retrospective Training Methods

Current methods construct the policy update from $r_t$, optionally combined with $c_t$. Scalar-advantage methods such as GRPO [33] normalize the reward within a group of rollouts sharing the same query; token-level distillation methods such as OPD [1, 44, 14] use $c_t$ to construct a hindsight teacher whose

action distribution is distilled into the student; combined methods sum the two signals [42]. Full advantage formulations are given in App. C.1. All three families share the same hindsight-only property: the learning signal is computed entirely from post-action quantities ($r_t$, $c_t$, or both), and the agent's prospective belief at action time is invisible to the update. We abstract any such instantiation as a base loss $\ell^{\text{base}}$ whose precise form does not matter for the analysis below.

### 3.3 The Self-Evaluation Blind Spot

Hindsight-only feedback collapses two distinct questions about each step into a single label. The first is observable from the retrospective evaluator: *what happened*, captured by $r_t$. The second is invisible: *whether the agent could have anticipated what would happen* when it chose $a_t$. Formally, let $z_t \in \{-1, 0, +1\}$ denote the agent's prospective belief about $y_t$, formed from $(s_t, a_t)$ alone before $c_t$ is observed: $z_t = +1$ encodes an expectation of success, $z_t = -1$ an expectation of failure, and $z_t = 0$ an explicit acknowledgement of uncertainty (*honest uncertainty*). The base loss is a function of post-action quantities alone, $\ell^{\text{base}} = \mathcal{F}(\tau, \{c_t\}_{t=1}^{T})$, and therefore independent of any such $z_t$:

$$\mathbb{E}\big[\nabla_\theta \ell^{\text{base}} \,\big|\, z_t, y_t\big] \;=\; \mathbb{E}\big[\nabla_\theta \ell^{\text{base}} \,\big|\, y_t\big] \qquad \forall\, z_t \in \{-1, 0, +1\}, \tag{2}$$

i.e., the gradient distribution depends on the realized outcome $y_t$ alone and is invariant to the agent's prospective belief.

We call the absence of supervision on $z_t$ the *self-evaluation blind spot*: standard updates assign the same gradient to two rollouts that produced the same outcome, regardless of whether the agent acted with calibrated belief, miscalibrated belief, or honest uncertainty. Two consequences follow on the calibrated subset $z_t \in \{-1, +1\}$ (the $z_t{=}0$ case inherits the same invariance and is treated as a neutral class). A confident success and a lucky success receive identical reinforcement, even though the former reflects calibrated competence while the latter is an unrecognized capability the agent cannot deploy with confidence; symmetrically, an overconfident failure is penalized exactly like an aware failure, even though the former is the more dangerous mode — the agent's own model endorses an action the verifier rejects, so the same mistake is likely to recur. Closing this blind spot requires a learning signal tied to the gap between $z_t$ and $y_t$, but only if that gap is structurally meaningful rather than noise that should be averaged out.

### 3.4 Why Prediction–Reality Gaps Are Informative

To characterize the structural quantity that the discrete prospective belief $z_t$ should approximate, consider its Bayes-optimal continuous idealization, $f^\star(s_t, a_t) := \mathbb{E}[y_t \mid s_t, a_t] = \Pr(y_t{=}1 \mid s_t, a_t)$, the minimum-MSE estimator of $y_t$ available without observing $c_t$.

**Proposition 1** (Privileged-information gap)**.** *For any $(s_t, a_t)$ in the support of the rollout distribution,*

$$\operatorname{Var}(y_t \mid s_t, a_t) \;=\; \underbrace{\mathbb{E}[\operatorname{Var}(y_t \mid s_t, a_t, c_t) \mid s_t, a_t]}_{\textit{aleatoric noise}} \;+\; \underbrace{I^\star(s_t, a_t)}_{\textit{privileged-information gap}}, \tag{3}$$

*where* $I^\star(s_t, a_t) := \operatorname{Var}(\mathbb{E}[y_t \mid s_t, a_t, c_t] \mid s_t, a_t) \geq 0$.

The first term is the aleatoric noise irreducible even with $c_t$; the second, $I^\star(s_t, a_t)$, is precisely the contribution that no action-time predictor $f(s_t, a_t)$ can resolve. States with large $I^\star$ identify exactly where the agent's epistemic horizon is most limited and where the prediction–reality gap is informative rather than residual noise. This places our setting in the Learning Under Privileged Information framework [40, 30]: signals available at training time but not at decision time fundamentally constrain the achievable predictor (proof and full LUPI connection in App. B.1). The decomposition motivates a learning rule: maintain a self-evaluator that produces a prospective belief $z_t$ from action-time information alone, and reweight $\ell^{\text{base}}$ as a function of the gap between $z_t$ and $y_t$.

## 4 The Prospective Hindsight Method

Prospective Hindsight (PH) converts this gap into a usable training signal through three components: a self-evaluator that produces a per-rollout belief from action-time information; a four-cell taxonomy that partitions rollouts by belief and outcome; and a one-line reweighting of the base loss driven by a binary surprise indicator. We show that the same rule has a complementary interpretation as implicit calibration. The end-to-end procedure is summarized in Algorithm 1(App. C).

### 4.1 Prospective Self-Evaluator

The agent $\pi_\theta$ produces both the action $a_t$ and a prospective self-evaluation of the same action. The self-evaluator distribution $p_\theta^{\text{eval}}(\cdot \mid s_t, a_t)$ is realized by prompting $\pi_\theta$ with a template that asks whether the eventual outcome of $a_t$ will be a success; the answer is parsed from a `\boxed{·}` field as a ternary score in $\{-1, 0, +1\}$ (success / failure / honest uncertainty; App. C.4). Crucially, the self-evaluator sees only $(s_t, a_t)$ and does not observe the privileged context $c_t$ (full templates in App. F). To reduce variance, we draw $M$ independent samples per step and aggregate by majority vote: $z_t = \text{maj}\left(z_t^{(1)}, \ldots, z_t^{(M)}\right), z_t^{(m)} \overset{\text{i.i.d.}}{\sim} p_\theta^{\text{eval}}(\cdot \mid s_t, a_t)$. The only training-time overhead is $M$ extra forward passes per rollout step.

### 4.2 A Calibration Taxonomy of Rollouts

As shown in Table 1, the pair $(z_t, y_t)$ partitions every rollout with $z_t \in \{-1, +1\}$ into one of four *calibration cells* along two epistemic axes, the outcome axis ($y_t$, observable from the verifier) and the anticipation axis ($z_t$, the agent's prospective self-assessment). Diagonal cells (CS, AF) are calibrated; off-diagonal cells (OF, US) are miscalibrated and capture the asymmetry analyzed in §3.3: OF rollouts are the most consequential failure mode (the agent's own model endorses an action the verifier rejects), while US rollouts identify capability the agent has but cannot yet recognize.

Table 1: Calibration cells of rollouts, indexed by the agent's prospective belief $z_t \in \{-1, +1\}$ and the realized outcome $y_t \in \{0, 1\}$. Rollouts with $z_t{=}0$ are deferred to App. C.4.

| Cell | $z_t$ | $y_t$ | Epistemic interpretation |
|---|---|---|---|
| **CS** (confident success) | $+1$ | 1 | calibrated competence |
| **OF** (overconfident failure) | $+1$ | 0 | miscalibrated; agent trusts a wrong action |
| **US** (underconfident success) | $-1$ | 1 | miscalibrated; agent doubts a correct action |
| **AF** (aware failure) | $-1$ | 0 | calibrated incompetence |

### 4.3 The Prospective Hindsight Update Rule

PH attaches to each rollout a binary *surprise* scalar and amplifies the base loss in proportion.

**Surprise.** For $z_t \in \{-1, +1\}$ and $y_t \in \{0, 1\}$, the per-rollout surprise is the binary indicator of a prediction–reality gap,

$$\xi_t = \mathbb{1}\big[(z_t, y_t) \in \{\text{OF}, \text{US}\}\big] \in \{0, 1\}. \tag{4}$$

$\xi_t{=}1$ exactly when the agent's prospective belief disagrees with the realized outcome (overconfident or underconfident); $\xi_t{=}0$ when they agree (CS or AF). This is the per-rollout realization of the privileged-information gap $I^\star$ from Proposition 1: rollouts with $\xi_t{=}1$ are exactly those whose existence makes $I^\star$ visible from inside the agent's own model.

**Surprise-weighted update.** Let $\ell_t^{\text{base}}(\theta)$ be the per-rollout base loss (§3.2); PH is agnostic to whether the base method is GRPO, OPD, or their combination. The PH-modified loss is

$$\ell_t^{\text{PH}}(\theta) = \big(1 + \alpha \cdot \xi_t\big) \cdot \ell_t^{\text{base}}(\theta), \qquad \nabla_\theta \ell_t^{\text{PH}}(\theta) = \big(1 + \alpha \cdot \xi_t\big) \cdot \nabla_\theta \ell_t^{\text{base}}(\theta), \tag{5}$$

with $\alpha \geq 0$ the only hyperparameter and $\xi_t$ a stop-gradient constant. Calibrated rollouts ($\xi_t{=}0$) keep the base loss unchanged; surprising rollouts ($\xi_t{=}1$) are upweighted by $1{+}\alpha$. Setting $\alpha{=}0$ recovers the base method exactly; we use $\alpha{=}1$ as the default.

### 4.4 PH as Calibration through Reweighting

We also develop a complementary interpretation of the same rule as an implicit calibration mechanism. As the prospective evaluator $p_\theta^{\text{eval}}$ shares parameters with the policy $\pi_\theta$, every PH update changes both the action distribution and the agent's prospective belief; PH biases this joint update toward states in which the two agree. We frame the analysis below as *motivation* for why PH would be expected to reduce miscalibration as a side effect of optimization, rather than as a formal guarantee.

**Surprise-weighted decomposition.** The PH expected loss decomposes additively into the base objective plus a non-negative surprise residual:

$$\mathbb{E}_{\rho_\theta}\big[\ell_t^{\mathrm{PH}}\big] \;=\; \mathbb{E}_{\rho_\theta}\big[\ell_t^{\mathrm{base}}\big] \;+\; \alpha \cdot \mathcal{R}(\theta), \quad \mathcal{R}(\theta) \;:=\; \mathbb{E}_{\rho_\theta}\big[\xi_t \cdot \ell_t^{\mathrm{base}}\big] \;\geq\; 0, \tag{6}$$

where $\rho_\theta$ is the joint rollout distribution induced by $\pi_\theta$, $p_\theta^{\mathrm{eval}}$, and the verifier (App. B.2). The surprise residual $\mathcal{R}(\theta)$ is the base loss restricted to the miscalibrated cells OF and US: PH minimizes a regularized objective in which surprising rollouts contribute through both their own loss and the surprise penalty.

**Miscalibration rate.** Define the agent's miscalibration rate against the verifier:

$$\mathcal{M}(\theta) \;:=\; \mathbb{E}_{\rho_\theta}\big[\xi_t\big] \;=\; \Pr_{\rho_\theta}\big[(z_t, y_t) \in \{\mathrm{OF}, \mathrm{US}\}\big], \tag{7}$$

which equals the binary Brier score of the deterministic predictor $z_t'{=}(z_t{+}1)/2$ against $y_t$ (App. B.2). Let $c(\theta) \;:=\; \mathbb{E}_{\rho_\theta}\big[\ell_t^{\mathrm{base}} \,\big|\, \xi_t = 1\big]$ denote the average base loss on surprising rollouts. By the tower rule,

$$\mathcal{R}(\theta) \;=\; c(\theta) \cdot \mathcal{M}(\theta), \tag{8}$$

which is an *exact* identity rather than an inequality (Prop. 3 of App. B.2). Eq. 8 admits two descent pathways for the residual: (i) reducing $\mathcal{M}(\theta)$ by aligning the prospective belief with the verifier outcome, or (ii) reducing $c(\theta)$ by lowering the base loss on the miscalibrated set. Each PH gradient step is one or the other (or both), without an explicit calibration loss ever entering the optimization.

**When does the residual identity bind?** Eq. 8 is informative as a calibration lever in the regime where $c(\theta)$ remains bounded away from zero. Three training phases follow naturally. Early on, both $c(\theta)$ and $\mathcal{M}(\theta)$ are large, and minimizing $\mathcal{R}(\theta)$ admits both pathways simultaneously; this is the regime where PH delivers its largest empirical gains (§5). In mid-training, both factors decrease as the agent's self-model improves. At convergence, $c(\theta) \to 0$ and $\mathcal{M}(\theta) \to 0$ together, so $\mathcal{R}(\theta)$ vanishes and PH and the base method share the same fixed points (App. B.2). The calibration interpretation is therefore not a uniform statement about the optimization landscape but a description of how PH's residual *decomposes* during the operative training window.

**Reported metric.** Among the two cells contributing to $\mathcal{M}(\theta)$, OF carries the dominant practical weight (§3.3). We therefore report the *overconfident-failure rate*

$$\mathrm{OFR}(\theta) \;:=\; \Pr_{\rho_\theta}\big[\, y_t{=}0 \,\big|\, z_t{=}+1 \,\big] \;=\; \tfrac{\Pr[\mathrm{OF}]}{\Pr[z_t{=}+1]}, \tag{9}$$

the OF-conditional projection of $\mathcal{M}(\theta)$, alongside task success. This connects PH to implicit calibration of language models [15, 39, 22], in which post-training feedback induces well-calibrated confidence without an explicit calibration loss; PH makes such implicit calibration the explicit driver of optimization rather than an emergent side effect. We emphasize that this connection is *diagnostic*: PH does not minimize $\mathcal{M}(\theta)$ as an explicit objective, and the alignment between the PH update direction and $-\nabla_\theta \mathcal{M}(\theta)$ is a quantitative motivation rather than a pointwise guarantee (App. B.2).

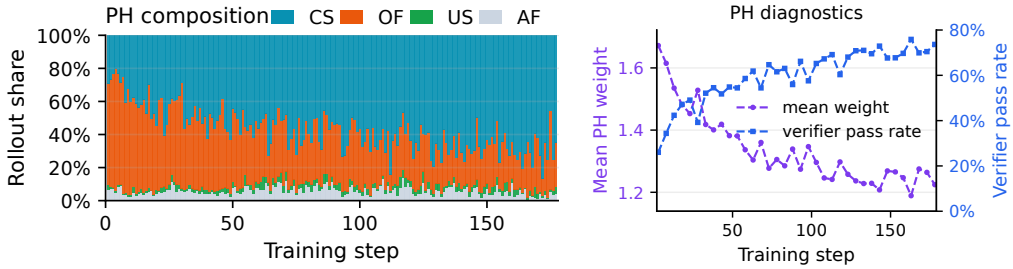



Figure 2: **Self-evaluation dynamics under PH on Science Q&A** ($\alpha{=}1$)**.** Left: per-step composition of the four PH cells (CS / OF / US / AF); overconfident failures shrink while confident successes grow. Right: the mean PH weight decays as the verifier pass rate rises, indicating that PH acts as an adaptive training signal that anneals as the prediction–reality gap closes.

Table 2: Single-turn results. Validation accuracy is the average / maximum logged `mean@16` (%); OFR $=\mathrm{OF}/(\mathrm{CS}+\mathrm{OF})$ (failure rate among predicted successes) and Surprise $=\mathrm{OF}+\mathrm{US}$ (total prediction–reality mismatch percentage);*Early*/*Late* are the first / last 5 training steps and $\Delta$=Late − Early.

| Dataset | Method | Val. Acc. | | OFR | | | Surprise | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Avg. ↑ | Max ↑ | Early | Late ↓ | Δ ↓ | Early | Late ↓ | Δ ↓ |
| Science Q&A | SDPO | 63.0 | 73.4 | 75.3 | 34.2 | -41.1 | 70.9 | 35.2 | -35.7 |
| | +Random | 65.4 | 74.3 | 74.1 | 33.3 | -41.7 | 69.0 | 35.3 | -33.7 |
| | +Failure-only | 65.1 | 74.7 | 73.9 | 25.0 | -48.9 | 69.0 | 24.4 | -44.6 |
| | +PH ($\alpha = 0.5$) | 65.7 | 75.5 | 76.8 | **20.9** | **-56.0** | 71.7 | 21.8 | **-49.9** |
| | +PH ($\alpha = 1.0$) | 64.0 | 74.2 | 73.2 | 23.9 | -49.3 | 68.6 | **21.1** | -47.5 |
| | +PH ($\alpha = 2.0$) | **65.8** | **76.3** | 74.4 | 24.9 | -49.5 | 69.9 | 24.8 | -45.1 |
| Tool Use | SDPO | 57.7 | 60.2 | 77.7 | 40.7 | -37.0 | 72.8 | 41.6 | -31.2 |
| | +Random | 58.2 | 61.6 | 77.4 | 36.3 | -41.0 | 72.3 | 36.2 | -36.1 |
| | +Failure-only | 58.1 | 60.6 | 76.9 | 39.3 | -37.6 | 72.7 | 39.2 | -33.5 |
| | +PH ($\alpha = 0.5$) | 57.3 | 61.0 | 77.9 | 36.2 | -41.7 | 73.0 | 36.2 | -36.8 |
| | +PH ($\alpha = 1.0$) | **58.2** | **61.7** | 76.6 | **33.6** | **-43.0** | 71.9 | **33.5** | **-38.4** |
| | +PH ($\alpha = 2.0$) | **58.2** | 60.9 | 77.6 | 38.2 | -39.4 | 73.0 | 38.3 | -34.7 |

# 5 Experiments

## 5.1 Experimental Setup

We evaluate PH in two complementary regimes that stress retrospective signals of different fidelity. **Single-turn** uses verifiers on SCIENCE Q&A [9] and TOOL USE [38] with SDPO [14] as the base loss. **Multi-turn** uses an LLM-based PRM judge on the OpenClaw-RL [42] PERSONAL AGENT task (GPT-4.1 user simulator), where PH plugs into RL (GRPO [33]), OPD [1, 44, 14], and their Combined variant. All main runs use OLMO-3-7B-INSTRUCT [29] as both policy and self-evaluator (distinct prompts, shared parameters). The prospective predictor uses $M{=}1$ sample in single-turn (deterministic verifier) and $M{=}3$ in multi-turn (stochastic PRM); cross-scale evidence on GPT-OSS-20B [2] and full hyperparameters are deferred to App. D.

**Baselines and ablations.** The boundary case $\alpha{=}0$ recovers the base method exactly (*Baseline*). To separate PH's surprise-based *selection* of which rollouts to upweight from the contribution of additional gradient mass, we compare against two strong reweighting controls (single-turn, App. D.3): **Random** Reweight, which assigns the same per-step total weight as PH but to a uniformly sampled subset of rollouts, and **Failure-only** Reweight, which upweights all verifier failures ($y_t{=}0$). Both use the same multiplier $(1{+}\alpha)$ as PH, so only the selection rule differs.

**Metrics and Evaluations.** Capability: `mean@16` (single-turn) or external-judge score (multi-turn). Calibration: $\mathrm{OFR}{=}\Pr[y_t{=}0 \mid z_t{=}{+}1]$, surprise rate $\mathcal{M}(\theta){=}\Pr[\xi_t{=}1]$, and prospective accuracy $\mathrm{PAcc}{=}1{-}\mathcal{M}(\theta)$. Single-turn tables report *Early*/*Late* as the average over the first/last 5 training steps; multi-turn tables report *Avg* over post-initial checkpoints, *Best*, and $\Delta_{\mathrm{best}}{=}\mathrm{Best}{-}\mathrm{initial}$. All headline numbers are aggregated over 3 seeds.

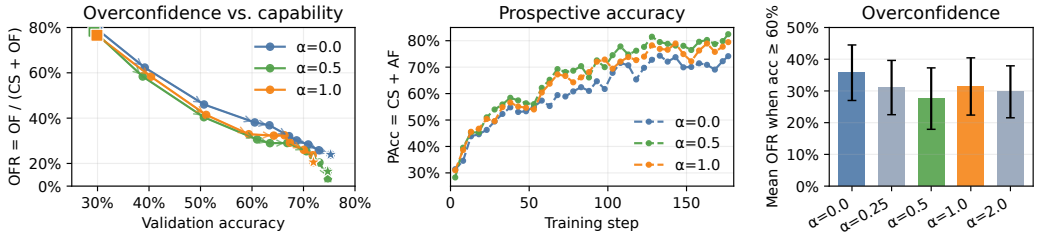



Figure 3: Capability and calibration dynamics on Science Q&A across $\alpha$. **Left:** OFR vs. Validation Accuracy; **Middle:** prospective accuracy over training. **Right:** Effect of $\alpha$ on overconfidence rate.

Table 3: Multi-turn personal-agent results (GPT-4.1 simulator). PH plugged into RL, OPD, and the Combined recipe; performance is the teacher-side personalization-evaluator score, calibration is computed over teacher-side rollouts within mixed training, $\Delta_{\text{best}}$=Best − initial.

| | | Capability | | | Calibration | | |
|---|---|---|---|---|---|---|---|
| Method | Setting | Avg. ↑ | Best ↑ | $\Delta_{\text{best}}$ ↑ | OFR ↓ | Surprise ↓ | Prosp. Acc. ↑ |
| RL | GRPO | 68.6 | 77.7 | +8.5 | 5.3 | 16.5 | 83.5 |
| | +PH ($\alpha = 0.5$) | 72.6 | 77.7 | +8.7 | 4.9 | 15.0 | 85.0 |
| | +PH ($\alpha = 1.0$) | **75.8** | **82.9** | **+14.8** | 3.1 | **12.2** | **87.8** |
| | +PH ($\alpha = 2.0$) | 68.6 | 77.3 | +9.4 | **2.5** | 16.0 | 84.0 |
| OPD | SDPO | 54.0 | 65.6 | +2.5 | 15.9 | 37.2 | 62.8 |
| | +PH ($\alpha = 0.5$) | 61.8 | **77.7** | +5.6 | **1.5** | **13.7** | **86.3** |
| | +PH ($\alpha = 1.0$) | **69.3** | 75.2 | +2.7 | 2.6 | 18.1 | 81.9 |
| | +PH ($\alpha = 2.0$) | 59.2 | 77.3 | **+17.1** | 5.0 | 17.2 | 82.8 |
| Combined | RL+OPD | 61.2 | 73.5 | +0.8 | 10.0 | 40.1 | 59.9 |
| | +PH ($\alpha = 0.5$) | **66.5** | 75.6 | **+8.1** | 4.3 | **28.4** | **71.6** |
| | +PH ($\alpha = 1.0$) | 65.1 | **78.5** | **+8.1** | 6.8 | 36.0 | 64.0 |
| | +PH ($\alpha = 2.0$) | 60.3 | 64.4 | +6.0 | **2.5** | 29.5 | 70.5 |

### 5.2 Single-Turn Verifiable Tasks

Across both domains in Table 2, PH preserves or slightly improves SDPO while substantially shrinking calibration error. On Science Q&A, late-training OFR drops from $34.2\%$ (SDPO) to $\mathbf{20.9}\%$ at $\alpha$=0.5, and the surprise rate tracks OFR closely ($35.2\%\rightarrow\mathbf{21.8}\%$). On Tool Use, $\alpha$=1.0 jointly attains the best capability and calibration (`mean@16`=$\mathbf{58.2}/\mathbf{61.7}$, OFR $\mathbf{33.6}\%$, surprise $\mathbf{33.5}\%$). Figure 4 replicates the SDPO vs. SDPO+PH comparison on GPT-OSS-20B. Using the same $\alpha$=1.0, PH delivers comparable or larger improvements on all three metrics, showing that it tracks a structural property of hindsight-only training that persists across model scale.

**Selection, not mass, drives the gain.** A natural concern is that PH simply adds gradient mass to a subset of rollouts; any reweighting with the same per-step total weight might do equally well. **Random** reweighting leaves calibration essentially unchanged on Science Q&A (OFR $33.3\%$ vs. SDPO's $34.2\%$) and improves it only modestly on Tool Use; the gain over SDPO is far smaller than PH's. **Failure-only** reweighting recovers most of PH's OFR reduction on Science Q&A ($25.0\%$ vs. $20.9\%$) but trails PH on the surprise rate ($24.4\%$ vs. $\mathbf{21.8}\%$). The OFR gap reflects that single-turn miscalibration is OF-dominated (Figure 5), so a 1D outcome-axis baseline already captures most of it; the surprise-rate gap reflects the cells PH amplifies but failure-only misses underconfident successes.

**PH is a self-extinguishing adaptive curriculum.** Figure 2 shows the per-step PH-cell composition and the corresponding PH-weight trajectory on Science Q&A. The overconfident-failure (OF) cell shrinks monotonically across training while the confident-success (CS) cell grows; the underconfident-success (US) and aware-failure (AF) cells stay small throughout. Crucially, the mean PH weight (right) decays as the verifier pass rate rises: PH amplifies gradient signal early when miscalibration is rampant and naturally anneals once the policy has learned to predict its own outcomes, matching the asymptotic argument in App. B.2 ($w_t\rightarrow 1$ as $\xi_t\rightarrow 0$). This means PH does not require a learning-rate schedule for the surprise weight: the data itself controls the curriculum.

**Calibration emerges as a byproduct of surprise reweighting.** Figure 3 (middle) tracks prospective accuracy $\text{PAcc}(\theta)=1-\mathcal{M}(\theta)$ across training. The pre-training self-evaluator is essentially uncalibrated but PH lifts PAcc from $\sim 30\%$ to $\sim 80\%$ while the SDPO baseline plateaus near $73\%$. The predictor receives no direct gradient (the reweighting is stop-gradient on $\xi_t$); the gap is therefore a *byproduct* of policy optimization, evidence that PH co-trains a calibrated self-model alongside the policy at no additional parameter cost. This is the mechanism behind the simultaneous capability and calibration improvement: the same gradient steps that reduce $\ell^{\text{base}}$ on miscalibrated rollouts also drive $\mathcal{M}(\theta)\rightarrow 0$ via the parameter-sharing channel. The full $\alpha$-sweep confirms the entire window improves both axes simultaneously, with $\alpha$=0.5 attaining the lowest mean OFR at matched capability.

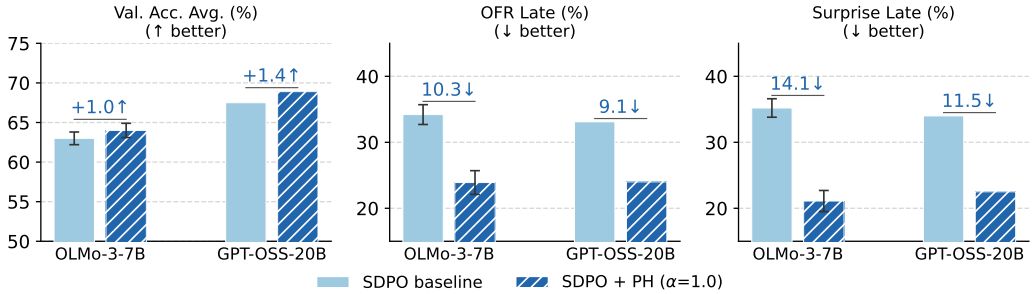



Figure 4: Capability and calibration gains from PH transfer across model scale using SDPO baseline and SDPO+PH ($\alpha$=1.0) on OLMo-3-7B-Instruct and gpt-oss-20B.

### 5.3 Multi-Turn Personal Agent

**Multi-turn shifts the dominant miscalibration mode from OF to US.** Before discussing capability and calibration numbers, we characterize the *kind* of miscalibration the multi-turn regime produces. Figure 5 aggregates $(z_t, y_t)$ pairs across training under $\alpha$=1.0 and contrasts single-turn tasks with the three multi-turn base methods. The picture is structurally different: while single-turn rollouts are dominated by overconfident failures (OF= 34.1%, US= 2.7%), multi-turn rollouts shift the dominant miscalibration to *underconfident successes* (US), which grow monotonically across base methods, RL (13.8%) < OPD (27.9%) < Combined (34.5%), while OF stays uniformly small (4–5%). The two regimes therefore expose complementary failure modes: single-turn agents over-trust wrong actions, while multi-turn agents under-trust correct ones.

**PH improves all three base methods, with the largest lift on OPD.** At $\alpha$=1.0, PH improves the trajectory-averaged teacher-side score over the baseline in every block of Table 3: RL (+7.2), OPD (+15.3), and Combined (+3.9). The OPD block is the most striking: its baseline has the weakest absolute performance, and PH simultaneously stabilizes the trajectory (+15.3 Avg) and raises the peak. The cell-composition picture above is consistent with the OPD-vs-RL ordering: OPD's privileged-context teacher leaves the student with substantially more underconfident successes than RL (US: 27.9% vs. 13.8%), and PH amplifies precisely those rollouts. The Combined block is a partial exception to this ordering since its US share (34.5%) is the highest of the three, yet its PH gain is the smallest.

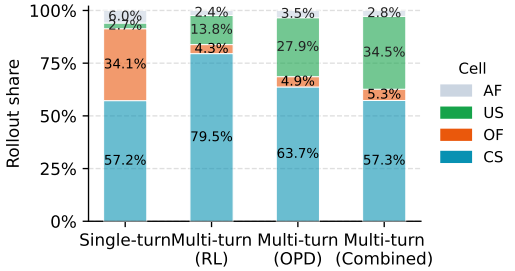



Figure 5: PH cell composition differs structurally between single-turn and multi-turn settings.

**Calibration improves uniformly at $\alpha$=1.0.** At the headline $\alpha$=1.0, OFR drops in every block and prospective accuracy rises in every block, peaking at **87.8**% (RL+PH, $\alpha$=1.0) and **86.3**% (OPD+PH, $\alpha$=0.5). The surprise gate concentrates gradient mass on exactly the cells (OF, US) that contribute to $\mathcal{M}(\theta)$, and those cells empirically shrink under PH training. A per-turn breakdown (Figure 8) shows that PH compresses overconfidence most in late-turn positions, where the agent's self-model is least developed. Larger $\alpha$ trades stability for further OFR reduction: at $\alpha$=2.0, the RL block reaches the lowest OFR in the table (2.5%) but the predictor collapses toward a near-constant $z_t$=−1 output (Surprise rate 16.0%, prospective accuracy 84.0%), so the calibration improvement comes at the cost of a degraded self-model rather than from genuinely better belief–outcome agreement. Replacing the GPT-4.1 user simulator with GPT-5.4 preserves the qualitative pattern of Table 3 (see Figure 9).

**What PH is doing, across regimes.** The base method's gradient is invariant to the agent's prospective belief; the surprise indicator $\xi_t$ is the per-rollout realization of the privileged-information gap $I^\star$ that this invariance hides. Three observations across both regimes are consistent with this view: (i) calibration improves as a byproduct of optimization, so the same gradient steps that drive the base loss also reduce $\mathcal{M}(\theta)$; (ii) the dominant miscalibration cell flips between regimes yet the same update rule covers both; and (iii) the surprise weight self-extinguishes as the prediction–reality gap closes,

so PH asymptotically reduces to the base method. Together these suggest that what PH amplifies is not a particular failure mode but the existence of a prediction–reality discrepancy itself, which is why the same scalar $\alpha$ transfers across tasks, base methods, and model scales without re-tuning.

## 6 Conclusion

Prospective Hindsight treats the gap between an agent's prospective belief and the verifier's retrospective outcome as a learning resource rather than a diagnostic, amplifying the gradient on exactly the rollouts where retrospective signals are uninformative. The same principle should extend beyond the binary verifiers and per-turn PRMs studied here, to longer horizons such as software engineering, terminal-use, GUI, and research agents, and to base methods beyond GRPO and on-policy distillation. We see PH as a first but critical step toward reliable agent-RL pipelines that optimize task performance and self-calibration jointly, by construction rather than through a separate calibration objective.

## Acknowledgements and Disclosures

This work was supported by Salesforce AI Research. No external funding was received for this work. The authors declare no competing interests beyond their institutional affiliations.

## References

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

[2] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

[3] 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.

[4] Fangming Cui, Ruixiao Zhu, Cheng Fang, Sunan Li, and Jiahong Li. Rethinking agentic reinforcement learning in large language models. *arXiv preprint arXiv:2604.27859*, 2026.

[5] Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training lms to reason about their uncertainty. *arXiv preprint arXiv:2507.16806*, 2025.

[6] Ken Ding. Hdpo: Hybrid distillation policy optimization via privileged self-distillation. *arXiv preprint arXiv:2603.23871*, 2026.

[7] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

[8] Jinhao Duan, James Diffenderfer, Sandeep Madireddy, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. Uprop: Investigating the uncertainty propagation of llms in multi-step agentic decision-making. *arXiv preprint arXiv:2506.17419*, 2025.

[9] Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*, 2024.

[10] Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595, 2024.

[11] Tobias Groot and Matias Valdenegro-Toro. Overconfidence is key: Verbalized uncertainty evaluation in large language and vision-language models. *arXiv preprint arXiv:2405.02917*, 2024.

[12] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[13] Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Towards uncertainty-aware language agent. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6662–6685, 2024.

[14] Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

[15] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[16] Adam Tauman Kalai, Ofir Nachum, Santosh S Vempala, and Edwin Zhang. Why language models hallucinate. *arXiv preprint arXiv:2509.04664*, 2025.

[17] Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

[18] Michael Kirchhof, Gjergji Kasneci, and Enkelejda Kasneci. Position: Uncertainty quantification needs reassessment for large language model agents. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025.

[19] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023.

[20] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[21] Jixuan Leng, Chengsong Huang, Banghua Zhu, and Jiaxin Huang. Taming overconfidence in LLMs: Reward calibration in RLHF. *arXiv preprint arXiv:2410.09724*, 2024.

[22] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

[23] Hao Liu, Zi-Yi Dou, Yixin Wang, Nanyun Peng, and Yisong Yue. Uncertainty calibration for tool-using language agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16781–16805, 2024.

[24] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[25] Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. https://thinkingmachines.ai/blog/on-policy-distillation.

[26] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

[27] Potsawee Manakul, Adian Liusie, and Mark Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017, 2023.

[28] Changdae Oh, Seongheon Park, To Eun Kim, Jiatong Li, Wendi Li, Samuel Yeh, Xuefeng Du, Hamed Hassani, Paul Bogdan, Dawn Song, et al. Uncertainty quantification in llm agents: Foundations, emerging challenges, and opportunities. *arXiv preprint arXiv:2602.05073*, 2026.

[29] Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

[30] Dmitry Pechyony and Vladimir Vapnik. On the theory of learnining with privileged information. *Advances in neural information processing systems*, 23, 2010.

[31] Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. On-policy self-distillation for reasoning compression. *arXiv preprint arXiv:2603.05433*, 2026.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[33] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[34] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

[35] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

[36] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[37] Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

[38] Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

[39] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442, 2023.

[40] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

[41] Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. Tcod: Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents. *arXiv preprint arXiv:2604.24005*, 2026.

[42] 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.

[43] Sin-Han Yang, Cheng-Kuang Wu, Chieh-Yen Lin, Yun-Nung Chen, Hung-yi Lee, and Shao-Hua Sun. On calibration of large language models: From response to capability. *arXiv preprint arXiv:2602.13540*, 2026.

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

[45] Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025.

[46] Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems? *arXiv preprint arXiv:2509.03312*, 2025.

[47] Jiaxin Zhang, Prafulla Kumar Choubey, Kung-Hsiang Huang, Caiming Xiong, and Chien-Sheng Wu. Agentic uncertainty quantification. *arXiv preprint arXiv:2601.15703*, 2026.

[48] Jiaxin Zhang, Wendi Cui, Zhuohang Li, Lifu Huang, Bradley A Malin, Caiming Xiong, and Chien-Sheng Wu. From passive metric to active signal: The evolving role of uncertainty quantification in large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 41525–41544, 2026.

[49] Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley Malin, and Sricharan Kumar. Sac3: reliable hallucination detection in black-box language models via semantic-aware cross-check consistency. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15445–15458, 2023.

[50] Jiaxin Zhang, Xiangyu Peng, Qinglin Chen, Qinyuan Ye, Caiming Xiong, and Chien-Sheng Wu. The illusion of certainty: Decoupling capability and calibration in on-policy distillation. *arXiv preprint arXiv:2604.16830*, 2026.

[51] Jiaxin Zhang, Caiming Xiong, and Chien-Sheng Wu. Agentic confidence calibration. *arXiv preprint arXiv:2601.15778*, 2026.

[52] Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025.

[53] Qiwei Zhao, Dong Li, Yanchi Liu, Wei Cheng, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Huaxiu Yao, Chen Zhao, et al. Uncertainty propagation on llm agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6073, 2025.

[54] Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

[55] Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

# Contents

## Contents


**1 Introduction** **2**

**2 Related work** **3**

**3 The Hindsight Trap in Agent RL** **3**

3.1 Agentic Reinforcement Learning . . . . . 3

3.2 Retrospective Training Methods . . . . . 3

3.3 The Self-Evaluation Blind Spot . . . . . 4

3.4 Why Prediction–Reality Gaps Are Informative . . . . . 4

**4 The Prospective Hindsight Method** **4**

4.1 Prospective Self-Evaluator . . . . . 5

4.2 A Calibration Taxonomy of Rollouts . . . . . 5

4.3 The Prospective Hindsight Update Rule . . . . . 5

4.4 PH as Calibration through Reweighting . . . . . 5

**5 Experiments** **7**

5.1 Experimental Setup . . . . . 7

5.2 Single-Turn Verifiable Tasks . . . . . 8

5.3 Multi-Turn Personal Agent . . . . . 9

**6 Conclusion** **10**

**A Extended Related Work and Discussion** **16**

**B Theoretical Details and Proofs** **17**

B.1 Proof of Proposition 1 . . . . . 17

B.2 PH as Calibration through Reweighting: Detailed Analysis . . . . . 18

**C Algorithm and Implementation Details** **22**

C.1 Base Method Advantage Formulations . . . . . 22

C.2 PH Algorithm Details . . . . . 22

C.3 Discrete and Continuous PH Gates . . . . . 23

C.4 Honest Uncertainty: The $z_t = 0$ Class . . . . . 24

**D Additional Experimental Setup** **24**

D.1 Single-Turn Verifiable Tasks . . . . . 24

D.2 Multi-Turn Personal Agent . . . . . 25

D.3 Reweighting-Baseline Protocols . . . . . 27

D.4 Compute and Reproducibility . . . . . 28

**E Additional Experiment Results and Analysis** **28**

E.1 Single-Turn $\alpha$-Sweep: SDPO + PH on Science Q&A . . . . . 28

E.2 Multi-Turn $\alpha$-Sweep: Per-Round Capability and Per-Turn Calibration . . . . . 28

**F Prompt Templates** **30**

F.1 Single-Step Tasks . . . . . 30

F.2 Multi-Turn Tasks . . . . . 32

**G Limitations and Future Work** **37**

**H Broader Impact** **38**

**I Licenses for Existing Assets** **38**

**J Declaration of LLM Usage** **39**

# A Extended Related Work and Discussion

**Agent RL: from outcome rewards to process supervision.** Reinforcement learning for language-model agents has progressed from sparse outcome-only rewards (e.g., GRPO [33] applied at the trajectory level) to dense, step-level supervision delivered by process reward models and structured frameworks. OpenClaw-RL [42] provides a unified asynchronous training stack that combines binary RL, on-policy distillation, and their composition in one rollout–score–update loop, together with a per-turn PRM that grades each turn from the next environmental state. Buening et al. [3] push this further by showing that raw multi-turn user interactions are already sufficient supervision for self-distillation, eliminating the need for a separately trained reward model. Across these systems, the per-turn signal varies considerably in fidelity and modality: rule-based binary verifiers for verifiable single-turn tasks, learned scalar PRMs for soft multi-turn signals, and hindsight-teacher token distributions for OPD [25, 54]. What does *not* vary is the structural property that motivates this paper: every learning signal is computed from quantities observable only after the action is committed, and the agent's prospective belief at action time is invisible to the gradient. Prospective Hindsight is orthogonal to the choice of base method, leaves the post-action signal untouched, and adds a single per-rollout reweighting on top.

**Hindsight distillation and the OPD lineage.** On-policy distillation [1] introduced the idea of distilling a teacher's improved actions into the student under on-policy rollouts; subsequent work has extended this in several directions. Ye et al. [44] formalize on-policy context distillation, in which the teacher conditions on a privileged context $c_t$ that the student does not see at deployment; Hübotter et al. [14] and Shenfeld et al. [34] develop self-distillation variants in which the teacher and student share parameters and the privileged context is constructed on-the-fly from verified-correct rollouts. We position PH as a complementary mechanism in this lineage. Standard OPD [25] uses privileged information to construct *better actions* for the teacher, and the resulting per-token signal is the dominant training signal. PH instead uses the gap between privileged and non-privileged *evaluations* to identify which rollouts deserve more weight, and absorbs OPD's per-token signal into a single per-rollout multiplier. The two mechanisms can be applied jointly without interaction: in our multi-turn experiments PH is plugged on top of GRPO, OPD, and the OpenClaw-RL Combined recipe with no method-specific tuning.

**Calibration and self-awareness in LLMs.** Modern confidence calibration in LLMs draws on several lines of work. At the diagnostic level, Guo et al. [12] pioneered the study of miscalibrated softmax outputs in deep networks, and Kadavath et al. [15] systematically measured whether language models "know what they know" across knowledge tasks. Methods to elicit calibrated confidence include verbalized confidence prompts [39, 22] and sampling-based estimators of semantic uncertainty [19]; Geng et al. [10] provide a survey. At the agentic level, Kirchhof et al. [18] argue for trajectory-level calibration diagnostics, and a series of recent papers [8, 53, 23, 48] quantify uncertainty propagation through multi-step trajectories; Han et al. [13] use uncertainty as an inference-time control signal. Kalai et al. [16] take a different angle and argue that hallucination itself is incentivized by training procedures that reward confident guesses over calibrated abstention, linking training-procedure design to overconfidence at the model-output level. This literature largely treats calibration as a diagnostic property to measure or to control at inference, rather than as a training signal to optimize. PH inverts that relationship: it converts the binary self-evaluation mismatch into a per-rollout training-time weight, and the binary Brier score of the agent's self-evaluator coincides with the miscalibration rate that the PH residual lower-bounds. Calibration emerges as a side effect of the same gradient steps that drive task success, rather than as a post-hoc check or a separate auxiliary loss.

**Calibration-augmented RL and the capability–calibration trade-off.** A line of recent work specifically targets the overconfidence introduced by hindsight distillation. CaOPD [50] attributes this overconfidence to the information asymmetry between the privileged-context teacher and the student and proposes *explicit confidence-token replacement* in the teacher distribution; Leng et al. [21] and Damani et al. [5] study calibration-augmented RL recipes that add explicit confidence supervision to the loss. Across these approaches, calibration improvements typically come at the cost of a modest capability regression, and the calibration objective is a separate term added to the base RL loss. PH is complementary along two dimensions. First, the base method is left untouched: PH does not rewrite confidence targets, does not add a separate calibration loss, and does not require a calibration-specific reward model. Second, the per-rollout reweighting connects calibration and

capability through the surprise residual decomposition: a gradient step that reduces the base loss on miscalibrated rollouts and a gradient step that reduces the miscalibration rate are both valid descent directions on. Empirically, this removes the capability–calibration trade-off seen in the cited baselines: PH preserves capability while substantially cutting OFR on single-turn and lifts both the trajectory-averaged and best-checkpoint scores across all three multi-turn base methods.

**Privileged information and prediction-error signals.** Our framing of the prediction–reality gap connects directly to Vapnik's Learning Under Privileged Information framework [40, 30]. In LUPI, training examples are accompanied by privileged information that is not available at deployment, and the goal is a predictor that uses standard inputs alone. Lopez-Paz et al. [24] unify LUPI with knowledge distillation, showing that distillation can be viewed as a transfer of privileged information from teacher to student. Our agent-RL setting instantiates LUPI with $(s_t, a_t)$ as the standard input, $y_t$ as the target, and $c_t$ as the privileged signal. Standard OPD-style methods use $c_t$ to construct improved actions for the teacher; PH instead uses the *evaluation* gap that $c_t$ creates, formalized as the non-negative privileged-information gap $I^\star(s_t, a_t)$ in Proposition 1. To our knowledge this is the first explicit use of $I^\star$ as a sample-level training signal in agent RL: the surprise indicator $\xi_t$ is the per-rollout realization of $I^\star$ from inside the agent's own model, and upweighting surprising rollouts concentrates gradient budget exactly where action-time information is insufficient.

A related but distinct line is intrinsic-motivation RL, in which prediction errors between two predictors of the next state drive *exploration* of novel states. PH differs in three ways. First, the prediction error is computed in the *outcome / evaluation* space, not the state space, and is used for training-signal prioritization rather than for action selection. Second, the predictor is the same model as the policy, parameterized by the same $\theta$ and queried via a different prompt, rather than a separate frozen network. Third, the surprise scalar is stop-gradient: it modulates the magnitude of the base gradient on the existing on-policy rollouts but does not introduce a new gradient component of its own. The two perspectives are complementary — one amplifies novelty in state space, the other amplifies prediction errors in evaluation space — and we view their combination as a natural avenue for future work.

**Self-correction, self-refinement, and reasoning chains.** A parallel line of work uses model self-evaluation to drive *inference-time* behavior: Reflexion [36] and Self-Refine [26] iterate on a draft response using a self-critic. PH operates at training time rather than inference time; the prospective prediction is consumed once, used to compute $\xi_t$, and discarded after the gradient step. This is a deliberate design choice: PH does not change deployment behavior, does not lengthen the agent's effective context, and does not require the agent to reason about its own confidence at inference. Inference-time self-correction is therefore composable with PH (it operates on the policy distribution after PH has shaped it), but the two mechanisms target different stages of the pipeline. A second related line is hallucination characterization (e.g., [27, 49, 54]), which studies the conditions under which LLMs produce unsupported content; we view PH's reduction of overconfident-failure rate as operating in the same regime, but at the agent-RL training-loop level rather than at the per-output classification level.

# B Theoretical Details and Proofs

## B.1 Proof of Proposition 1

Fix $(s_t, a_t)$ and apply the law of total variance to $y_t$ with conditioning random variable $c_t$:

$$\begin{aligned}\mathrm{Var}(y_t \mid s_t, a_t) &= \mathbb{E}\big[\mathrm{Var}(y_t \mid s_t, a_t, c_t) \,\big|\, s_t, a_t\big] \\ &\quad + \mathrm{Var}\big(\mathbb{E}[y_t \mid s_t, a_t, c_t] \,\big|\, s_t, a_t\big) \\ &= \mathbb{E}\big[\mathrm{Var}(y_t \mid s_t, a_t, c_t) \,\big|\, s_t, a_t\big] + I^\star(s_t, a_t), \end{aligned} \tag{10}$$

where the second equality uses the definition of $I^\star$. Both terms are non-negative, and the second vanishes if and only if $\mathbb{E}[y_t \mid s_t, a_t, c_t]$ is almost surely a function of $(s_t, a_t)$, that is, $c_t$ provides no additional information about $y_t$ beyond $(s_t, a_t)$. □

**Connection to Learning Under Privileged Information** The decomposition in Proposition 1 places the agent RL setting of §3.1 inside the Learning Under Privileged Information (LUPI) paradigm introduced by [40]. In LUPI, training examples are accompanied by privileged information available

only at training time, and the goal is a predictor that uses standard inputs alone. [30] formalize the statistical-learning theory of this setting. [24] unify LUPI with knowledge distillation, showing that distillation can be viewed as a transfer of privileged information from teacher to student.

Our setting instantiates LUPI with $(s_t, a_t)$ as the standard input, $y_t$ as the target, and $c_t$ as the privileged signal. The base methods of §3.2 use $c_t$ in different ways: GRPO uses $c_t$ only through the scalar reward $r_t = R(s_t, a_t, c_t)$; OPD-style methods condition the hindsight teacher distribution on $c_t$ and distill to a student that drops $c_t$ at inference. Both retain the blind spot of §3.3: training never asks the policy to predict $y_t$ from $(s_t, a_t)$ before $c_t$ arrives, so the privileged-information gap $I^\star(s_t, a_t)$ is never measured per step. Our method (§4) makes this gap an explicit training signal.

**Equivalent information-theoretic statement.** Equation (3) can equivalently be expressed in entropy form. For binary $y_t$, let $H(y \mid \cdot) := -\mathbb{E}[\log p(y \mid \cdot)]$. Then

$$H(y_t \mid s_t, a_t) - H(y_t \mid s_t, a_t, c_t) \;=\; I\big(y_t; c_t \,\big|\, s_t, a_t\big) \;\geq\; 0, \tag{11}$$

that is, the conditional mutual information between $y_t$ and $c_t$ given $(s_t, a_t)$ measures the same privileged-information increment as $I^\star$. The equality $I(y_t; c_t \mid s_t, a_t) = 0$ characterizes the degenerate case in which $I^\star(s_t, a_t) = 0$ for almost every $(s_t, a_t)$: no prospective predictor can be improved by access to $c_t$ at training time.

### B.2 PH as Calibration through Reweighting: Detailed Analysis

This appendix provides the formal setup, the exact residual identity and its consequences, a gradient-flow analysis of the PH update direction, and a discussion of the assumptions' empirical validity. We frame the results below as *motivation* for the calibration interpretation of PH outlined in §4.4, not as formal guarantees of monotone calibration descent.

#### Setup and notation

We work in a single-step task to keep notation light; the multi-turn extension is recorded in §B.2. Let $\theta \in \Theta$ parameterize a shared autoregressive model that defines:

- the policy $\pi_\theta(a \mid s)$, a conditional distribution over actions $a \in \mathcal{A}$ given state $s \in \mathcal{S}$;
- the prospective self-evaluator $p_\theta^{\text{eval}}(z \mid s, a)$, a conditional distribution over evaluator outputs $z \in \{-1, 0, +1\}$.

The verifier $y$ is sampled from a parameter-independent kernel $p_{\text{env}}(y \mid s, a)$ with $y \in \{0, 1\}$; the state distribution is $\rho_0(s)$. The joint rollout distribution induced by $\theta$ is

$$\rho_\theta(s, a, z, y) \;=\; \rho_0(s)\, \pi_\theta(a \mid s)\, p_\theta^{\text{eval}}(z \mid s, a)\, p_{\text{env}}(y \mid s, a). \tag{12}$$

We restrict attention to $z \in \{-1, +1\}$; the residual $z = 0$ class is treated separately in App. C.4 and contributes neutrally to all expressions below since $\xi_t = 0$ there. The surprise indicator (Eq. 4 of the main text) is

$$\xi(z, y) \;=\; \mathbb{1}\big[(z, y) \in \{\text{OF}, \text{US}\}\big], \tag{13}$$

and we write $\xi_t$ when the rollout is implicit. We assume the base loss $\ell_t^{\text{base}}(\theta)$ is non-negative; this is satisfied by all retrospective methods of §3.2 after additive constant shifts that do not affect the gradient.

#### Surprise-weighted decomposition

**Proposition 2** (Surprise-weighted decomposition)**.** *For any $\alpha \geq 0$,*

$$\mathbb{E}_{\rho_\theta}\big[\ell_t^{\text{PH}}\big] \;=\; \mathbb{E}_{\rho_\theta}\big[\ell_t^{\text{base}}\big] + \alpha \cdot \mathcal{R}(\theta), \quad \mathcal{R}(\theta) := \mathbb{E}_{\rho_\theta}\big[\xi_t \cdot \ell_t^{\text{base}}\big].$$

*The surprise residual $\mathcal{R}(\theta)$ is non-negative and vanishes if and only if $\xi_t \cdot \ell_t^{\text{base}} = 0$ holds $\rho_\theta$-almost surely.*

*Proof.* By Eq. 5 of the main text, $\ell_t^{\text{PH}} = (1 + \alpha\xi_t)\, \ell_t^{\text{base}} = \ell_t^{\text{base}} + \alpha\xi_t \ell_t^{\text{base}}$. Linearity of expectation gives the decomposition. Non-negativity of $\mathcal{R}$ follows from $\xi_t \in \{0, 1\}$ and $\ell_t^{\text{base}} \geq 0$. The vanishing condition is the standard a.s. zero condition for non-negative integrands. □

The two pathways for residual minimization—reducing $\Pr[\xi = 1]$ (calibration) or reducing $\ell^{\text{base}}$ on miscalibrated rollouts (base-objective improvement on those rollouts)—are formalized exactly in the next subsection.

### Exact residual identity

**Proposition 3** (Exact residual identity)**.** *Define the agent's miscalibration rate*

$$\mathcal{M}(\theta) \;:=\; \mathbb{E}_{\rho_\theta}\big[\xi_t\big] \;=\; \Pr_{\rho_\theta}\big[(z_t, y_t) \in \{\text{OF}, \text{US}\}\big], \tag{14}$$

*and the conditional mean base loss on surprising rollouts*

$$c(\theta) \;:=\; \mathbb{E}_{\rho_\theta}\big[\ell_t^{\text{base}} \,\big|\, \xi_t = 1\big]. \tag{15}$$

*Then*

$$\mathcal{R}(\theta) \;=\; c(\theta) \cdot \mathcal{M}(\theta). \tag{16}$$

*Proof.* By the tower rule,

$$\mathcal{R}(\theta) \;=\; \mathbb{E}\big[\xi_t \ell_t^{\text{base}}\big] \;=\; \mathbb{E}\big[\ell_t^{\text{base}} \,\big|\, \xi_t = 1\big] \cdot \Pr\big[\xi_t = 1\big] \;=\; c(\theta) \cdot \mathcal{M}(\theta). \qquad \square$$

Eq. 16 is an *exact* identity, not an inequality: the residual factors cleanly into a base-loss-on-miscalibrated component $c(\theta)$ and a miscalibration-rate component $\mathcal{M}(\theta)$. As a corollary, whenever $c(\theta) \geq c_0 > 0$ on a contiguous training window, $\mathcal{R}(\theta) \geq c_0 \cdot \mathcal{M}(\theta)$ on that window, and minimizing $\mathcal{R}(\theta)$ via PH must reduce $\mathcal{M}(\theta)$ at rate at least $c_0$ relative to the rate of residual descent.

**When does Eq. 16 drive calibration?** The identity is most informative as a calibration lever when both factors are non-negligible. Three regimes arise as training progresses:

- **Early training.** $c(\theta)$ is large (the base method has not yet learned to solve surprising rollouts) and $\mathcal{M}(\theta)$ is large (many rollouts are miscalibrated). Both pathways for residual minimization are active, and PH's surprise reweighting allocates additional gradient budget to cells where the prediction–reality gap is widest.
- **Mid training.** $c(\theta)$ and $\mathcal{M}(\theta)$ decrease together as the agent's self-model and the base policy co-evolve. The dual-pathway decomposition continues to hold, but the per-step weight of each pathway shifts.
- **Convergence.** $c(\theta) \to 0$ and $\mathcal{M}(\theta) \to 0$ jointly, $\mathcal{R}(\theta) \to 0$, and PH and the base method share the same fixed points (App. B.2). The identity becomes uninformative because both sides are zero, but PH itself is no longer doing work in this regime.

The bound $\mathcal{R}(\theta) \geq c_0 \cdot \mathcal{M}(\theta)$ is therefore not a uniform property of the optimization landscape: it characterizes the operative training window during which PH's calibration gains accrue.

**Equivalence to the binary Brier score.** For binary $z_t' = (z_t + 1)/2 \in \{0, 1\}$ and $y_t \in \{0, 1\}$, $(z_t' - y_t)^2 = \mathbb{1}(z_t' \neq y_t) = \xi_t$, so $\mathcal{M}(\theta)$ coincides with the standard binary Brier score $\mathbb{E}[(z_t' - y_t)^2]$ of the deterministic predictor $z_t'$ against the verifier $y_t$. We use the term *miscalibration rate* throughout because it more directly conveys the operational meaning—the fraction of rollouts on which the agent's prospective belief disagrees with the verifier outcome—and because the experimental section reports a one-sided projection of $\mathcal{M}$ rather than the full Brier score (§B.2).

### Connection to the overconfident-failure rate (OFR)

The miscalibration rate $\mathcal{M}(\theta)$ in Eq. 14 aggregates both miscalibrated cells, OF and US, symmetrically. The experimental section reports the OF-conditional projection

$$\text{OFR}(\theta) \;:=\; \Pr_{\rho_\theta}\big[y_t = 0 \,\big|\, z_t = +1\big] \;=\; \frac{\Pr[\text{OF}]}{\Pr[\text{CS}] + \Pr[\text{OF}]} \;=\; \frac{\Pr[\text{OF}]}{\Pr[z_t = +1]}. \tag{17}$$

This is a one-sided, conditional version of $\mathcal{M}$ focused on the OF cell, which is the dominant miscalibration mode singled out in §3.3. The two metrics are related by

$$\Pr[\text{OF}] \;\leq\; \mathcal{M}(\theta), \qquad \text{OFR}(\theta) \;=\; \Pr[\text{OF}] \,/\, \Pr[z_t = +1], \tag{18}$$

so reductions in $\mathcal{M}(\theta)$ imply reductions in $\Pr[\mathrm{OF}]$ whenever $\Pr[\mathrm{US}]$ does not increase, and reductions in $\Pr[\mathrm{OF}]$ imply reductions in $\mathrm{OFR}(\theta)$ whenever $\Pr[z_t = +1]$ does not collapse. Both monotonicity assumptions hold in our experiments: the prospective predictor's confidence rate $\Pr[z_t = +1]$ stays in a stable range across training, and the underconfident-success rate $\Pr[\mathrm{US}]$ remains a small fraction of $\mathcal{M}(\theta)$ at all checkpoints. We therefore report $\mathrm{OFR}(\theta)$ as a faithful empirical proxy for $\mathcal{M}(\theta)$ in §5.

**Gradient-flow analysis: when is the PH update calibration-aligned?**

We now characterize the direction in parameter space along which PH gradient flow moves, relative to the negative gradient of the miscalibration rate. The result below should be read as a *motivation* for why one would expect PH to reduce $\mathcal{M}(\theta)$, not as a guarantee that it does so monotonically in every step.

Let $u^{\mathrm{PH}} := -\nabla_\theta \mathbb{E}_{\rho_\theta}[\ell^{\mathrm{PH}}]$ and $u^{\mathrm{base}} := -\nabla_\theta \mathbb{E}_{\rho_\theta}[\ell^{\mathrm{base}}]$ denote the PH and base update directions. By Prop. 2,

$$u^{\mathrm{PH}} \;=\; u^{\mathrm{base}} - \alpha \nabla_\theta \mathcal{R}(\theta). \tag{19}$$

**Score-function decomposition of $\nabla_\theta \mathcal{R}$.** Applying the score-function trick to $\mathcal{R}(\theta) = \mathbb{E}_{\rho_\theta}[\xi_t \ell_t^{\mathrm{base}}]$,

$$\nabla_\theta \mathcal{R}(\theta) \;=\; \underbrace{\mathbb{E}_{\rho_\theta}[\xi_t \nabla_\theta \ell_t^{\mathrm{base}}]}_{\text{base-loss descent on } \{\xi=1\}} \;+\; \underbrace{\mathbb{E}_{\rho_\theta}[\xi_t \ell_t^{\mathrm{base}} \nabla_\theta \log \rho_\theta]}_{\text{distributional shift in } \xi}, \tag{20}$$

where $\nabla_\theta \log \rho_\theta = \nabla_\theta \log \pi_\theta(a \mid s) + \nabla_\theta \log p_\theta^{\mathrm{eval}}(z \mid s, a)$, since $\rho_0$ and $p_{\mathrm{env}}$ are parameter-free. Similarly,

$$\nabla_\theta \mathcal{M}(\theta) \;=\; \mathbb{E}_{\rho_\theta}[\xi_t \nabla_\theta \log \rho_\theta]. \tag{21}$$

**Theorem 4** (PH gradient under uniform-loss assumptions)**.** *Let $\bar{\ell}(\theta) := \mathbb{E}_{\rho_\theta}[\ell_t^{\mathrm{base}} \mid \xi_t = 1] = c(\theta)$ and $\delta(\theta) := \mathrm{Var}_{\rho_\theta,\, \xi_t=1}[\ell_t^{\mathrm{base}}]$. Suppose $\mathbb{E}[\xi_t \|\nabla_\theta \log \rho_\theta\|^2] < \infty$. Then*

$$u^{\mathrm{PH}} \;=\; u^{\mathrm{base}} \;-\; \alpha \cdot \mathbb{E}_{\rho_\theta}[\xi_t \nabla_\theta \ell_t^{\mathrm{base}}] \;-\; \alpha\, \bar{\ell}(\theta) \cdot \nabla_\theta \mathcal{M}(\theta) \;+\; r(\theta), \tag{22}$$

*where the remainder $r(\theta)$ satisfies*

$$\|r(\theta)\| \;\le\; \alpha \sqrt{\delta(\theta)} \cdot \sqrt{\mathbb{E}[\xi_t \|\nabla_\theta \log \rho_\theta\|^2]}. \tag{23}$$

*In the limit $\delta(\theta) \to 0$ (the base loss is constant on surprising rollouts), the PH update direction differs from the base direction by exactly a $-\alpha\bar{\ell} \cdot \nabla_\theta \mathcal{M}(\theta)$ component plus a base-loss descent component on the miscalibrated set.*

*Proof.* Decompose the second term of Eq. 20 via the identity

$$\mathbb{E}[\xi_t \ell_t^{\mathrm{base}} \nabla_\theta \log \rho_\theta] \;=\; \bar{\ell} \cdot \mathbb{E}[\xi_t \nabla_\theta \log \rho_\theta] \;+\; \mathbb{E}\big[\xi_t (\ell_t^{\mathrm{base}} - \bar{\ell}) \nabla_\theta \log \rho_\theta\big].$$

The first term equals $\bar{\ell} \cdot \nabla_\theta \mathcal{M}(\theta)$ by Eq. 21. The second term is bounded in norm by Cauchy–Schwarz:

$$\big\|\mathbb{E}[\xi_t (\ell_t^{\mathrm{base}} - \bar{\ell}) \nabla_\theta \log \rho_\theta]\big\| \;\le\; \sqrt{\mathrm{Var}_{\xi_t=1}[\ell_t^{\mathrm{base}}]} \cdot \sqrt{\mathbb{E}[\xi_t \|\nabla_\theta \log \rho_\theta\|^2]} \;=\; \sqrt{\delta(\theta)} \cdot \sqrt{\mathbb{E}[\xi_t \|\nabla_\theta \log \rho_\theta\|^2]},$$

which gives Eq. 23 after multiplication by $\alpha$. Substituting back into Eqs. 20 and 19 yields Eq. 22. □

*Remark* 1 (Empirical magnitude of the remainder). The remainder bound in Eq. 23 is not an assumption but a quantity measurable from training logs. We define the relative remainder

$$\rho(\theta) \;:=\; \frac{\sqrt{\delta(\theta)}}{c(\theta)} \;=\; \frac{\sqrt{\mathrm{Var}_{\xi_t=1}[\ell_t^{\mathrm{base}}]}}{\mathbb{E}_{\xi_t=1}[\ell_t^{\mathrm{base}}]}, \tag{24}$$

the coefficient of variation of the base loss conditional on $\xi_t = 1$. Theorem 4 is most informative when $\rho(\theta)$ is small; the limit $\rho(\theta) \to 0$ recovers the uniform-loss case in which the leading term $-\alpha\bar{\ell} \cdot \nabla_\theta \mathcal{M}$ dominates the remainder.

**Calibration alignment, qualified.** Taking the inner product of Eq. 22 with $-\nabla_\theta \mathcal{M}(\theta)$ gives

$$\begin{aligned}\left\langle u^{\mathrm{PH}}, -\nabla_\theta \mathcal{M}(\theta)\right\rangle \;=\; &\left\langle u^{\mathrm{base}}, -\nabla_\theta \mathcal{M}(\theta)\right\rangle \\ &+ \alpha \bar{\ell}(\theta) \cdot \|\nabla_\theta \mathcal{M}(\theta)\|^2 \\ &- \alpha \left\langle \mathbb{E}[\xi_t \nabla_\theta \ell_t^{\mathrm{base}}], -\nabla_\theta \mathcal{M}(\theta)\right\rangle \\ &+ \left\langle r(\theta), -\nabla_\theta \mathcal{M}(\theta)\right\rangle. \end{aligned} \tag{25}$$

Among the four terms on the right-hand side, the second is the leading calibration-aligned term and is non-negative; the fourth is bounded by $\alpha\sqrt{\delta(\theta)} \cdot \sqrt{\mathbb{E}[\xi_t \|\nabla_\theta \log \rho_\theta\|^2]} \cdot \|\nabla_\theta \mathcal{M}(\theta)\|$ via Eq. 23. The third term is a base-loss descent component on the miscalibrated set; its inner product with $-\nabla_\theta \mathcal{M}$ is sign-indeterminate in general, but in policy-gradient base losses $\nabla_\theta \ell_t^{\mathrm{base}}$ is bounded by the per-batch advantage normalization of §3.2, so its contribution scales with the magnitude of the base advantage.

The PH update direction is therefore *biased toward* $-\nabla_\theta \mathcal{M}(\theta)$ rather than *exactly aligned* with it. Whether this bias is large enough to drive monotone descent on $\mathcal{M}(\theta)$ depends on the relative magnitudes of the four terms in Eq. 25, which are themselves a function of the training step. We do not claim monotone descent; we claim that the leading term provides a calibration-aligned restoring force that is consistent with the empirical observation in §5 that $\mathrm{OFR}(\theta)$ decreases monotonically under PH training.

**Scope and limitations of the alignment claim.** We make four honest qualifications. First, Theorem 4 is an exact identity, but the calibration-alignment interpretation that follows it requires the leading $\bar{\ell} \cdot \nabla_\theta \mathcal{M}$ term to dominate the remainder—i.e., $\rho(\theta)$ to be small relative to $\bar{\ell}/\|\nabla_\theta \mathcal{M}\|$. Second, the $\mathbb{E}[\xi_t \nabla_\theta \ell_t^{\mathrm{base}}]$ term has indeterminate sign relative to $-\nabla_\theta \mathcal{M}(\theta)$, and we do not claim it contributes to calibration descent. Third, the analysis is gradient-flow, not finite-step SGD: under standard learning-rate and smoothness assumptions, the conclusions hold approximately for finite steps with higher-order corrections, but we have not derived sharp rates. Fourth, the analysis is per-step: it does not compose across the trajectory of training to yield a convergence-rate result on $\mathcal{M}(\theta)$.

We therefore frame Theorem 4 as a *motivation* for why PH's update direction would be expected to reduce $\mathcal{M}(\theta)$, not as a guarantee that it does so. The empirical observation that $\mathrm{OFR}(\theta)$ decreases monotonically under PH training across all base methods and tasks in §5 is then evidence that the motivation translates into practice, rather than a derivation of that fact.

### Discussion

**Single-step versus multi-turn.** The arguments above treat each rollout as a single $(s, a, z, y)$ tuple. In multi-turn settings, the per-turn surprise $\xi_t$ is computed from $(z_t, y_t)$ at turn $t$, and the joint distribution $\rho_\theta$ extends to a sequence over turns. Prop. 2 extends turn-by-turn (linearity of expectation across turns), and Theorem 4 applies at each turn under the same assumptions; the per-turn alignment is then aggregated over the trajectory. The remainder bound in Eq. 23 is typically larger in multi-turn settings because the per-turn variance of the base loss is amplified by stochasticity in the LLM-based PRM judge (App. D.2).

**Tabular versus neural-network policies.** Theorem 4 is stated as an exact gradient-flow identity modulo the explicit remainder $r(\theta)$. For finite-step gradient descent with neural-network policies, the conclusion holds approximately under standard learning-rate, smoothness, and Lipschitz assumptions, with equalities replaced by inequalities up to higher-order terms in the learning rate.

**Asymptotic behavior.** As $\theta$ converges, both $\ell^{\mathrm{base}}$ and $\xi$ approach zero on the support of $\rho_\theta$, so $c(\theta) \to 0$, $\mathcal{M}(\theta) \to 0$, and $\mathcal{R}(\theta) \to 0$. By Prop. 2, PH and the base method share the same fixed points; PH differs only in the optimization trajectory leading up to convergence, in which additional gradient budget is allocated to surprising rollouts via Eq. 16. The miscalibration rate also vanishes asymptotically, consistent with calibration being achieved at convergence.

**Relation to existing calibration losses.** The miscalibration rate $\mathcal{M}(\theta)$ is the binary specialization of mean-squared-error calibration (the Brier score). Expected Calibration Error (ECE), defined for predictors that emit confidence $p \in [0, 1]$, coincides with $\mathcal{M}(\theta)$ when predictions are binarized

via thresholding. Negative Log-Likelihood of the prospective evaluator is a different but related calibration objective; PH does not directly minimize NLL, but does so implicitly through the parameter-sharing channel that couples the policy and evaluator gradients. Among these losses, $\mathcal{M}$ is the one most naturally connected to the PH update rule, since PH's reweighting is itself an indicator-weighted scheme. The OF-conditional projection $\mathrm{OFR}(\theta)$ used in our experiments (Eq. 17) is a one-sided form of $\mathcal{M}$ tailored to the OF-focused narrative of §3.3.

# C Algorithm and Implementation Details

## C.1 Base Method Advantage Formulations

We give the explicit advantage formulations of the three retrospective base methods abstracted as $\ell^{\text{base}}$ in §3.2.

**GRPO [33].** For a group of $G$ rollouts $\{(s_t, a_t^{(g)}, r_t^{(g)})\}_{g=1}^{G}$ sharing the same query, the per-rollout advantage is

$$A_t^{\text{GRPO}} \;=\; \frac{r_t - \bar{r}}{\sigma_r + \epsilon}, \tag{26}$$

where $\bar{r} = \frac{1}{G}\sum_g r_t^{(g)}$ and $\sigma_r$ is the group standard deviation; $\epsilon > 0$ is a small constant for numerical stability.

**OPD [1, 44, 14].** The privileged context $c_t$ (in multi-turn, $c_t = s_{t+1}$) is used to construct a hindsight teacher conditioned on $(s_t, c_t)$, whose per-token action distribution is distilled into the student:

$$A_{t,j}^{\text{OPD}} \;=\; \log \pi_\theta(a_{t,j} \mid a_{t,<j}, s_t, c_t) - \log \pi_\theta(a_{t,j} \mid a_{t,<j}, s_t), \tag{27}$$

where $a_{t,j}$ is the $j$-th token of $a_t$ and $a_{t,<j}$ is its prefix.

**Combined [42].** The combined advantage sums the scalar and token-level signals:

$$A_{t,j}^{\text{comb}} \;=\; w_{\text{rl}} A_t^{\text{GRPO}} + w_{\text{opd}} A_{t,j}^{\text{OPD}}, \tag{28}$$

with $w_{\text{rl}} = w_{\text{opd}} = 1$ in our experiments.

In all three cases, the per-rollout (or per-token) base loss $\ell_t^{\text{base}}$ is the policy-gradient surrogate of the corresponding advantage; the precise form does not affect any analysis in §3–§4.

**How PH wraps each base method.** PH's per-rollout reweighting in Eq. 5 preserves all internal structure of the base advantage. For GRPO, the group normalization $\sigma_r$ is computed on the original rewards $r_t$, not the surprise-weighted loss. For OPD, the per-token signal $A_{t,j}^{\text{OPD}}$ is aggregated as in the base method; PH then multiplies the resulting per-rollout (single-turn) or per-turn (multi-turn) loss by $(1+\alpha\xi_t)$. The surprise scalar is broadcast across action tokens within the rollout/turn, which is the operation that absorbs OPD's per-token signal into the per-rollout reweighting.

## C.2 PH Algorithm Details

Algorithm 1 states the end-to-end training step. Inside any base method of §3.2, PH inserts three operations into each rollout step: (i) draw $M$ samples from the prospective self-evaluator and aggregate them by majority vote (§4.1); (ii) compute the surprise indicator $\xi_t$ from the resulting $z_t$ and the verifier outcome $y_t$ (§4.2, §4.3); (iii) multiply the base per-rollout loss by $1 + \alpha\xi_t$ (§4.3). The base optimizer then updates the parameters using the resulting PH loss; the rollout sampling distribution, the group reward normalization, the per-token aggregation in OPD, and any auxiliary regularizer in $\ell^{\text{base}}$ are all unchanged.

**Cost.** The only additional compute cost relative to the base method is $M$ prospective forward passes per rollout step (line 5 of Algorithm 1). With shared parameters between the action policy and the self-evaluator, no extra trainable parameters are introduced, so PH does not change the memory footprint of the base method beyond the storage of $z_t$ and $\xi_t$ per step. With $M = 1$, the prospective overhead is one extra forward pass per step. The verifier and privileged-context computations on line

**Algorithm 1** Prospective Hindsight (PH) training step.

**Require:** parameters $\theta$, batch of prompts $\{x_b\}_{b=1}^{B}$, base method (GRPO, OPD, or Combined), hyperparameters $\alpha \geq 0$ and $M \geq 1$

**for** each prompt $x_b$, $b = 1, \ldots, B$ **do**
  **for** each step (or turn) $t$ of the rollout **do**
    $a_t \sim \pi_\theta(\cdot \mid s_t)$ ▷ base method's action sampling
    **for** $m = 1, \ldots, M$ **do**
      $z_t^{(m)} \sim p_\theta^{\text{eval}}(\cdot \mid s_t, a_t)$ ▷ Eq. 4.1, prospective forward pass
    **end for**
    $z_t \leftarrow \text{maj}\big(z_t^{(1)}, \ldots, z_t^{(M)}\big)$
    Receive $c_t$ and $y_t$ from environment and verifier
    $\xi_t \leftarrow \mathbb{1}\big[(z_t, y_t) \in \{\text{OF}, \text{US}\}\big]$ ▷ Eq. 4
    Compute $\ell_t^{\text{base}}(\theta)$ following §3.2
    $\ell_t^{\text{PH}} \leftarrow \big(1 + \alpha \cdot \xi_t\big) \cdot \ell_t^{\text{base}}(\theta)$ ▷ Eq. 5
  **end for**
**end for**
$\theta \leftarrow \texttt{optimizer_step}\big(\theta,\ \nabla_\theta \sum_{b,\,t} \ell_t^{\text{PH}}\big)$ ▷ $\xi_t$ is stop-gradient
**return** $\theta$

8 are inherited unchanged from the base method, since the retrospective methods of §3.2 already require them. PH thus adds a single training-time hyperparameter $\alpha$ and a single inference-time hyperparameter $M$ on top of the base method.

**Special cases.** Two boundary cases are useful for sanity checks. First, $\alpha = 0$ collapses Algorithm 1 into the base method exactly, since $1 + \alpha\xi_t = 1$ for all $t$. Second, $M = 1$ removes the majority vote and uses a single prospective sample per rollout step, which is the configuration in all our single-turn experiments.

### C.3 Discrete and Continuous PH Gates

The PH update rule of Eq. 5 weights the base loss by $w_t = 1 + \alpha\,\xi_t$ for a non-negative surprise scalar $\xi_t$. We use two concrete realizations of $\xi_t$, depending on whether the retrospective signal is deterministic or stochastic; both share the same shape $w_t \in [1, 1 + \alpha]$ and the same asymptotic behaviour $w_t \to 1$ as the prospective belief and the realized outcome agree.

**Discrete gate (single-turn).** With a rule-based binary verifier $y_t \in \{0, 1\}$ and a single ternary prospective sample $z_t \in \{-1, 0, +1\}$ ($M$=1, App. D.1), the surprise indicator is the binary cell membership of Eq. 4,

$$\xi_t^{\text{disc}} \;=\; \mathbb{1}\big[(z_t, y_t) \in \{\text{OF}, \text{US}\}\big] \;\in\; \{0, 1\}, \qquad w_t \;=\; 1 + \alpha\,\xi_t^{\text{disc}} \;\in\; \{1,\ 1{+}\alpha\}. \tag{29}$$

Honest-uncertainty rollouts ($z_t = 0$) are folded into $\xi_t = 0$ to preserve the neutral-class semantics of §C.4, so $w_t = 1$ on $\{z_t = 0\}$.

**Continuous gate (multi-turn).** With an LLM-based per-turn PRM judge that emits an aggregated turn score $r_t \in \{-1, 0, +1\}$ and $M$ predictor samples averaged into $\bar{z}_t \in [-1, +1]$ (App. D.2), we use the continuous relaxation

$$\xi_t^{\text{cont}} \;=\; \tfrac{1}{2}\,\big|\bar{z}_t - r_t\big| \;\in\; [0, 1], \qquad w_t \;=\; 1 + \alpha\,\xi_t^{\text{cont}} \;\in\; [1,\ 1{+}\alpha]. \tag{30}$$

The factor $\frac{1}{2}$ rescales the maximum disagreement $|\bar{z}_t - r_t| = 2$ (i.e., $\bar{z}_t = +1$, $r_t = -1$ or vice versa) to $\xi_t = 1$. Eq. 30 reduces to Eq. 29 on the corners of the $(\bar{z}_t, r_t)$ grid: when both $\bar{z}_t \in \{-1, +1\}$ and $r_t \in \{-1, +1\}$, $\xi_t^{\text{cont}} = \xi_t^{\text{disc}}$. Intermediate values of $\bar{z}_t$ encode predictor uncertainty (averaged over $M$ samples) and intermediate values of $r_t$ encode PRM-judge uncertainty; the continuous gate carries this uncertainty through to $w_t$ rather than collapsing it to $\{0, 1\}$.

**Self-extinguishing curriculum.** Both gates share the limiting behaviour $w_t \to 1$ as $\xi_t \to 0$, i.e., as the prospective belief and the realized outcome converge. This is the mechanism behind the

empirical PH-weight decay reported in Figs. 2 and 7: as training closes the prediction–reality gap, the prospective signal naturally fades and the PH loss reduces to the base loss. By Proposition 2, the surprise residual $\mathcal{R}(\theta) \to 0$ in the same limit, so PH and the base method share the same fixed points (App. B.2).

**Implementation flag.** A single training-time flag `SURPRISE_MODE` selects between the two gates. Setting `SURPRISE_MODE=discrete` uses Eq. 29; setting `SURPRISE_MODE=continuous` uses Eq. 30. The single-turn experiments of §5.2 use the discrete gate; the multi-turn experiments of §5.3 use the continuous gate. All other PH-side hyperparameters ($\alpha$, $M$, weight clip) are independent of this choice.

### C.4 Honest Uncertainty: The $z_t = 0$ Class

The prospective self-evaluator of §4.1 is prompted to return a ternary score $z_t \in \{-1, 0, +1\}$; the value $z_t = 0$ encodes *honest uncertainty*: an explicit signal that the agent's prospective belief about $y_t$ from $(s_t, a_t)$ alone is inconclusive. This appendix records our handling of this residual class.

**Treatment as a neutral class.** Rollouts with $z_t = 0$ inherit the gradient invariance of Eq. 2 but do not fit the four-cell calibration taxonomy of Table 1, because the agent has not committed to a directional prediction. The PH update rule (§4.3) assigns weight 1.0 to all such rollouts, matching the weight given to the calibrated cells (CS, AF) so that $z_t = 0$ neither amplifies nor dampens the base loss.

**Tie-breaking under majority vote.** When $M > 1$ samples in Eq. 4.1 produce a strict majority for one of $\{-1, +1\}$, we use that value. Under a tie, or when all $M$ samples are unparseable (e.g., the model fails to emit a valid `\boxed{·}` field), we default to $z_t = 0$. This conservative rule prevents PH from amplifying ambiguous rollouts and folds parsing failures into the neutral class.

**Empirical frequency.** Across our main experiments, $z_t = 0$ accounts for a small minority of rollouts in both single-step and multi-turn settings, with the exact rate varying by task and model. The frequency is largely independent of the base method (GRPO, OPD, Combined) and decreases as $M$ increases in the majority vote, since aggregating over multiple samples collapses borderline cases toward the dominant prediction.

## D Additional Experimental Setup

This appendix expands §5.1 with all information needed to reproduce our experiments. §D.1 covers the single-turn verifiable tasks (Science Q&A and Tool Use) built on top of the SDPO [14] backbone, and §D.2 covers the multi-turn personal agent built on top of the OpenClaw-RL [42] framework. §D.4 summarizes compute, software environment, and code release. The two regimes differ along one important axis — *single-turn uses a rule-based binary verifier with no learned reward model, while multi-turn uses an OLMo-3-7B-Instruct PRM judge* — which explains the configuration differences below.

### D.1 Single-Turn Verifiable Tasks

**Setting in one sentence.** Each rollout is a single prompt $x$ and a single model response $a$, with a rule-based verifier $R(x, a) \in \{0, 1\}$ that returns binary correctness and no auxiliary reward model; this is the simplest possible instantiation of PH (single prediction, single outcome, $T$=1, deterministic retrospective signal).

**Datasets and verifiers.** We use the Chemistry subset of SCIKNOWEVAL [9] following the released CaOPD [50] preprocessing. Each prompt asks the model to reason step-by-step and emit a single letter $\hat{y} \in \{A, B, C, D\}$; the verifier is rule-based, $R(x, a)$=1 iff the parsed letter equals the gold answer. We also use the Tool Use split derived from TOOLALPACA [38] as preprocessed in CaOPD; each prompt provides API documentation in a ReAct-style template and the verifier checks (a) the action name matches the gold tool and (b) the JSON arguments match the gold schema under canonical normalization. Both training sets contain 4–5k prompts; we evaluate on the corresponding test split.

Together they probe both knowledge-intensive reasoning (Science Q&A) and structured agentic prediction (Tool Use), and were the two domains used in CaOPD, which lets our calibration numbers be directly compared against theirs.

**Backbone OPD method (SDPO) and privileged-context construction.** We build the single-turn experiments on top of SDPO [14], using the official codebase as modified in CaOPD [50]. SDPO is an on-policy self-distillation method: the same model serves as both student $\pi_\theta(\cdot|x)$ and teacher $\pi_\theta(\cdot|x, z)$, where $z$ is a privileged context not available at deployment. We use the SDPO "correct-solution" mode of CaOPD: for each prompt, after the rollout phase, one verified-correct rollout $a^\star$ is chosen on-the-fly and inserted as $z$, so the teacher is then asked to "Correctly solve the original question" conditioned on $(x, z)$. The training loss is per-token reverse KL between student and teacher on the teacher trajectory, with rollout importance sampling correction and SDPO defaults (top-$K$=100 distillation, EMA teacher update rate 0.05, IS clip 2); thinking mode is disabled (`enable_thinking=false`) so that the trajectory we distill is the answer trajectory.

**Models, training framework, and hardware.** Backbones are **OLMo-3-7B-Instruct** [29] (primary, both Science Q&A and Tool Use) and **gpt-oss-20B** [2] (scale ablation). Both are used as student and EMA-teacher (the EMA teacher is the same model with the privileged context $z$). We deliberately keep the verbalized-confidence template of CaOPD intact so that PH plugs in cleanly: each rollout still ends with a parseable `Confidence: {x}` token, but PH ignores this string and uses only the binary verifier outcome $R(x, a)$ as the retrospective signal. The codebase forks CaOPD's SDPO branch, implemented on top of **verl** [35] with PyTorch FSDP2 for sharded optimizer state and **vLLM** [20] for batched rollout generation; our PH plug-in adds (i) a per-prompt prospective prediction call, (ii) a stop-gradient weight $w(\xi_t)$ on the per-rollout reverse-KL contribution, and (iii) lightweight surprise/OFR bookkeeping in the training logger. Each run uses one node of `p5en.48xlarge` (8× NVIDIA H200), which is sufficient for both 7B and 20B-MoE training under FSDP2.

**Prospective Hindsight plug-in.** For each rollout, after the student emits its answer $a$ but before the verifier evaluates $R(x, a)$, we issue an additional generation call to the same student model with a different prompt that strips the verbalized-confidence instruction and asks the model to predict whether its previous answer is correct, incorrect, or uncertain. The exact predictor template is given in App. F. Because the verifier here is rule-based and deterministic, a single ternary prediction $z_t \in \{-1, 0, +1\}$ per rollout already gives a low-variance signal, so we use $M$=1 predictor sample (no majority vote). Predictor generation is capped at `max_new_tokens`= 128 tokens at temperature 0.6, top-$p$=0.95; un-parseable outputs are treated as $z_t$=0 (uncertain). The surprise indicator and weight are

$$\xi_t = \mathbf{1}\big[(z_t, y_t) \in \{\text{OF}, \text{US}\}\big], \qquad w_t = 1 + \alpha\xi_t,$$

with $\alpha$=1.0 default and clip $w_t \in [0.5, 2.0]$ as a safety net (the unclipped weight is in $\{1, 2\}$ already, so the clip is rarely active). The weight is a stop-gradient scalar and multiplies the SDPO per-rollout reverse-KL loss; no gradient flows back through the predictor.

**Hyperparameters.** Table 5 reports the hyperparameters for all single-turn runs. All values are taken either from the CaOPD defaults or from a one-shot tuning on a held-out validation slice; no hyperparameter is tuned per-method (the same configuration is used for SDPO, SDPO + PH, and the $\alpha$-sweep ablation).

**Evaluation protocol and metrics.** Capability is reported as `mean@16`: for each held-out test prompt we sample $K_{\text{val}}$=16 rollouts at temperature 0.6, top-$p$=0.95, evaluate the verifier on each, and report the mean accuracy. From the same $K_{\text{val}}$=16 rollouts we compute, per prompt and then averaged across the test set, (i) the surprise rate $\widehat{\mathcal{M}} = \Pr[\xi_t{=}1]$, (ii) the overconfident-failure rate $\Pr[y_t{=}0 \mid z_t{=}{+}1]$, and (iii) the honest-uncertainty rate $\Pr[z_t{=}0]$ (App. C.4, where we verify that the neutral cell does not collapse under PH). All headline numbers are mean±std over 3 seeds, reported at the end of epoch 3; the $\alpha$-sweep additionally logs per-step trajectories of mean@16 and OFR (App. E.1).

### D.2 Multi-Turn Personal Agent

**Setting in one sentence.** A trained student model serves as an OpenAI-compatible chat agent inside the OpenClaw runtime; an external LLM "user" (GPT-4.1) drives multi-turn math-tutoring

conversations grounded in GSM8K, an OLMo-3-7B-Instruct PRM judges each main-line turn, and PH attaches a per-turn surprise signal to the OpenClaw-RL training loop.

**Task and agent design.** We follow the personal-agent track of OpenClaw-RL [42] verbatim. At each training round we alternate between two scenarios: (i) a *Student* scenario in which the simulated user is a "lazy student" who needs the agent to solve a math problem in a non-AI-style tone (no bold, no bullet points); the student writes the problem to `homework/i.txt`, asks the agent to solve, critiques the style, and ends with the literal sentinel `DONE` once satisfied; and (ii) a *Teacher* scenario in which the simulated user is a strict math teacher who wants warm, specific, prose-only grading comments. The agent has access to the OpenClaw runtime's standard file-I/O tools but only text generation is exercised in this benchmark. Each session is one (problem, scenario) pair with up to $T_{\max}{=}48$ turns, ending whenever the simulator emits `DONE`; turns inside a session are tagged `turn_type=main` (used for training) or `turn_type=side` (eval-only). We split GSM8K into disjoint training and evaluation pools (offsets $36{:}\infty$ training; $0{:}36$ student-eval, $0{:}24$ teacher-eval), with `sessions_per_round`$=2$ and 16 training rounds.

**User simulators and conversation protocol.** The simulators are GPT-4.1 calls (temperature $0$) with the system prompts taken verbatim from the OpenClaw-RL evaluation script; the full strings are reproduced in App. F. The salient content of the *Student* prompt is: "You are role-playing as a lazy student who knows nothing… If the AI's answer has stuff like bold text, numbered lists, **Final answer:**, or anything too AI-like, tell it to rewrite… After the AI shows you a satisfactory version, say exactly: `DONE`." The *Teacher* prompt asks for warm, specific, prose-only grading comments and ends similarly with `DONE`. The first user message in each scenario is templated from the GSM8K problem and (for the Teacher scenario) the gold answer.

**Base method: Combined (Binary RL + OPD).** The default base method we plug PH into is the **Combined** recipe of OpenClaw-RL [42]; we also evaluate Binary RL only and OPD only as ablations. For each main-line turn the PRM judge produces a scalar score $r_t \in \{-1, 0, +1\}$ from the next-state signal (the simulator's reply); GRPO advantages are computed within the trajectory with mean and standard deviation across the same session, with `disable_rewards_normalization` so that the per-turn scale of $r_t$ is preserved. The OPD component uses the same model conditioned on a textual hint extracted from the next-state signal; the teacher's per-token log-probability gap with the student forms the token-level directional advantage $A^{\text{OPD}}_{t,j}$ (EMA teacher tracking rate $0.05$). The combined per-turn advantage is $A^{\text{comb}}_t = w_{\text{rl}} A^{\text{GRPO}}_t + w_{\text{opd}} A^{\text{OPD}}_t$ with $w_{\text{rl}}{=}w_{\text{opd}}{=}1$; the PPO surrogate uses asymmetric clip $\varepsilon_{\text{lo}}{=}0.2$, $\varepsilon_{\text{hi}}{=}0.28$ from GRPO [33] and `kl_loss_coef`$= 0$.

**PRM judge and OPD hindsight teacher.** Both the PRM and the OPD teacher share weights with the student (full self-distillation): they all run OLMo-3-7B-Instruct, served by separate sglang engines (separate from the rollout engine) so they do not block training. We use `prm_num_gpus=1`, `prm_teacher_num_gpus=1`, `prm_m=1` (PRM majority vote; we ablate `prm_m=3`), `prm_temperature=0.6`, `prm_max_new_tokens=2048`, and the OPD teacher source is `megatron` so that the teacher distribution is on-policy with the actor.

**Training framework: Slime + Megatron-LM.** We use the asynchronous 4-component training loop of OpenClaw-RL [42], implemented on `slime` (RL orchestrator) and Megatron-LM (training backend), with `sglang` serving rollouts/PRM/teacher. The 8 GPUs of one `p5en.48xlarge` node split as: 4 GPUs for the actor (TP=4, sequence-parallel, bf16); 2 GPUs for rollout (one engine per GPU, TP=1); 1 GPU for the PRM judge; 1 GPU for the OPD teacher. PH inserts a per-turn predictor call between the actor's response and the PRM's score; in the asynchronous regime the predictor runs concurrently with PRM scoring (neither reads the next-state signal), so PH adds essentially zero wall-clock overhead in steady state. Each session writes a JSONL record of (`session_id`, `turn_index`, `turn_type`, `prm_score`, `prospective_score`, `surprise`, `weight_version`) to disk, allowing offline reconstruction of all calibration metrics.

**Prospective Hindsight plug-in.** For each main-line turn $t$, after the agent emits its response $a_t$ but before the PRM scores it, we issue an additional generation request to the same actor with a prompt that asks the model to predict whether its own response in the current dialog state will be judged correct/incorrect/uncertain by an external judge. The predictor sees $(s_{1:t}, a_t)$ but never the

simulator's reply $s_{t+1}$. *Because the per-turn PRM signal is itself an LLM-based score and therefore stochastic, we use $M$=3 predictor samples, parsed to $\{-1, 0, +1\}$ and averaged into a continuous* $\bar{z}_t \in [-1, +1]$. We then use the continuous PH gate (App. C.3):

$$\xi_t \;=\; \tfrac{1}{2}\big|\bar{z}_t - r_t\big| \;\in [0,1], \qquad w_t \;=\; 1 + \alpha\xi_t,$$

where $r_t \in \{-1, 0, +1\}$ is the PRM-aggregated turn score. The implementation flag is `SURPRISE_MODE=continuous`; setting `SURPRISE_MODE=discrete` recovers the binary $\xi_t \in \{0, 1\}$ used in single-turn experiments. The weight applies to the per-turn combined advantage $A_t^{\text{comb}}$ before the PPO surrogate (equivalently, broadcast across all tokens of $a_t$ since the surprise is a turn-level scalar) and is a stop-gradient constant. PH-specific knobs: `PROSPECTIVE_M=3`, `PROSPECTIVE_MAX_TOKENS=256`, `SURPRISE_ALPHA=1.0` default, no weight clip (`SURPRISE_WEIGHT_MAX` unset; the gate already gives $w_t \in [1, 1+\alpha]$).

**Hyperparameters.** Table 4 reports the hyperparameters for all multi-turn runs. The same configuration is used for the Combined baseline, Combined + PH, and the $\alpha$-sweep ablation; only the PH-specific rows differ across rows of Table 3.

**Evaluation protocol and metrics.** We use the personalization evaluator of OpenClaw-RL [42] (script `personalization_evaluator.py`). For each held-out evaluation prompt we collect a single first response (turn 1, no follow-up) at temperature 0.3, and score it with GPT-4.1 against a fixed preference template that requests a boxed score from $\{0, 0.25, 0.5, 0.75, 1\}$ at temperature 0 with $N_{\text{votes}}$=3 parallel votes whose successful boxed scores are averaged. The user-side preference text is one of `PREFERENCE_STUDENT` (no AI-style formatting; full reasoning shown) or `PREFERENCE_TEACHER` (warm, specific, prose-only grading comments); the full strings are in App. F. Capability is the mean evaluator score across the eval pool, reported separately for Student ($N$=36) and Teacher ($N$=24) plus the macro-average. We define multi-turn OFR as the per-turn analogue $\Pr[r_t \le 0 \mid z_t{=}{+}1]$, computed from the per-turn JSONL bookkeeping over the held-out eval pool, and additionally report surprise rate $\widehat{\mathcal{M}} = \Pr[\xi_t > 0]$ and prospective accuracy. Headline numbers are reported at the final checkpoint of round 16; dynamics plots evaluate every 4 weight updates.

### D.3 Reweighting-Baseline Protocols

To isolate the contribution of PH's surprise-based selection from the contribution of additional gradient mass, we compare PH against two reweighting controls in the single-turn setting (Table 2). Both use the same multiplier $(1{+}\alpha)$ as PH and are detached from the computation graph (stop-gradient); only the rule for *which* rollouts receive the multiplier differs.

**Random reweighting.** At each training step, we compute the empirical fraction $p_t$ of surprising rollouts the PH protocol produces on the same data. We then sample uniformly at random a subset of size $\lfloor p_t \cdot N \rfloor$ from the current batch and assign these rollouts weight $(1{+}\alpha)$; the remaining rollouts retain weight 1. By construction this matches PH's per-step total weight $\sum_i (1 + \alpha \cdot \xi_t^{(i)})$, so the only difference between Random and PH is the *location* of the upweighted mass: uniform vs. surprise-targeted. Selection seeds are derived from `hash(rollout_id, step)` so that within an epoch each rollout receives a deterministic weight; we use $\alpha$=1.0 throughout, matching the headline PH configuration.

**Failure-only reweighting.** For every rollout with verifier outcome $y_t$=0, we set the weight to $(1{+}\alpha)$; rollouts with $y_t$=1 retain weight 1. This corresponds to upweighting cells OF $\cup$ AF in the calibration taxonomy (Table 1), as opposed to PH's OF $\cup$ US. We again use $\alpha$=1.0. Failure-only therefore differs from PH in exactly two cells: it includes AF (calibrated incompetence, which PH leaves unweighted) and excludes US (underconfident successes, which PH amplifies).

**What each baseline controls for.** Random isolates the effect of *selection*: any gain of PH over Random must come from *which* rollouts are upweighted, not from how much weight is added. Failure-only matches PH on the dominant single-turn cell (OF; see Figure 5) but differs on the smaller cells, isolating whether the prospective signal $z_t$ contributes information beyond the verifier outcome $y_t$ alone. The empirical comparison is reported in Table 2 and discussed in §5.2; the PH-cell composition over training is shown in Figure 2.

### D.4 Compute and Reproducibility

**Hardware and software.** Both single-turn and multi-turn experiments run on a single node of `p5en.48xlarge` (8× NVIDIA H200). Multi-node is supported by the underlying frameworks (verl / Slime) but not used in our reported runs. Software: `CUDA 12.x`, `PyTorch 2.5+`, `vLLM` for single-turn rollouts, `sglang` for multi-turn rollouts/PRM/teacher, `Megatron-LM` (multi-turn actor backend), `slime` (multi-turn async orchestrator), `verl` (single-turn orchestrator).

**Wall-clock and overhead from PH.** *Single-turn* (3 epochs, 32 prompt batch, $K$=8 rollouts): $\sim$ 8 hours per run on Science Q&A, $\sim$ 6 hours on Tool Use; the PH plug-in adds $\sim$ 6% wall-clock overhead from the predictor calls (amortized into the same vLLM batch). *Multi-turn* (16 rounds, 2 sessions/round, $T_{\max}$=48): $\sim$ 4 hours per run; the PH plug-in adds essentially zero wall-clock overhead because the predictor runs concurrently with the PRM in the asynchronous loop.

## E Additional Experiment Results and Analysis

This appendix expands the experiments of §5.2 (single-turn SDPO + PH on Science Q&A and Tool Use) and §5.3 (multi-turn RL / OPD / Combined + PH) with the full $\alpha$-sweeps that the main tables abridge, finer-grained training-time dynamics, and per-turn breakdowns. All configurations match App. D unless stated otherwise.

### E.1 Single-Turn $\alpha$-Sweep: SDPO + PH on Science Q&A

**PH cell composition is qualitatively robust across $\alpha$.** Figure 6 repeats the per-step cell-composition plot of Figure 2 (which fixes $\alpha$=1.0) for the additional weights $\alpha \in \{0.1, 0.25, 0.5, 2.0\}$ on Science Q&A. The four panels share a common pattern: the overconfident-failure (OF) cell shrinks throughout training while the confident-success (CS) cell grows, and the underconfident-success (US) and aware-failure (AF) cells stay small. The fact that this dynamics holds essentially unchanged across two orders of magnitude of $\alpha$ tells us that PH is not a delicate optimization-noise instrument: it is a structural reweighting whose direction (compress OF, grow CS) is set by the data, with $\alpha$ controlling only the speed of that compression.

**OFR decreases monotonically and the PH weight self-extinguishes.** Figure 7 (left) plots OFR over training for each $\alpha$ on Science Q&A. All five trajectories descend monotonically; $\alpha$=0.5 reaches one of the lowest late-stage OFR values, consistent with the main-text observation in §5.2. Figure 7 (right) plots the corresponding mean PH weight $\mathbb{E}[w_t]$: it starts above 1 (because $\xi_t > 0$ on many rollouts initially), decays toward 1 as $\xi_t \to 0$ on the bulk of rollouts, and the larger the $\alpha$ the higher the initial weight but also the steeper the relaxation. The weight curriculum is therefore self-extinguishing — exactly the behaviour predicted by the continuous-relaxation analysis of App. C.3.

**Why $\alpha$=0.5 is the Science Q&A sweet spot.** Table 6 numerically compares the five $\alpha$ values along three axes: late-training OFR, the early-to-late OFR drop, and the verifier pass rate at the final checkpoint. $\alpha$=0.5 is best on all three: largest absolute OFR drop (0.560), largest relative drop (72.9%), and highest final verifier pass rate (73.2%). The weight column shows why: $\alpha$=0.5's mean weight starts at 1.35 (large enough to materially reweight miscalibrated samples) and anneals to 1.09 (small enough to not perturb late-stage learning), giving the cleanest curriculum. $\alpha$=1.0 and $\alpha$=2.0 start the curriculum from much higher initial weights (1.67, 1.68) and do not anneal as cleanly, trading a fraction of the calibration improvement for slightly noisier capability. Conversely, $\alpha \in \{0.1, 0.25\}$ are nearly identity reweightings throughout training (initial weight 1.07, 1.17) and so leave the OFR drop close to the small-$\alpha$ floor of $\approx$ 0.49.

### E.2 Multi-Turn $\alpha$-Sweep: Per-Round Capability and Per-Turn Calibration

**$\alpha$ sweep across capability and per-turn calibration.** Figure 8 plots the multi-turn $\alpha$-sweep on the Combined base method under the GPT-4.1 simulator: teacher-side score across training rounds (left), per-turn surprise rate across conversation turns (middle), and per-turn OFR across conversation turns (right). The capability trajectory (left) reproduces the headline of Table 3: $\alpha$=1.0 rises fastest and plateaus highest, $\alpha$=0.5 tracks closely, and $\alpha$=2.0 exhibits the highest round-to-round variance.

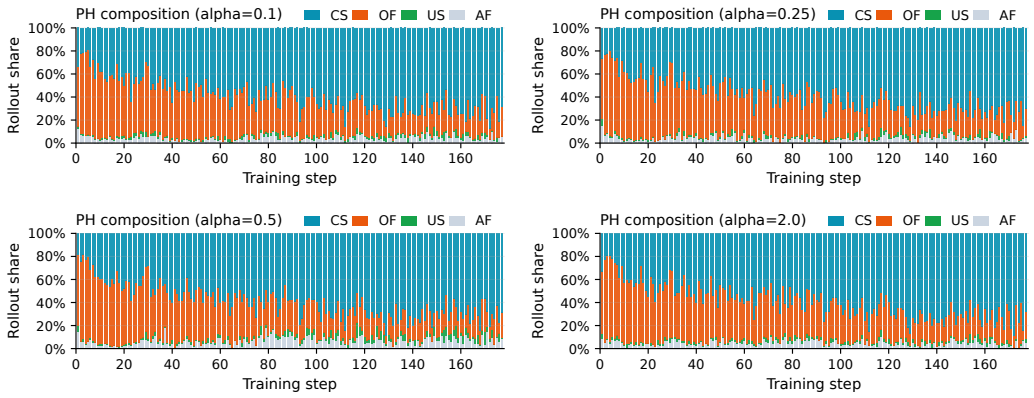



Figure 6: **PH cell composition across $\alpha$ on Chemistry.** Per-step decomposition of training rollouts into CS / OF / US / AF for $\alpha \in \{0.1, 0.25, 0.5, 2.0\}$ (the $\alpha$=1.0 panel is in Figure 2). The compress-OF / grow-CS pattern is qualitatively unchanged across two orders of magnitude of $\alpha$.

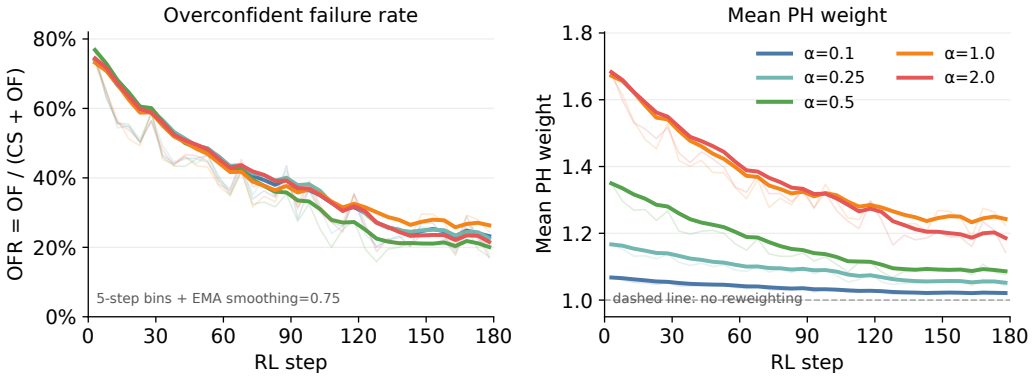



Figure 7: **OFR and PH-weight dynamics across $\alpha$ on Science Q&A.** Left: OFR over training for each $\alpha$; all five trajectories descend monotonically. Right: the mean PH weight starts above 1 and anneals toward 1 as $\xi_t \to 0$ (App. C.3). Faint lines: 5-step bins; solid lines: EMAs.

The per-turn surprise (middle) and per-turn OFR (right) reveal a structural pattern that the trajectory-averaged metrics in the main text do not surface: *later turns have higher initial overconfidence*, consistent with the long-horizon credit-assignment intuition that the agent's self-model is least developed for late, context-saturated turns; and PH compresses exactly this region most. At $\alpha$=1.0, late-turn OFR drops to levels comparable to the early-turn OFR of the baseline, effectively flattening the per-turn calibration profile.

**When $\alpha$=2.0 overshoots in long horizons.** Figure 8 also concretizes the $\alpha$=2.0 caveat from §5.3: the right panel shows that $\alpha$=2.0 achieves the lowest per-turn OFR overall, but the left panel shows that its capability oscillates round-to-round and ends below $\alpha$=1.0. The mechanism is the variance-amplification effect quantified in App. B.2: very large $\alpha$ multiplies the per-rollout combined-advantage by up to $1 + \alpha = 3$ on miscalibrated rollouts, and in long-horizon training those rollouts also tend to be the longest token sequences (overconfident failures are correlated with more aggressive generation), so the per-batch gradient norm spikes relative to the baseline distribution. We did not observe this regime in single-turn (Table 2) because the per-rollout token count there is bounded and similar across all rollouts; the multi-turn regime, with its variable-length sessions of up to $T_{\max}$=48 turns (App. D.2), exposes it.

**Practitioner recommendation.** For both regimes, a single $\alpha$=1.0 default works well across methods, datasets, and simulators. The per-domain optimum may shift to $\alpha$=0.5 (Science Q&A, Combined) or remain at $\alpha$=1.0 (Tool Use, RL), but no $\alpha$ outside the $\{0.5, 1.0\}$ band ever beats the

in-band defaults on *both* capability and calibration simultaneously. We therefore recommend $\alpha$=1.0 as the practitioner default and $\alpha$=0.5 as a slightly more conservative alternative whenever capability stability is paramount.

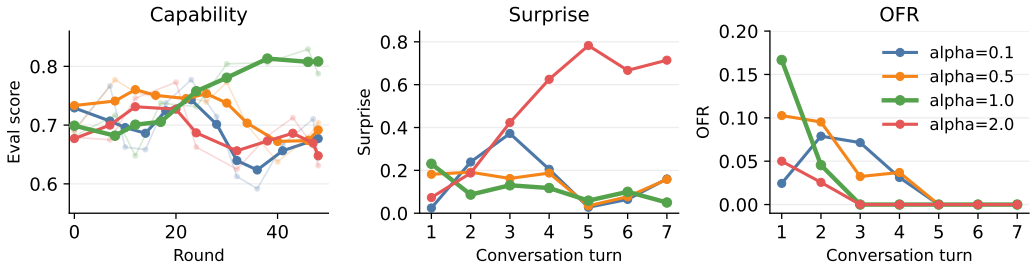



Figure 8: **Multi-turn $\alpha$-sweep (Combined, GPT-4.1 simulator).** Left: teacher-side evaluation score across training rounds. Middle: teacher-side surprise rate across conversation turns. Right: teacher-side OFR across conversation turns. Curves: $\alpha \in \{0.1, 0.5, 1.0, 2.0\}$. Later turns start more overconfident than early turns, and PH compresses this region most.

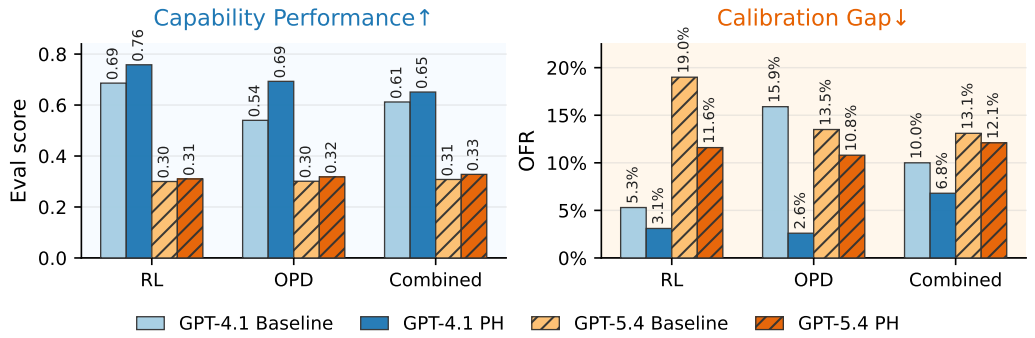



Figure 9: **Capability and calibration across user simulators.** OLMo-3-7B-Instruct trained on the multi-turn personal-agent task with 8k response-length budget and mixed Student/Teacher rounds, under two simulators (GPT-4.1 and GPT-5.4). Left: teacher-side personalization-evaluator score (higher is better). Right: teacher-side OFR (lower is better).

# F Prompt Templates

This appendix lists the prompt templates used by PH's prospective self-evaluator (§4.1) and, in multi-turn settings, the retrospective process reward model (PRM) used as the verifier $R$ in Eq. 1. All prompts elicit a ternary score parsed from a `\boxed{·}` field as $z_t$ or $r_t \in \{-1, 0, +1\}$; ties under majority voting are broken toward 0.

## F.1 Single-Step Tasks

For single-step tasks (Science Q&A, Tool-Use, etc.), the prospective self-evaluator uses a task-type-specific template that wraps a common ternary scoring instruction. The task type is routed automatically from the dataset identifier: math (GSM8K, MATH, AIME, etc.) → `math`; multiple-choice (SciKnowEval, GPQA, MMLU) → `mcq`; tool-use → `tool_use`; code (LiveCodeBench, HumanEval+, etc.) → `code`; otherwise → `default`. Placeholders `{problem}` and `{solution}` are filled with the task instance and the agent's just-emitted action $a_t$.

The common ternary scoring instruction below is appended verbatim to every single-step template (Listing 1); each task type then wraps it with its own header (Listings 2–6).

```
Return exactly one score and no explanation:
+1 if you are confident it is correct
```

```
-1 if you believe it is wrong
0 if you are unsure
Your entire response must be exactly one of: +1, 0, -1
```

Listing 1: Common ternary scoring instruction (single-step).

```
Review your solution to the following problem.

Problem:
{problem}

Your solution:
{solution}

Do you believe your final answer is correct?
[common scoring instruction]
```

Listing 2: Single-step prospective self-evaluator: Math template.

```
Review the question and your chosen answer below.

Question:
{problem}

Your response:
{solution}

Do you believe you selected the correct answer?
[common scoring instruction]
```

Listing 3: Single-step prospective self-evaluator: Multiple-choice template.

```
Review the task and your proposed solution below.

Task:
{problem}

Your solution:
{solution}

Do you believe you selected the correct actions and inputs to
complete this task?
[common scoring instruction]
```

Listing 4: Single-step prospective self-evaluator: Tool-use template.

```
Review the problem and your code solution below.

Problem:
{problem}

Your solution:
{solution}

Do you believe your code produces the correct output for all
test cases?
[common scoring instruction]
```

Listing 5: Single-step prospective self-evaluator: Code template.

```
Review your solution to the following problem.
```

```

Problem:
{problem}

Your solution:
{solution}

Do you believe your solution is correct?
[common scoring instruction]
```

Listing 6: Single-step prospective self-evaluator: Default template (fallback).

### F.2 Multi-Turn Tasks

In multi-turn settings, two distinct prompts are used: a *prospective* prompt that the agent's policy answers immediately after emitting $a_t$ (without observing the next state $c_t = s_{t+1}$), and a *retrospective* PRM prompt that scores the same response after $s_{t+1}$ is revealed. The retrospective PRM serves as the verifier $R$ in Eq. 1; the prospective prompt produces $z_t$ as in Eq. 4.1. Placeholders are filled at runtime: `{response_text}` is the agent's action $a_t$, `{last_user_message}` is the most recent user turn from the conversation history, `{next_state_text}` is $s_{t+1}$, and `{next_state_role}` indicates whether the next state is a user reply or a tool return.

**Prospective self-evaluator (general multi-turn).** This is the default prospective prompt used by PH at every turn $t$. It receives only $(s_t, a_t)$ via `{last_user_message}` and `{response_text}`; the next state $s_{t+1}$ is not exposed.

```
User request:
{last_user_message}

Assistant response:
{response_text}

Does this response correctly and completely address the user's
request?
Answer \boxed{1} for yes, \boxed{-1} for no, \boxed{0} for unsure.
Be brief. End your answer with \boxed{SCORE}.
```

Listing 7: Multi-turn prospective self-evaluator (general).

**Prospective self-evaluator (Personal Agent, style-aware).** This variant is used in the Personal Agent setting where user satisfaction depends on stylistic conformance (natural prose, no AI-style formatting) in addition to factual correctness. The prompt is split into a system message that defines the user's style preferences and a user message that asks for the prediction.

```
[SYSTEM]
You are predicting whether a user will be SATISFIED or
DISSATISFIED with an AI assistant's response, based on the
user's strong style preferences.

The user HATES AI-style formatting. Specifically, the user
will ask for a rewrite if the response contains ANY of these:
- Markdown bold (**text**), headers (## or ###), or horizontal
  rules
- Numbered lists (1. 2. 3.) or bullet points (- or *)
- Overly structured step-by-step layouts
- Phrases like "Here is", "Sure!", "Great question!", "Let me",
  "Certainly"
- Emojis or excessive exclamation marks

The user WANTS natural, human-like prose, as if a real person
(student or teacher) wrote it casually. Flowing sentences, no
```

```
formatting artifacts.

Scoring:
- \boxed{-1}: response has formatting issues or sounds AI-
  generated; user will request a rewrite
- \boxed{1}: response is natural, human-like prose with no
  formatting artifacts; user will be satisfied
- \boxed{0}: borderline / unsure

Be strict. Even one instance of bold text or a numbered list
means \boxed{-1}.

[USER]
User message:
{last_user_message}

Assistant response:
{response_text}

Check the response for AI-style formatting (bold, headers,
numbered lists, bullet points, AI phrases). Does it read like
natural human writing?
Answer with \boxed{SCORE}.
```

Listing 8: Multi-turn prospective self-evaluator (Personal Agent, style-aware).

**Retrospective PRM (verifier).** The PRM observes the next state $s_{t+1}$ as evidence and assigns the outcome $r_t \in \{-1, 0, +1\}$.

```
[SYSTEM]
You are a process reward model (PRM) evaluating an AI
assistant. You will see the assistant's output and the
subsequent next state. Your task: decide whether the
assistant's output successfully fulfilled the user's intent
at that step, using the next state as evidence.

Understanding the next state's role:
- role='user': a reply from the user.
- role='tool': the return value of a tool the assistant
  invoked. This content was NOT available before the
  assistant's action; it exists BECAUSE the assistant called
  the tool. A successful, non-error tool output means the
  assistant's action worked correctly and should be scored
  positively.

Scoring rules:
- \boxed{1}  (good):    next state shows the task progressed
                        as expected (user moves on, says
                        thanks, environment confirms success,
                        tool returns successful non-error
                        result).
- \boxed{-1} (bad):     next state signals the assistant's
                        output was wrong, incomplete, or
                        unwanted. Key negative signals:
                        * user asks the assistant to redo,
                          retry, or repeat the same action
                        * user requests a correction or
                          modification
                        * user rephrases or restates the same
                          request
                        * environment returns an error,
                          failure, or unexpected result.
- \boxed{0}  (neutral): next state is ambiguous.

```

```
A change request is negative feedback: it means the previous
output did not meet the user's need; do not treat it as a
neutral new instruction. Think step-by-step, then give your
final score inside \boxed{}.

[USER]
Assistant output:
{response_text}

Next state [role: {next_state_role}]:
{next_state_text}

First, classify the next state: is it (a) positive
progression, (b) a correction / redo / change request, or
(c) ambiguous? Then assign \boxed{1}, \boxed{-1}, or \boxed{0}.
```

Listing 9: Multi-turn retrospective PRM verifier (next-state grounded).

Table 4: Hyperparameters for multi-turn personal agent: Combined (Binary RL + OPD) and Combined + PH on OpenClaw-RL.

| **Parameter** | **Value** |
|---|---|
| *Models, GPU layout* | |
| Backbone (student / PRM / teacher) | OLMo-3-7B-Instruct (shared) |
| Total GPUs | 8 (1 node) |
| Actor | 4 GPUs, TP=4, sequence-parallel |
| Rollout (sglang) | 2 GPUs, TP=1 each |
| PRM judge (sglang) | 1 GPU |
| OPD teacher (sglang+Megatron) | 1 GPU |
| *Rollout / context* | |
| Rollout batch size | 16 |
| Samples per prompt | 1 (multi-turn rollout = full session) |
| Rollout temperature | 0.6 |
| Max response length / turn | 8192 tokens |
| Max context length | 32768 tokens |
| Max tokens per GPU | 32768 |
| *Conversation* | |
| Sessions per round | 2 |
| Training rounds | 16 |
| Max turns per session $T_{\max}$ | 48 |
| Scenario schedule | alternate Student / Teacher each round |
| Training problems (offsets) | GSM8K $36{:}\infty$ |
| Eval problems (offsets) | 0:36 student, 0:24 teacher |
| *PRM judge / OPD teacher* | |
| PRM majority vote $M_{\text{prm}}$ | 1 |
| PRM temperature | 0.6 |
| PRM max new tokens | 2048 |
| EMA teacher update rate | 0.05 |
| OPD teacher source | Megatron actor checkpoints |
| *Combined advantage / PPO surrogate* | |
| Advantage estimator | GRPO |
| $w_{\text{rl}}$, $w_{\text{opd}}$ | 1.0, 1.0 |
| PPO clip $\varepsilon_{\text{lo}}/\varepsilon_{\text{hi}}$ | 0.2 / 0.28 |
| KL loss coefficient | 0.0 |
| Entropy coefficient | 0.0 |
| *Prospective Hindsight* | |
| Surprise mode | continuous |
| Predictor sample size $M$ | 3 (averaged into $\bar{z}_t$; PRM noise) |
| Predictor max new tokens | 256 |
| Surprise amplification $\alpha$ | 1.0 default; sweep $\{0.5, 1.0, 2.0\}$ |
| Surprise weight clip | off (gate already in $[1, 1+\alpha]$) |
| Predictor gradient | stop-gradient |
| *Optimization* | |
| Optimizer | Adam (Megatron) |
| Learning rate | $1\times10^{-6}$ |
| LR decay style | constant |
| Weight decay | 0.1 |
| Adam $\beta_1, \beta_2$ | 0.9, 0.98 |
| Save interval (in weight-update steps) | 25 |
| Train epochs (data passes) | 1 |

Table 5: Hyperparameters for single-turn SDPO and SDPO + PH on Science Q&A and Tool Use.

| **Parameter** | **Value** |
|---|---|
| *Models, data, and inference* | |
| Backbone models | OLMo-3-7B-Instruct; gpt-oss-20B (ablation) |
| Thinking mode | disabled (`enable_thinking=false`) |
| Max prompt length | 2048 tokens |
| Max response length | 8192 tokens |
| Inference engine | vLLM |
| *Rollout (training)* | |
| Question batch size | 32 |
| Rollouts per question $K_{\text{train}}$ | 8 |
| Rollout temperature | 1.0 |
| *Validation* | |
| Rollouts per question $K_{\text{val}}$ | 16 (`val_kwargs.n`, "mean@16") |
| Validation temperature | 0.6 |
| Validation top-$p$ | 0.95 |
| *SDPO loss (CaOPD defaults)* | |
| Distillation top-$K$ | 100 |
| Distillation divergence | token-level reverse KL |
| EMA teacher update rate | 0.05 |
| Rollout IS clip | 2 |
| *Prospective Hindsight (this work)* | |
| Predictor max new tokens | 128 |
| Predictor temperature / top-$p$ | 0.6 / 0.95 |
| Predictor sample size $M$ | 1 (deterministic verifier; no majority vote) |
| Surprise amplification $\alpha$ | 1.0 default; sweep $\{0.0, 0.25, 0.5, 1.0, 2.0, 5.0\}$ |
| Weight clip $[w_{\min}, w_{\max}]$ | $[0.5, 2.0]$ |
| Predictor gradient | stop-gradient |
| *Optimization* | |
| Optimizer | AdamW |
| Learning rate | $1\times10^{-5}$ |
| LR warmup steps | 10 |
| Weight decay | 0.1 |
| Gradient clip norm | 1.0 |
| PPO mini-batch size | 32 |
| Total epochs | 3 |
| *Hardware* | |
| Hardware | 1× `p5en.48xlarge` (8× H200) |
| Sharding | FSDP2, bf16 |

Table 6: **PH weight strength on Science Q&A: numerical summary across the full $\alpha$-sweep.** OFR $=\mathrm{OF}/(\mathrm{CS}+\mathrm{OF})$; Early/Late are first/last 5 training steps; Mean Weight and Verifier Pass Rate are reported as early $\rightarrow$ late. **Bold**: best per column.

| $\alpha$ | Early OFR | Late OFR | OFR Drop | Rel. Drop | Mean Weight | Verifier Pass Rate |
|---|---|---|---|---|---|---|
| 0.1 | 0.741 | 0.253 | 0.487 | 65.8% | $1.07 \rightarrow 1.02$ | $24.8\% \rightarrow 71.4\%$ |
| 0.25 | 0.739 | 0.250 | 0.489 | 66.2% | $1.17 \rightarrow 1.06$ | $25.6\% \rightarrow 71.2\%$ |
| **0.5** | **0.768** | **0.209** | **0.560** | **72.9%** | $\mathbf{1.35 \rightarrow 1.09}$ | $\mathbf{22.9\% \rightarrow 73.2\%}$ |
| 1.0 | 0.732 | 0.239 | 0.423 | 57.8% | $1.67 \rightarrow 1.29$ | $26.0\% \rightarrow 67.5\%$ |
| 2.0 | 0.744 | 0.249 | 0.495 | 66.5% | $1.68 \rightarrow 1.22$ | $24.9\% \rightarrow 71.7\%$ |

## G Limitations and Future Work

While Prospective Hindsight (PH) provides a robust and simple mechanism for self-calibration in agent reinforcement learning, we identify several limitations that offer avenues for future work:

- **Variance-Amplification at High $\alpha$:** High values of the surprise amplification hyperparameter ($\alpha \geq 2.0$) can lead to gradient norm spikes, particularly in multi-turn, long-horizon tasks where overconfident failures are correlated with longer token sequences . While $\alpha = 1.0$ is a stable default, extreme upweighting may destabilize training.
- **Theoretical Assumptions:** Our gradient-flow analysis, which shows the PH update is strictly calibration-aligned, relies on a "uniform-base-loss" assumption—specifically, that the base loss is approximately constant on surprising rollouts. In practice, this alignment is quantitative rather than absolute, as the per-rollout loss varies with specific log-probabilities.
- **Training-Time Compute Overhead:** PH requires $M$ additional forward passes per rollout step to elicit the prospective belief. While this adds negligible wall-clock time in asynchronous multi-turn settings (approx. 0%), it introduces a small overhead (approx. 6%) in single-turn synchronous training.
- **Dependence on Self-Model Quality:** The effectiveness of the surprise signal depends on the policy's ability to produce meaningful ternary predictions. Although PH facilitates the co-evolution of the self-model and policy, the initial training phase may provide noisier signals if the base model is extremely poorly calibrated.
- **Scope of Empirical Evaluation:** Our experiments focused on verifiable tasks (Science Q&A, Tool Use) and a math-tutoring personal agent. While we demonstrated cross-scale transfer from 7B to 20B models, the performance of PH in purely creative or subjective generation tasks where retrospective rewards are inherently ambiguous remains to be fully explored.

The limitations above suggest several directions where the Prospective Hindsight principle can be extended or its current evidence base strengthened.

- **Independent characterization of the prospective evaluator.** Decoupling the evaluator's quality from the training trajectory is important both for methodological cleanliness and for understanding when PH should be expected to work. Promising directions include: (i) measuring evaluator accuracy on held-out prompts at fixed checkpoints, (ii) comparing evaluator outputs against a stronger external judge, (iii) calibrating the evaluator's token-level confidence against its boxed score, and (iv) studying the regime in which evaluator and policy share systematic blind spots and how PH behaves under such failure.
- **Scaling to longer-horizon agentic tasks.** Our multi-turn experiments cap conversations at $T_{\max} = 48$ turns. Genuinely long-horizon agents—software engineering trajectories spanning hundreds of tool calls, multi-stage research assistants, or extended pair-programming sessions—raise two questions PH does not yet answer: how does the surprise signal interact with sparse, delayed verifier feedback at long horizons, and does the per-turn calibration cell composition (Fig. 5) shift further as horizons grow? Combining PH with hierarchical credit-assignment schemes that already address sparse-reward long-horizon RL is a natural follow-up.
- **Continuous-valued PRM scores and richer surprise signals.** The current surprise indicator is binary (discrete gate) or a half-distance in $[0, 1]$ (continuous gate). Recent process reward models emit graded scores, confidence intervals, or per-criterion breakdowns [42]. Extending PH to exploit these richer retrospective signals—for example, by defining $\xi_t$ as a function of the PRM's confidence interval on $r_t$ rather than its point estimate—may further reduce the variance amplification at large $\alpha$ that we observe in long horizons.
- **Composition with inference-time self-correction.** PH operates entirely at training time and leaves the agent's deployment behavior unchanged. Inference-time self-correction methods such as Reflexion [36] and Self-Refine [26] act on the policy distribution *after* PH has shaped it. Whether the two mechanisms compose constructively—PH producing a better-calibrated self-model, then inference-time self-correction exploiting that calibration to drive re-tries—is an open question with practical implications for deployed agentic systems.

## H Broader Impact

The development of Prospective Hindsight (PH) aims to bridge the gap between capability and calibration in language model agents. As LLMs are increasingly deployed in autonomous agentic roles—such as personal assistants, tool-use interfaces, and tutoring systems—the societal implications of their reliability become paramount.

**Positive Societal Impacts:**

- **Enhanced Safety and Reliability:** By explicitly penalizing overconfident failures, PH encourages agents to recognize their own limitations. This reduction in "confidently wrong" behavior is critical in high-stakes domains where an agent should ideally invite human intervention rather than proceeding with a flawed trajectory.
- **Improved Human-AI Trust:** Calibration is a cornerstone of effective human-AI collaboration. Agents that can accurately communicate their uncertainty (through honest uncertainty markers) allow users to set appropriate expectations and intervene when necessary.
- **Resource Efficiency:** PH is a self-extinguishing curriculum that reduces the need for manually designed reward schedules or expensive, separate calibration-specific training phases. This can lower the computational and human-labor barriers to developing reliable AI systems.

**Potential Negative Impacts and Mitigations:**

- **Risk of Deceptive Calibration:** While PH aligns the agent's prospective belief with verifier outcomes, there is a theoretical risk that an agent could learn to "appear" calibrated by avoiding difficult but necessary tasks to minimize surprise. We mitigate this by using PH as a plug-in on top of task-success objectives, ensuring capability is not sacrificed for calibration.
- **Over-reliance on Automated Feedback:** The method relies on the quality of retrospective verifiers or PRMs. If these evaluators contain biases or incorrect logic, PH may amplify these errors by aggressively upweighting rollouts that "surprise" the model based on flawed ground truth. Developers should ensure that the underlying verifiers are robust and subject to human auditing.
- **Dual-Use Concerns:** Improved agentic autonomy and tool-use capabilities, while beneficial for productivity, could be misused to automate malicious activities. However, PH's primary contribution is the *alignment* of internal belief with reality, which is a fundamentally defensive property aimed at reducing unintended failures rather than increasing raw harmful capabilities.

## I Licenses for Existing Assets

We utilized several existing datasets, models, and software frameworks in this study. All assets were used in accordance with their respective licenses and terms of use:

- **Datasets:**
  - **Science Q&A (SciKnowEval):** We use the Chemistry subset of SciKnowEval [9], specifically the version preprocessed by CaOPD [50]. The original dataset is released under the **MIT License**.
  - **Tool Use (ToolAlpaca):** This dataset [38] is used as preprocessed in CaOPD [50]. It is available under the **Apache License 2.0**.
  - **GSM8K:** Used for the multi-turn personal agent task. It is released under the **MIT License**.
- **Models:**
  - **OLMo-3-7B-Instruct:** Our primary backbone model [29] is released under the **Apache License 2.0**.
  - **gpt-oss-20B:** Used for scale ablation studies [2]. This model is released under the **Apache License 2.0**.
  - **GPT-4.1 / GPT-5.4:** Used as user simulators and evaluators via the OpenAI API. Usage adheres to OpenAI's **Terms of Service**.
- **Software and Frameworks:**

- **OpenClaw-RL:** The multi-turn training framework is used under its provided open-source license (**Apache License 2.0**).
- **verl:** The single-turn RL orchestrator [35] is released under the **Apache License 2.0**.
- **vLLM & sglang:** Used for high-throughput inference [20]. Both frameworks are released under the **Apache License 2.0**.
- **Megatron-LM:** The training backend for multi-turn experiments is released under the **BSD 3-Clause License**.

# J Declaration of LLM Usage

In accordance with the NeurIPS 2026 guidelines, we declare that Large Language Models (LLMs) serve as an original and non-standard component of the core methodology in this research. Their usage is integral to the training loop and experimental design, extending beyond simple writing or formatting assistance:

- **Methodological Core (Prospective Self-Evaluation):** The primary contribution of this work, Prospective Hindsight (PH), involves re-querying the model policy ($\pi_\theta$) during the training process to produce a ternary prospective self-evaluation $z_t \in \{-1, 0, +1\}$ regarding the anticipated success of its just-emitted action. This self-generated signal is used to reweight the Reinforcement Learning gradient.
- **Step-Level Process Supervision:** In our multi-turn experimental regime, an LLM (specifically OLMo-3-7B-Instruct) serves as a Process Reward Model (PRM) judge. It evaluates each turn of a conversation based on the subsequent environmental state to provide a grounded outcome signal $r_t$.
- **Automated User Simulation:** We utilize LLMs (GPT-4.1 and GPT-5.4) as user simulators to drive complex, multi-turn tutoring and personal-agent conversations. These simulators provide the dynamic environmental states necessary for training and evaluating agent reliability in long-horizon settings.
- **Scientific Rigor and Reproducibility:** To ensure the transparency of these LLM-based components, we have provided comprehensive prompt templates for all self-evaluation and verification tasks—including specific instructions for math, multiple-choice, tool-use, and code-generation domains—in App. F.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper formally defines the "Hindsight Trap" and introduces Prospective Hindsight (PH), a plug-in method that utilizes the "prediction-reality gap" as a training signal. These claims are supported by experimental gains across three families of RL methods: GRPO, OPD, and their combination.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Check Appendix G. The paper discusses technical and optimization limitations, such as the variance-amplification risks of high $\alpha$ values in long-horizon tasks and the reliance on the "uniform-loss" assumption for theoretical guarantees. These discussions are primarily detailed in the gradient-flow analysis and multi-turn sweep results in Appendices B.2 and E.2.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper provides formal statements and proofs for all three of its primary propositions. Proposition 1, regarding the privileged-information gap, is proved in Appendix B.1 using the law of total variance. Propositions 2 and 3, which establish the surprise-weighted decomposition and the miscalibration-rate lower bound, are formally proved in Appendix B.2. Additionally, the paper includes a detailed gradient-flow analysis under specified "uniform-base-loss" assumptions to characterize the calibration-aligned update direction.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides a step-by-step training algorithm, comprehensive hyperparameter tables for all experiments, and full prompt templates for self-evaluation and verification. These details, along with the software environment specifications and the release of code and checkpoints, are documented across Sections 3.1, 3.3, and Appendices C, D, and F.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper states that all training details, hyperparameters and prompt templates. Additionally, Appendix D provides specific details on the software environment, hardware, and dataset preprocessing required to reproduce the results. The specific source code and reproduction scripts are not provided in the current submission to maintain anonymity and will be released upon acceptance.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides exhaustive experimental details, including comprehensive hyperparameter tables for all tasks, specific optimizer configurations (Adam/AdamW), and exact data split offsets for both single-turn and multi-turn regimes. These settings are documented in the experimental setup and further detailed in Appendix D, which specifies how parameters were chosen based on prior defaults or validation tuning. All necessary training and test information, including hardware and software environments, is fully disclosed in Appendix D and summarized in Tables 4 and 5.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All headline numbers are reported as the mean ± standard deviation aggregated over 3 seeds, while training figures include shaded 95% confidence intervals. These protocols and the calculation methods for error bars and shaded regions are explicitly detailed in the evaluation and reporting sections of the appendix. The use of these statistical measures to represent variability across independent initializations is documented in Sections 4.1 and Appendix D.4.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper specifies the exact hardware used for all experiments, noting that each run was conducted on a single node of p5en.48xlarge containing 8x NVIDIA H200 GPUs with 80 GB of HBM3 memory. Detailed execution times are provided for both regimes, citing approximately 6–8 hours per run for single-turn tasks and 4 hours per run for multi-turn sessions. This comprehensive breakdown of compute workers, memory, and wall-clock overhead is primarily documented in Section 4.1 and further expanded in Appendix D.4.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics, as the study utilizes open-source models and established, publicly available datasets for its experiments. All experiments were conducted in controlled, simulated environments, such as rule-based verifiers and LLM-based simulators, without the involvement of human subjects. Furthermore, the authors have maintained anonymity by preparing an anonymized URL for code and checkpoint release.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We illustrate it in Appendix H.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release any new high-risk datasets or pre-trained base models.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We illustrate in Appendix I.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: the paper does not release new assets.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We describe the use of LLMs in Appendix J.