CONSUMPTION IS UNNECESSARY: LEARNING A SELF-ASSESSED VALUE ESTIMATE IS ENOUGH FOR GROUP-RELATIVE AGENT RL

Anonymous authors
Paper under double-blind review

ABSTRACT

A common pattern in training LLM agents is to let the policy estimate its own progress or its chance of success and then use that estimate — as an advantage, a reward, or a rule for choosing actions. Whether the gain comes from learning the estimate or from using it has not been separated. We separate the two. Our method, RAISE, trains the estimate with a listwise rank loss whose only label is the trajectory outcome, and then never reads it: not in the reward, not in the advantage, not at inference. The loss reaches the policy only through the shared parameters. On ALFWorld, Sokoban and WebShop, with three training seeds each, this objective alone beats the published GiGPO baselines by $+8.6$, $+9.1$ and $+8.0$ points at the same budget, and reaches their final success within 60–73% of it. Where we test consumption — a forward difference of the estimate injected into the advantage at two doses, and three rules for choosing the action with it at inference — it adds nothing we can detect: the injection is flat at both doses and the selection rules do not help. Randomising the targets removes the gain, so what matters is that the targets are correct. What the loss teaches is which states are promising, not which action to take: inside a state, the order it gives the candidate actions is no better than one constant per state. The policy's hidden state already carries the outcome before the loss is added. The practical result is a recipe that needs no demonstrations and no extra forward pass.

1 INTRODUCTION

Reinforcement learning for long-horizon LLM agents started with trajectory-level, critic-free group methods (Shao et al., 2024; Feng et al., 2025) and has moved towards objectives that add a second, finer-grained source of supervision. A popular choice is a *self-assessment signal*: the policy is asked for a self-reported progress, confidence, or success estimate, and that estimate is then *used* — as an advantage (Yang et al., 2026; Liang et al., 2026a), as a reward (Ma et al., 2026; Zhu, 2026; Damani et al., 2026; Zhao et al., 2026), or as a rule for selecting among candidate actions. These methods report real gains, and the field now takes the rule behind them for granted: to turn an estimate into behaviour, you have to consume it.

That rule has never been tested, because learning an estimate and using it are always bundled. Every method above trains the estimate and consumes it in the same run, so the reported gain is the sum of two effects: what the extra supervision teaches the policy, and what reading the estimate adds on top. The split matters in practice as well as in theory. If most of the gain comes from learning, then the expensive parts of the pipelines built on this rule — external demonstrations, supervised warm-up, a second pass over the trajectory, extra reward-shaping terms (Ma et al., 2026) — are not needed to obtain it.

We separate the two effects with a method we call *RAISE* (rank-trained assessment of internal self-estimates). It trains the estimate and *never reads it* (Figure 1). At every environment step the prompt carries one value slot per candidate action, and the policy's own success probability $q(a \mid s)$ is read from its logits at that slot, inside the generation that produces the action. A listwise rank loss on those numbers is driven only by the trajectory outcome, which is the only label available: on a successful trajectory the executed arm is pushed above the other candidates, and on a failed one

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

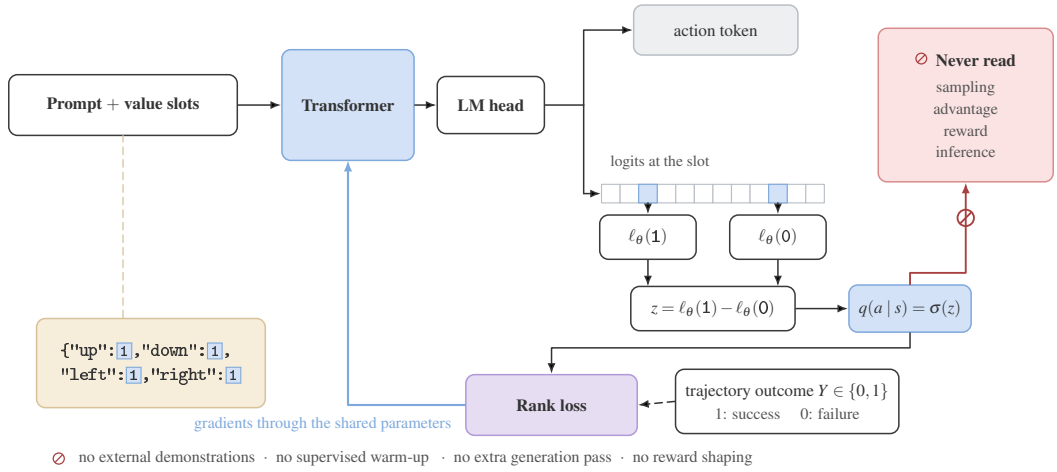


Figure 1: How RAISE works. The prompt carries one value slot per candidate action: the candidate list is the JSON object at the lower left, and each value in it is a constant "1" that the policy fills with `1` for success or `0` for failure, so the policy's own context never tells it what it estimated. The estimates come from the same generation, at the position before each slot: take the vocabulary logits of the tokens `1` and `0`, form $z = \ell_\theta(1) - \ell_\theta(0)$ and take the sigmoid. A rank loss over those numbers is trained with the trajectory outcome Y — `1` on a successful trajectory and `0` on a failed one, a label rather than a score, and only the arm that was actually executed carries any information — and the gradient reaches the policy only through the shared parameters (the blue path at the bottom). Nothing else reads the estimate: the crossed-out arrow marks sampling, the advantage, the reward and inference, all of which use the policy's own action choice. The note at the bottom lists the components of the demonstration-based pipelines (Table 4) that this recipe does not need.

its share of the candidate softmax mass is driven to zero. The estimate is then excluded from the sampling distribution, from the advantage, from the reward, and from inference. The objective adds no parameters and no forward pass: the extra output fields cost 28 prompt tokens (+6.8%) against the baseline scaffold. The only path from the loss to the policy is the shared parameters, so we also measure what that path changes (Section 5).

The unused objective behaves like a real training signal: on three environments and two group-relative backbones it beats the published GiGPO baselines by +8.6, +9.1 and +8.0 points at the same budget, and reaches their final success within 60–73% of it (Table 1 and Figure 2). Turning consumption on, at a dose calibrated across runs, adds nothing we can detect: in the three settings where our intervals are tight enough to bound the effect, the bound rules out a benefit as large as the one the loss itself provides (Table 2), and a larger injection does not change that. What the loss teaches is a state value, not an action chooser: across states the estimate is calibrated and discriminative (marginal AUROC 84–91% during training), but inside a state the order it gives the candidate actions is no better than a single constant (Table 3). The outcome is already decodable from the policy's hidden state before the loss is added (94% on Sokoban, 83% on ALFWorld), so the objective lines up a readout the policy already has. Randomising the targets removes the gain, which says that the correctness of the supervision is what matters.

The contribution is a controlled split between learning a self-assessment estimate and consuming it, a protocol for measuring this kind of signal (Appendix C), and a recipe that needs no demonstrations and no second pass.

2 Related Work

Self-assessment signals that are consumed. RePro (Ma et al., 2026) asks the policy to write a progress value after seeing the finished trajectory and turns its differences into a reward; SAPO (Liang et al., 2026a) reads a state/action value from the policy's own logits and uses it as the advantage; LaSeR (Yang et al., 2026) trains the last-token logit with an auxiliary loss and adds it to the

advantage; RLCR (Damani et al., 2026) and Intuitor (Zhao et al., 2026) turn self-reported confidence into a reward; RefGRPO (Zhu, 2026) trains reflection under outcome supervision; TAPO (Li et al., 2026) adds a transition-prediction auxiliary task and asserts that the shaped representations feed back into the policy; SR-GRPO (Tang & Yang, 2025) turns a geometric property of the representation into a reward; and RetroAgent (Zhang et al., 2026b) derives numerical and language feedback from each finished trajectory and consumes both as a reward and a retrieval memory. Every one of them trains an estimate and consumes it in the same run, so the reported gain mixes the two effects this paper separates. Our objective is the same kind of auxiliary supervision with the consumption removed, which turns the consumption switch into a controlled arm.

Step-level credit for long-horizon agents. GiGPO (Feng et al., 2025) recovers step-level credit by grouping repeated anchor states (the same observation met more than once), and a line of follow-up work improves the same quantity: GraphGPO (Cheng et al., 2026), RewardFlow (Feng et al., 2026), GACA (Liang et al., 2026b), HGPO (He et al., 2026), HyGAE (Zhang et al., 2026a) and GEPO (Yuan et al., 2025), which scores each state by its structural role in the state-transition graph. Another route to fewer training steps is to trim the completions a group is built from (Lin et al., 2025). All of these change how the advantage is estimated, or which completions it is estimated from; we leave that half of the pipeline fixed and change only what the policy is trained on. The group-relative gradient has been characterised as a U-statistic (Zhou et al., 2026) and analysed by Mroueh et al. (2025), which is why we measure how an auxiliary term interacts with the advantage. After the policy converges, anchor states that carry more than one action become rare: the median top-1 action share is 100%, and in Sokoban only 139 of 401 anchor states carry more than one executed action, of which 48 contain both a success and a failure. The step-level signal these methods rely on is data-starved in practice.

Probing the internal state. A recent line shows that the hidden states of multi-turn agents already encode the eventual outcome (Mammen et al., 2025; Ruan et al., 2025; Silva et al., 2025). We reproduce that premise in our own environments with a controlled probe (Section 5), and then ask what *training* a readout of that knowledge does to the policy. CAPO (Wang et al., 2026) proves a mismatch between group-relative advantage estimation and calibration objectives; PQM (Li & Li, 2025) trains a separate process reward model on rankings; MAGELLAN (Gaven et al., 2025) supports the view that self-assessment is a capability of its own.

Potential-based shaping. The classical account of when an extra reward term leaves the optimal policy alone is potential-based shaping (Ng et al., 1999), which is why we expect a term of this shape to do little. Its hypotheses do not hold for our injection — the potential is learned rather than fixed, it is read at the arm that was executed, and the injected term carries no discount — so the theorem motivates the experiment without predicting it. Appendix F states the mismatches.

3 Method: Train the Estimate, Never Read It

RAISE has two parts and one exclusion: a self-reported estimate $q(a \mid s)$ that the policy writes inside its own response, a listwise rank loss on those numbers labelled only by the trajectory outcome, and the rule that the numbers are never read.

3.1 Interface

The candidate actions are known in advance in these environments, so we render them as a JSON object with one value slot per action, and read $q(a \mid s) \in [0, 1]$ from the logits at that slot. The text of the slot is a constant: in the closed-vocabulary environments the object is part of the prompt, so the slot literally reads "1", and on the open-vocabulary pages the probe marker is forced to "1" as well, with the forced tokens excluded from the policy-gradient, KL and entropy terms. Either way, nothing the policy writes tells it what it estimated; the estimate lives in the logits alone. The prompt, the format penalty and the context length are otherwise identical to the baseline scaffold, the slots cost 28 prompt tokens on average ($+6.8\%$), and the read reuses the forward pass the policy is already doing, so the estimate adds no parameters and no extra pass. The value is read from the language head at the slot itself,

$$z(a \mid s) \;=\; \ell_\theta(\texttt{1} \mid s, a) \;-\; \ell_\theta(\texttt{0} \mid s, a), \qquad q(a \mid s) \;=\; \sigma\big(z(a \mid s)\big), \tag{1}$$

where $\ell_\theta(\cdot \mid s, a)$ is the logit of the token at the value slot that follows the entry for a. Appendix B gives the block verbatim for all three environments, including WebShop's search page, where the arms are free-text queries and the policy has to propose the candidate set itself.

3.2 Objective

The only label available in this setting is the trajectory outcome Y: within a state, exactly one arm was executed, and no environment provides the success rate of the arms that were not taken. We therefore train the estimate with a contrast that uses only that label — on a successful trajectory the executed arm is pushed above the other candidates, and on a failed one its share of the candidate softmax mass is driven to zero:

$$\mathcal{L} \;=\; \mathcal{L}_{\mathrm{RL}} \;+\; \lambda_{\mathrm{rank}}\,\mathcal{L}_{\mathrm{rank}}, \qquad \mathcal{L}_{\mathrm{rank}} \;=\; \begin{cases} -\log p_e, & Y = 1, \\ -\log\big(1 - p_e\big), & Y = 0, \end{cases} \quad p_e \;=\; \frac{e^{z_e}}{\sum_{a \in \mathcal{A}(s)} e^{z_a}}, \tag{2}$$

where $\mathcal{L}_{\mathrm{RL}}$ is the backbone objective (GiGPO or GRPO), $\lambda_{\mathrm{rank}} = 0.1$, e is the executed arm, $\mathcal{A}(s)$ is the candidate set, and the two cases are averaged with equal weight so that success and failure rows contribute equally. Ranking is the weakest assumption the label structure allows (Appendix C). Only the executed arm carries information, and the label is constant along a trajectory, so the cheapest solution the loss can find is a function of the state.

3.3 What the objective does not touch

Four exclusions define the method: the estimate is not sampled from, not added to the advantage, not added to the reward, and not read at inference. The rollout distribution, the reward function and the evaluation procedure are therefore those of the backbone, and the only path from $\mathcal{L}_{\mathrm{rank}}$ to the policy is the shared parameters. Any gain has to come through what the objective does to that path — the readout the policy already carries, and to a smaller extent the trunk.

3.4 Consumption at a calibrated dose

To turn consumption on, we inject the forward difference of the estimate into the advantage as a step-level credit, while the action is still chosen by π:

$$c_t \;=\; q_t - q_{t-1} \quad (q_{-1} = 0.5), \qquad \hat{A}_t \;\leftarrow\; \hat{A}_t + \alpha\, c_t, \qquad \frac{\mathbb{E}_t\,|c_t|}{\mathbb{E}_t\,|\hat{A}_t|} \;=\; \rho \;=\; 0.5, \tag{3}$$

where ρ is the realized dose: α is re-scaled each batch to hold it fixed, so the two arms differ in the injection and not in its magnitude. In words, each step's advantage is nudged by how far the estimate moved since the previous step, rescaled so that the nudge is a fixed fraction of the advantage. The injected term is computed from values the rollout has already written and added to the advantage tensor after generation, so it carries no gradient. A raw coefficient is not comparable across environments (the base advantage differs by $3.4\times$), so we scale the injected term to a target fraction of the base advantage and log the realized ratio at every step; at a target of 0.5 the realized median is 0.500 in every run.

4 Results

We evaluate on three long-horizon agent benchmarks: ALFWorld (Qwen2.5-1.5B), Sokoban (6×6 grids, Qwen2.5-VL-3B) and WebShop (Qwen2.5-1.5B), plus a 7B setting for two of them (Qwen2.5-7B on ALFWorld and WebShop). All of our numbers are read from step-150 checkpoints and averaged over three training seeds and three evaluation seeds on 128 held-out tasks. We compare against the published numbers of the official GiGPO release (Feng et al., 2025), always the variant with standard-deviation normalization, and we evaluate under that release's own protocol — the same task splits, the same success criterion and the same decoding settings — so both sides of every comparison are scored by one procedure.

Table 1: Success rate (%) against the published baselines of GiGPO. Baselines are the `GiGPO w/ std` numbers from the GiGPO paper (Table 1 for ALFWorld/WebShop, Appendix Table 3 for Sokoban), all trained for 150 iterations. Within each row the best value is in bold. RAISE rows are read from step-150 checkpoints, 3 training seeds $\times$ 3 evaluation seeds, 128 tasks, and the $\pm$ is the spread over training seeds. The number in parentheses inside each RAISE cell is that cell's margin over the baseline of its own row, so the GRPO and GiGPO columns can be read on their own.

Environment	GRPO	GiGPO w/ std	$\text{RAISE}_{\text{GRPO}}$	$\text{RAISE}_{\text{GiGPO}}$
ALFWorld, 1.5B	$72.8_{\pm 3.6}$	$86.7_{\pm 1.7}$	$75.6_{\pm 6.7}$ $(+2.8)$	$\mathbf{95.3_{\pm 0.8}}$ $(+8.6)$
Sokoban, VL-3B	$67.1_{\pm 4.7}$	$76.9_{\pm 2.7}$	$79.6_{\pm 3.7}$ $(+12.5)$	$\mathbf{86.0_{\pm 2.0}}$ $(+9.1)$
WebShop, 1.5B, succ.	$56.8_{\pm 3.8}$	$65.0_{\pm 3.2}$	$64.5_{\pm 2.9}$ $(+7.7)$	$\mathbf{73.0_{\pm 2.2}}$ $(+8.0)$
WebShop, 1.5B, score	$75.8_{\pm 3.5}$	$\mathbf{83.1_{\pm 1.6}}$	$79.0_{\pm 2.0}$ $(+3.2)$	$80.8_{\pm 3.0}$ (-2.3)
ALFWorld, 7B	$77.6_{\pm 5.2}$	$90.8_{\pm 1.3}$	$82.4_{\pm 4.9}$ $(+4.8)$	$\mathbf{94.2_{\pm 2.0}}$ $(+3.4)$
WebShop, 7B, succ.	$66.1_{\pm 3.7}$	$72.8_{\pm 3.2}$	$76.0_{\pm 9.0}$ $(+9.9)$	$\mathbf{81.9_{\pm 2.4}}$ $(+9.1)$
WebShop, 7B, score	$79.3_{\pm 2.8}$	$84.4_{\pm 2.9}$	$81.8_{\pm 10.8}$ $(+2.5)$	$\mathbf{85.8_{\pm 1.9}}$ $(+1.4)$

4.1 Comparison with published baselines

The unused objective improves the GiGPO backbone by $+8.6$, $+9.1$ and $+8.0$ points on ALFWorld, Sokoban and WebShop (Table 1), with little variation across the three training seeds (Table 9). The same objective on the weaker GRPO backbone gives $+2.8$, $+12.5$ and $+7.7$: the gain does not require the step-level credit GiGPO adds. On ALFWorld it also carries to the unseen half of the evaluation set, where RAISE reaches 87.2 against the baseline's 83.5 (Table 13 in the appendix). At 7B the picture repeats on ALFWorld and WebShop. On WebShop the objective beats the published GiGPO on success rate (73.0 against 65.0) but not on the continuous task score (80.8 against 83.1). Success rate and task score measure different things: WebShop counts a success when the purchase matches the request exactly, and reports the mean match reward over all tasks as the score. Reading the 1.5B GiGPO row that way, more purchases match exactly while the ones that miss exactness match less well, their mean reward falling from 0.52 to 0.29. The loss is trained on the trajectory outcome alone, so once a purchase succeeds nothing in it prefers a closer match.

4.2 The objective alone reaches the baseline well before the end

Figure 2 shows the curves behind Table 1, every five steps and with the spread over the training seeds. We call a level reached at the first plotted step where the three-seed mean of the RAISE curve is at least the published baseline's final success of Table 1. That happens at step 90 on ALFWorld (86.7 ± 1.6 against 86.7), at step 95 on Sokoban (77.3 ± 6.1 against 76.9) and at step 110 on WebShop (65.4 ± 3.3 against 65.0): between 60 and 73% of the baseline's 150-step budget.

4.3 The gain comes from the targets, not from the scaffold

Writing numbers into the response is not by itself what helps. We compare three arms that differ only in what the value slot is trained on: the full loss (***RAISE***), the same scaffold with nothing supervising the slots ($\lambda_{\text{rank}} = 0$), and the same loss with its targets replaced by a Bernoulli draw (***shuffled***). Early in training the $\lambda_{\text{rank}} = 0$ arm is indistinguishable from the original algorithm, so the slots, the format penalty and the extra context are inert on their own. Neither control then recovers the gain: at step 150 the no-loss arm sits at 89.3 on ALFWorld and 60.0 on Sokoban and the shuffled arm at 87.6 and 68.3, against RAISE's 95.3 and 86.0 (Table 10).

4.4 The decomposition: what does consumption add?

Table 2 compares the train-only arm with the consumption arm of Section 3.4, paired by training seed (Table 11) and evaluated separately at step 150 on the held-out tasks. Consumption adds nothing we can detect: the point estimates are negative in all six cells, and in the three GiGPO cells the interval is tight enough to rule out a benefit as large as the one the loss itself provides. Raising the dose recovers nothing either: the last two columns repeat the comparison at a fixed coefficient of one instead of a target ratio, and the differences keep their sign and grow (-3.8, -5.1, -15.4 and -7.3

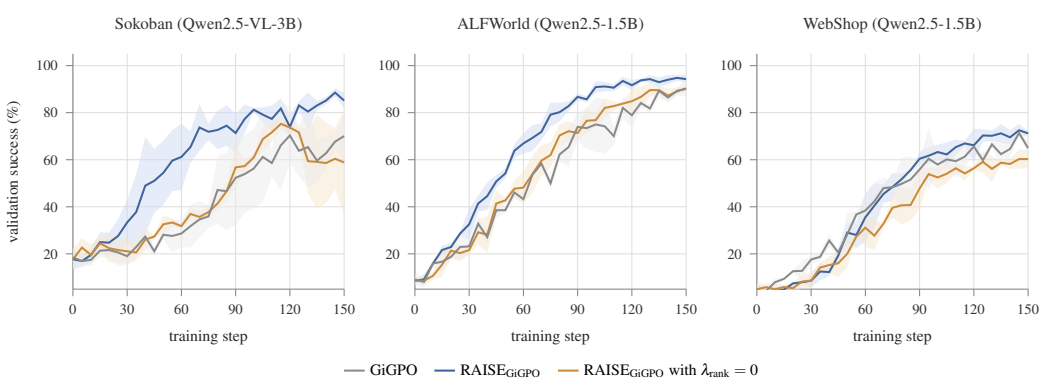


Figure 2: Validation success during training, every five steps and averaged over three training seeds. All three curves are the GiGPO backbone: the "original" curve runs the algorithm itself, "no loss" is the same prompt and the same value slots with nothing supervising them ($\lambda_{\text{rank}} = 0$), and "RAISE" adds the rank loss. The shaded area is ± 1 standard deviation over the seeds. The $\lambda_{\text{rank}} = 0$ curve never rises with RAISE in any panel, so the gain is not the act of writing numbers.

Table 2: Consumption at a calibrated dose, paired per training seed and evaluated separately at step 150 (128 tasks); every quantity is a success rate in percent. Δ is the per-seed difference (consumption minus RAISE) averaged over seeds, so it can differ by 0.1 from the difference of the two printed means; subscripts on the two success columns are the spread over the three training seeds, and on Δ the half-width of its 95% interval (paired t over the same seeds). The third column is the train-only objective of Table 1 at step 150. The effect that arm alone provides is the difference between its row and the no-loss row of Table 10: $+6.0$ points on ALFWorld, $+26.0$ on Sokoban and $+11.7$ on WebShop. The last two columns repeat the comparison for a consumption arm with a fixed coefficient of one instead of a target ratio: that arm realizes a median dose of 0.24–1.10 of the base advantage and was run on ALFWorld and Sokoban only.

Environment	Alg.	RAISE(train only)	Consume ($\rho = 0.5$)	Δ	Consume ($c = 1$)	Δ
ALFWorld	GiGPO	$95.3_{\pm 0.8}$	$93.7_{\pm 0.8}$	$-1.5_{\pm 3.7}$	$91.5_{\pm 1.5}$	$-3.8_{\pm 2.2}$
Sokoban	GiGPO	$86.0_{\pm 2.0}$	$84.9_{\pm 0.7}$	$-1.2_{\pm 6.2}$	$80.9_{\pm 5.2}$	$-5.1_{\pm 7.9}$
WebShop	GiGPO	$73.0_{\pm 2.2}$	$72.0_{\pm 1.9}$	$-1.0_{\pm 2.9}$	—	—
ALFWorld	GRPO	$75.6_{\pm 6.7}$	$69.4_{\pm 11.8}$	$-6.3_{\pm 39.9}$	$60.3_{\pm 13.4}$	$-15.4_{\pm 46.2}$
Sokoban	GRPO	$79.6_{\pm 3.7}$	$77.6_{\pm 6.2}$	$-2.0_{\pm 8.7}$	$72.3_{\pm 10.1}$	$-7.3_{\pm 18.0}$
WebShop	GRPO	$64.5_{\pm 2.9}$	$60.7_{\pm 11.6}$	$-3.8_{\pm 21.6}$	—	—

points): the flat curve holds at both doses. A zero-coefficient run of the same switch injects nothing (Appendix E). The three GRPO cells are too noisy to order the two arms.

4.5 Consuming at inference: what selection does

Inference-time use is a second kind of consumption: we also let the policy choose its action with the estimate, sampling from π among the arms whose estimate clears $\tau = 0.3$, taking the arm with the largest estimate, or sampling from a softmax over the estimates (Table 7). None of the three helps. On the ALFWorld GiGPO arm filtering costs 2.2 points and taking the largest estimate 5.2, and the softmax collapses to 26.5 against π's 95.3; the weaker GRPO arm gains 3.8 and 1.4 points on the first two, but both 95% paired intervals (± 4.2 and ± 11.7) include zero. On Sokoban the three rules leave π's 86.0 at 85.3, 82.1 and 83.8.

Table 3: One set of estimates, two calibers. Across states (second column) the estimate looks discriminative; inside a state the ordering it gives the candidate actions is no better than a single constant (third vs fourth column). The last two columns are the policy's own within-state alignment. All quantities are AUROC in percent, means over three training seeds, with the bracketed 95% cluster-bootstrap interval over states (percentile over 2000 resamples of the anchor-state groups). The marginal column scores the estimate on the pooled held-out dumps, which is why the range here (87–96%) sits above the training-time AUROC of Figure 3 (84–91%).

Config.	Marg.	Action	Per-state const.	RAISE-π	Orig.-π
ALFWorld, GiGPO	86.9	$63.0_{[60.2,\,65.9]}$	$64.7_{[61.5,\,67.8]}$	$57.9_{[54.8,\,60.9]}$	$67.7_{[62.7,\,73.1]}$
ALFWorld, GRPO	91.1	$60.9_{[58.2,\,63.6]}$	$64.9_{[60.9,\,68.7]}$	$58.0_{[55.5,\,60.6]}$	$52.8_{[47.7,\,58.4]}$
Sokoban, GiGPO	96.4	$46.6_{[35.8,\,56.9]}$	$57.0_{[49.7,\,70.8]}$	$44.3_{[33.5,\,56.6]}$	$57.5_{[43.5,\,74.3]}$
Sokoban, GRPO	91.3	$58.9_{[51.9,\,65.4]}$	$56.4_{[47.3,\,64.2]}$	$59.9_{[53.5,\,66.3]}$	$60.4_{[53.4,\,66.7]}$

5 What Is Learned

The objective trains the policy through the shared parameters only (Section 3), so the question is what those parameters end up carrying. We measure it two ways: by asking what the estimate itself can discriminate at the end of training, and by asking what the auxiliary loss changed in the trunk relative to an otherwise identical run without it.

5.1 The estimate becomes a state value

Across states, q predicts the outcome well: marginal AUROC (the chance the estimate ranks a successful rollout above a failed one) is 84–91% at the end of training, with calibration improving alongside (Figure 3): on Sokoban its calibration error falls from 28% to 16% and its correlation with the outcome rises from -1% to 66%; on ALFWorld the AUROC rises from 53% to 84%; on WebShop from 53% to 89% and the calibration error falls from 77% to 14%. The spread over the three training seeds is 2.8–6.7 points for AUROC, so these numbers do not order the environments against each other. The readout of the no-loss scaffold, whose slots are never trained, stays at chance and no better calibrated (46–55 AUROC, 24–46 ECE; Appendix C), so what Figure 3 shows is the loss at work rather than the act of writing numbers, and Appendix C reports the same calibration on the held-out dumps with intervals. The ranking objective is invariant to a common offset on the logits inside a state, so the level of q comes from the sigmoid rather than from the loss.

Inside a single state it is a different story: the order the estimate gives the candidate actions is no better than one constant per state, and the policy's own within-state preference is no better either (Table 3). Pairing the two columns over resampled states, the estimate sits significantly *below* the per-state constant in two of the four settings (Sokoban with GiGPO by 11.5 AUROC points and ALFWorld with GRPO by 4.1, both intervals excluding zero) and is indistinguishable from it in the other two. WebShop is not in that comparison, because its arms are free-text queries and two rollouts at the same step propose different candidate sets. The labels explain both results: inside a state there is nothing to compare, while across states the outcome variance dominates and the cheapest learnable function of the state wins. The estimate therefore ends up a state value with a small action-dependent residue, which is also why consumption cannot help: an estimate that cannot order the actions of a state carries no step-level credit to inject.

5.2 The premise: the outcome is already decodable from the trunk

Before attributing anything to the loss, we check what the policy already knows. We train a linear probe on the last hidden layer of the no-loss arm, which has the same prompt and the same value slots but nothing supervising them, and predict the trajectory outcome, splitting the data five-fold by trajectory with balanced classes. The probe reaches AUROC 94.0 on Sokoban and 82.9 on ALFWorld (per-seed values in Table 12), so the outcome is linearly decodable before the auxiliary loss exists. Restricting the Sokoban probe to the first 30% of each trajectory, which removes the trivial "close to the end" shortcut of a visual environment, still gives 85.7–96.5% across the three seeds. This reproduces the premise of the probing line (Mammen et al., 2025; Ruan et al., 2025; Silva et al., 2025) on two of the three environments; WebShop is open-vocabulary and has no probe.

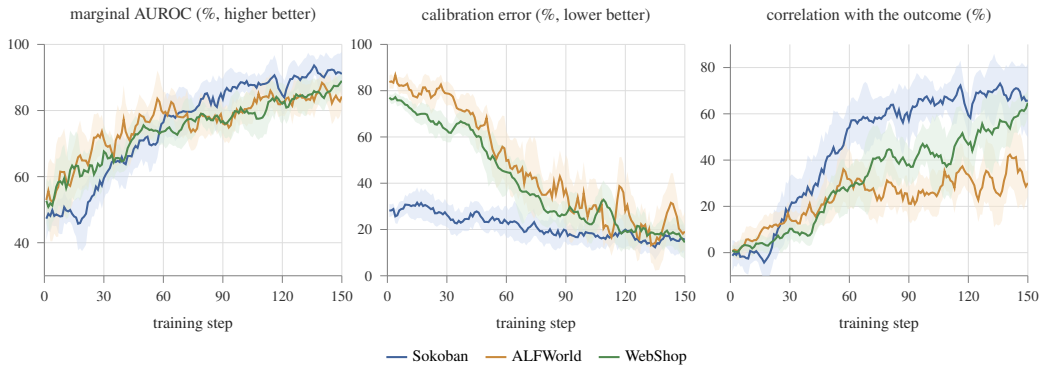


Figure 3: What the estimate becomes during training, one curve per environment (mean over three training seeds, shaded area ± 1 standard deviation, five-step moving average). Discrimination and correlation with the trajectory outcome rise while calibration error falls. The estimate settles into a calibrated measure of how good a state is.

5.3 Readout alignment, measured with a linear probe

We next ask what the loss changes, using the same probe on the trunks of the two arms. On Sokoban the rank-trained trunk is a little easier to decode ($+3.2$ points at the last layer, $+5.8$ on the first 30% of each trajectory, $+6.8$ at an intermediate layer; 12 of the 15 depth-by-seed cells of the full sweep are positive, and Table 12 gives three of those depths), but that change is small next to the change in the readout itself, and on ALFWorld it is gone (82.9 vs 82.7). So we describe the mechanism as the loss aligning a readout the policy already carries; whether the trunk also improves a little depends on the environment. A linear probe measures what is decodable, not what the policy can use.

6 A Recipe

Let the policy write its success estimates for the candidate actions in the same generation as the action, train those numbers with a listwise rank loss labelled only by the trajectory outcome ($\lambda_{\text{rank}} = 0.1$), and keep them out of the advantage, the reward and inference. That exclusion is what makes it cheap: no demonstrations, no second pass, no shaping term. While it trains, watch marginal AUROC, calibration error and the correlation with the outcome (Appendix C); the action-level numbers are too thin to steer by. Consume the estimate only if a dose-controlled experiment on your own environment shows a gain; on our three environments none did. Appendix G covers domains where the candidate actions are not given by the environment.

The auxiliary loss is close to free (Table 4): it adds 28 prompt tokens and no measurable per-step time (418 s against 430 s per step for the otherwise identical no-loss arm, same prompt length; token counts per environment are in Appendix D.2). The pipeline it replaces needs external demonstrations, a supervised warm-up, a second pass over the finished trajectory and three reward-shaping terms (Ma et al., 2026), and on the environments we study that pipeline buys no extra success, because the part that helps is the training of the estimate.

7 Conclusion

Work on self-assessment for agent RL usually bundles two things: learning an estimate of one's own progress, and reading it. We separate them. Trained with a listwise rank loss and never read — not in the reward, not in the advantage, not at inference — the estimate alone improves on the published baselines of its backbone on ALFWorld, Sokoban and WebShop, and reaches their end-of-training level within 60–73% of the baseline's budget. Reading the estimate changes none of that, in any of the forms we test: the advantage injection is flat at both doses and the inference-time selection rules do not help. What the loss teaches is a state value rather than a within-state action chooser: across states the estimate is calibrated and discriminative, inside a state it is no better than one constant per

Table 4: What the auxiliary loss costs, against the pipeline it replaces. The per-step figures are measured on the same code, prompt and hardware as the no-loss arm, and the token counts behind them are in Appendix D.2.

	RAISE	Demonstration-based pipelines
External demonstrations	none	required (large-model API)
Supervised warm-up	none	3 epochs on demonstrations
Extra generation pass	none (same generation)	one pass over the full trajectory
Reward-shaping terms	none	three
Prompt cost	$+28$ tokens ($+6.8\%$)	—
Per-step time vs no-loss arm	418 s vs 430 s (no difference)	—

state, and the outcome is already linearly decodable from the trunk before the loss is added. The practical consequence is a recipe that costs 28 prompt tokens and no extra pass, and a protocol for reporting this kind of signal.

8 LIMITATIONS

Novelty. Self-assessment with outcome supervision has been built on these environments before (Ma et al., 2026), and we use the simplest version of it that separates learning from consuming. The contribution is the measurement and what it implies in practice, not the algorithm.

The consumption result is a bound. Three of the six cells in Table 2 have intervals tight enough to rule out an effect as large as the auxiliary loss itself; the other three cannot order the two arms. We test one form of consumption (a forward difference in the advantage) and one inference-time use of the estimate, the latter on ALFWorld and Sokoban but not on WebShop, where the arms are free-text queries.

Coverage. Three environments, at 1.5B and 3B everywhere and a 7B setting for two of them, with three training seeds and 150 training steps. We compare against the published numbers and the design premises of RePro, SAPO and TAPO.

AI USE STATEMENT

AI assistants were used for language editing only: checking grammar, tightening phrasing, and formatting tables and figures. All research decisions, the design of the experiments, the interpretation of results, and the conclusions are the authors' own.

ETHICS STATEMENT

This work studies how a training objective shapes what a policy learns. The experiments use standard, publicly available agent benchmarks and open-weight models; no human subjects, personal data, or deployed systems are involved. We see no specific ethical risk beyond the general consideration that better training recipes for LLM agents can be used for beneficial or harmful ends. The protocol we recommend includes reporting the no-loss and shuffled-target controls alongside any positive result.

REPRODUCIBILITY STATEMENT

Appendix D lists the numbers behind every claim, Appendix A the training and evaluation settings, including the learning rate, schedule, group sizes, temperatures and step count, and Appendix B the prompt blocks verbatim, including the sentence that makes the model fill the value slots. Our numbers are read from step-150 checkpoints, averaged over three training seeds and three evaluation seeds on 128 held-out tasks per environment. The auxiliary objective, the consumption switch (with its dose normalization and diagnostics), the shuffled-target control and the zero-coefficient placebo are covered by unit tests in the released code, and the figures in this paper are generated by a script that reads the same numbers as the tables.

REFERENCES

Xin Cheng, Shuo He, Lang Feng, HaiYang Xu, Ming Yan, Lei Feng, and Bo An. Beyond trajectory-level attribution: Graph-based credit assignment for agentic reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2026.

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. In *International Conference on Learning Representations (ICLR)*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. GiGPO: Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xiao Feng, Bo Han, Zhanke Zhou, Jiaqi Fan, Jiangchao Yao, Ka Ho Li, Dahai Yu, and Michael Kwok-Po Ng. RewardFlow: Topology-aware reward propagation on state graphs for agentic RL with LLMs. In *International Conference on Learning Representations (ICLR)*, 2026.

Loris Gaven, Thomas Carta, Clément Romac, Cédric Colas, Sylvain Lamprier, and Pierre-Yves Oudeyer. MAGELLAN: Metacognitive predictions of learning progress guide autotelic LLM agents in large goal spaces. In *International Conference on Machine Learning (ICML)*, 2025.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations (ICLR)*, 2026.

Cong Li, Peixi Peng, Yisen Zhao, Xinyu Hu, Shudong Liu, Zhan Su, and Zhuojian Li. TAPO: Transition-aware policy optimization for LLM agents. *arXiv preprint arXiv:2607.27973*, 2026.

Wendi Li and Yixuan Li. Process reward model with Q-value rankings. In *International Conference on Learning Representations (ICLR)*, 2025.

Dayang Liang, Lang Feng, Bo An, and Yunlong Liu. SAPO: Single-rollout autoregressive policy optimization for agentic reinforcement learning. *arXiv preprint arXiv:2608.19842*, 2026a.

Taoran Liang, Yang Liu, Shang Luo, Yingguang Yang, Rongrong Zhang, Yingzong Min, Yulin Huang, Jianshen Zhang, Yongzhi Qi, Kefu Xu, Congjing Ran, and Bin Chong. Granularity-adaptive credit assignment for long-horizon LLM agent reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026b.

Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. CPPO: Accelerating the training of group relative policy optimization-based reasoning models. *arXiv preprint arXiv:2503.22342*, 2025.

Xinbei Ma, Congmin Zheng, Jiyang Qiu, Jiale Hong, Yao Yao, Xiangmou Qu, Jiaxin Yin, Xingyu Lou, Jun Wang, Weiwen Liu, Weinan Zhang, Zhuosheng Zhang, and Hai Zhao. Retrospective progress-aware self-refinement for LLM agent training. *arXiv preprint arXiv:2606.14302*, 2026.

Priyanka M. Mammen, Emil Joswin, and Srujananjali Medicherla. Do agents know when they succeed? calibrating agent confidence from internal representations. *arXiv preprint arXiv:2609.09448*, 2025.

Youssef Mroueh, Nicolas Dupuis, Brian Belgodere, Apoorva Nitsure, Mattia Rigotti, Kristjan Greenewald, Jiri Navratil, Jerret Ross, and Jesus Rios. Revisiting group relative policy optimization: Insights into on-policy and off-policy training. *arXiv preprint arXiv:2505.22257*, 2025.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, 1999.

Kai Ruan, Zihe Huang, Ziqi Zhou, Qianshan Wei, Jinghao Lin, Xuan Wang, and Hao Sun. Doomed from the start: Early abort of LLM agent episodes via a recall-controlled probe cascade. *arXiv preprint arXiv:2607.06503*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

André Silva, Han Tu, and Martin Monperrus. Latent programming horizons in coding agents. *arXiv preprint arXiv:2607.05188*, 2025.

Yixuan Tang and Yi Yang. SR-GRPO: Stable rank as an intrinsic geometric reward for large language model alignment. *arXiv preprint arXiv:2512.02807*, 2025.

Ziqi Wang, Xingzhou Lou, Meiqi Wu, Zhengqi Wen, and Junge Zhang. Calibration-aware policy optimization for reasoning LLMs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

Wenkai Yang, Weijie Liu, Ruobing Xie, Yiju Guo, Lulu Wu, Saiyong Yang, and Yankai Lin. LaSeR: Reinforcement learning with last-token self-rewarding. In *International Conference on Learning Representations (ICLR)*, 2026.

Jiazhen Yuan et al. Graph-Enhanced policy optimization in LLM agent training. *arXiv preprint arXiv:2510.26270*, 2025.

Wenxuan Zhang, Yuhui Wang, Donggang Jia, Xiaoqian Shen, Jian Ding, Ivan Viola, Jürgen Schmidhuber, and Mohamed Elhoseiny. Hybrid advantage estimation with unified critic for VLM agentic reinforcement learning. In *European Conference on Computer Vision (ECCV)*, 2026a.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. RetroAgent: From solving to evolving via retrospective dual intrinsic feedback. *arXiv preprint arXiv:2603.08561*, 2026b.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *International Conference on Learning Representations (ICLR)*, 2026.

Hongyi Zhou, Kai Ye, Erhan Xu, Jin Zhu, Ying Yang, Shijin Gong, and Chengchun Shi. Demystifying group relative policy optimization: Its policy gradient is a U-statistic. *arXiv preprint arXiv:2603.01162*, 2026.

Yinglun Zhu. Closing the reflection gap: A free calibration bonus for agentic RL. *arXiv preprint arXiv:2606.14211*, 2026.