CORD: CONFLICT-AWARE OPTIMIZATION FOR JOINT RL AND DISTILLATION IN AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint reinforcement learning and self-distillation combines task rewards with dense teacher guidance for agent training, but their gradients can partly cancel. A mixing coefficient controls relative scale without selectively removing opposing components. We present **CORD**, a general framework for conflict-aware optimization of two objectives. CORD separates the choice of protected objective from the condition that triggers projection, and combines the gradients once per optimizer mini-batch. A nearest-point formulation yields a closed-form update and characterizes its first-order effect on the protected objective before optimizer transformations. Gradient diagnostics connect conflict in agent training to lost local optimization progress and motivate selective correction. As a gradient-combination mechanism, CORD can be added to different joint training recipes while retaining their objectives. Across three such recipes, it improves ALFWorld success by 1.5–15.9 percentage points. On WebShop, it improves two of the three pairs, with SDAR rising from 60.7% to 64.6% success. Our analyses explain why protection direction and projection gate must be chosen together, and why the distillation objective and its mixing weight remain consequential after projection. The code is available at Anonymous GitHub.

1 INTRODUCTION

Language-model agents learn to complete tasks through sequences of actions, but outcome rewards provide limited credit for individual decisions (Feng et al., 2025). Group Relative Policy Optimization (GRPO) compares returns across sampled attempts (Shao et al., 2024), yet supplies little learning signal when rewards tie. On-policy distillation (OPD) provides dense token-level feedback on student trajectories (Agarwal et al., 2024), but its usefulness depends on the teacher's guidance. Joint training couples this guidance with reward-based optimization, allowing agents to learn from both task outcomes and informed action preferences (Lu et al., 2026).

Existing approaches refine how teacher guidance enters learning. Skill-conditioned self-distillation supplies an informed contextual view of the policy (Wang et al., 2026), while SDAR weights token supervision using detached teacher–student confidence gaps (Lu et al., 2026). RLSD incorporates detached log-probability differences into token advantages, preserving their signs without a separate distillation loss (Yang et al., 2026a). Confidence gates reweight token signals, whereas a gradient inner-product gate tests their accumulated effect. The two operations can be composed for separate RL and OPD losses.

Joint optimization still requires a rule for combining these gradients. Which contribution to preserve on conflict depends on the losses' roles and the training goal. A mixing coefficient changes scale but cannot selectively remove opposing components. Figure 1 diagnoses this interaction:

- **(a) Conflict occurs during agent training.** In the WebShop (Yao et al., 2022) log used for the left panel, 48.2% of mini-batch gradient records have negative RL–distillation cosine.
- **(b) Conflict reduces local optimization progress.** The middle panel shows that a conflicting distillation gradient reduces first-order progress on the RL surrogate relative to RL alone.
- **(c) The opposing component can be removed selectively.** On conflict, projecting one gradient onto the other's orthogonal complement moves the displayed cosine quantiles to zero and recovers the protected gradient's first-order progress. Aligned pairs remain unchanged.

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.

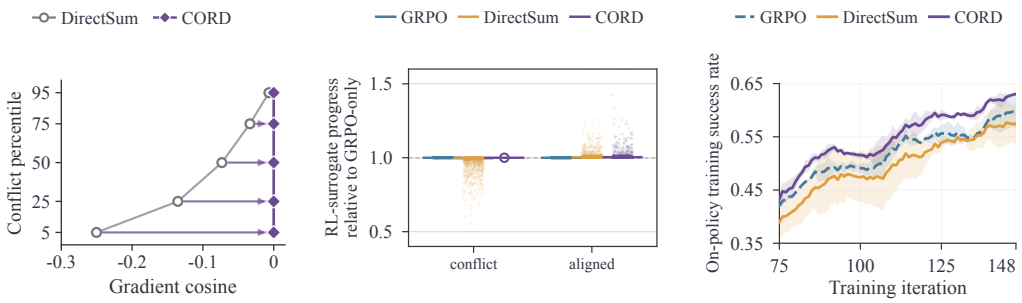


Figure 1: **From gradient conflict to improved agent training. Left, observations (a) and (c):** cosine quantiles of conflicting WebShop/Qwen3-1.7B gradient records (Qwen Team, 2025), before and after analytic CORD projection. **Middle, observation (b):** first-order RL-surrogate progress relative to RL alone on the same log, evaluated at $\lambda = 0.005$ with RL protection. **Right:** preliminary WebShop/3B training success with k_3, $\lambda = 0.001$, and conflict-only distillation protection. The right panel shows iterations 75–148 after 21-iteration smoothing. Curves average two training seeds and bands show their range. Appendix C.1 gives measurement and smoothing details.

We introduce **CORD**, a general framework for conflict-aware optimization of two objectives. CORD accumulates both gradients under the specified loss weights for each optimizer mini-batch. It retains the selected protected gradient and minimally modifies the other before combining them. Conflict-only projection removes opposing components, while an always-projection variant also removes aligned components. Either objective can be protected. Our analysis relates the resulting first-order contribution on fixed sampled losses to finite step size and optimizer transformations. The agent experiments evaluate protection, triggering, and loss construction while retaining each host's objectives. Appendix A discusses related work.

Our contributions are threefold:

- **Conflict-aware joint optimization.** We connect measured gradient conflict to lost first-order progress and formulate CORD to preserve a selected objective's contribution while correcting the conflicting component of the other.
- **Incremental gains across training methods.** CORD improves all three host recipes on ALFWorld by 1.5–15.9 percentage points, and two of three on WebShop. These paired comparisons establish the value of gradient combination while retaining each host's objectives.
- **Guidance for configuring joint training.** We show why protection direction and triggering must be selected together, and how the distillation loss, mixing weight, and aggregation determine the gradient that projection receives. Agent cases connect these choices to attribute completion, repeated actions, and token probability changes.

2 Preliminaries

2.1 Group Relative Policy Optimization (GRPO)

An agent samples response tokens y_t from a rollout policy π_{old}, conditioned on interaction history x_t. GRPO (Shao et al., 2024) compares returns within a group of attempts at the same task. Each response record receives a fixed advantage $\widehat{A}_t = (R_t - \mu_{g(t)})/(\sigma_{g(t)} + \epsilon)$, where $\mu_{g(t)}$ and $\sigma_{g(t)}$ are its group's return mean and standard deviation. The host specifies which response records enter these statistics. With probability ratio $r_t = \pi_\theta(y_t \mid x_t)/\pi_{\text{old}}(y_t \mid x_t)$, the clipped policy loss to minimize is

$$\mathcal{L}_{\text{clip}} = -\mathcal{A}_B\left[\min\left(r_t\widehat{A}_t, \text{clip}(r_t, 1-\varepsilon, 1+\varepsilon)\widehat{A}_t\right)\right]. \tag{1}$$

Here $\mathcal{A}_B$ aggregates valid response tokens in mini-batch B with the host's specified weights, with padding and observation tokens masked. Our GRPO actor loss $\mathcal{L}_{\text{GRPO}}$ includes the host's clipping safeguards, reference-policy KL regularization, and entropy term when enabled (Appendix B.3). All gradients below are loss gradients, so a descent update subtracts the combined gradient.

2.2 On-Policy Distillation

On-policy distillation scores the student's sampled tokens with a teacher rather than requiring teacher-generated trajectories (Agarwal et al., 2024; Zhao et al., 2026; Wang et al., 2026). In our skill-conditioned instantiation, the teacher uses the current policy with additional skill context $S(x_t)$ and gradients disabled (Lu et al., 2026). Define

$$s_t = \log \pi_\theta(y_t \mid x_t), \qquad q_t = \mathrm{sg}[\log \pi_\theta(y_t \mid x_t \oplus S(x_t))], \qquad \ell_t = s_t - q_t, \tag{2}$$

where sg denotes stop-gradient. A token objective ψ_t and confidence weight w_t define a separate loss

$$\mathcal{L}_{\mathrm{OPD}} = \mathcal{A}_B[w_t \psi_t], \qquad \psi_t^{k_3} = \rho_{\mathrm{on},t}(e^{-\ell_t} - 1 + \ell_t), \qquad \psi_t^{\rho_{\mathrm{d}}} = \rho_{\mathrm{on},t}\ell_t. \tag{3}$$

Equation 3 suppresses the numerical exponential cap. Gradients pass through the capped student-to-rollout ratio $\rho_{\mathrm{on},t}$ and s_t, while teacher scores and enabled confidence weights are detached. The host k_1 term and numerical caps follow the inherited recipes in Appendix B.2. Token weights enter before aggregation, while the mixing coefficient below scales the resulting loss.

2.3 Multi-Objective Optimization and Gradient Conflict

Bold symbols denote parameter-space vectors. Losses, inner products, and coefficients are scalars. We use $\boldsymbol{G}_j$ for a complete aggregated objective gradient and $\boldsymbol{g}_j^{(m)}$ for a micro-batch gradient. The role vectors $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{a}, \boldsymbol{b}$ refer to these aggregates after the stated weighting and protection choices.

For two mini-batch losses F_1, F_2 and parameters $\theta \in \mathbb{R}^d$, weighted-sum optimization minimizes $F_1 + \lambda F_2$. At fixed $\lambda > 0$, write

$$\boldsymbol{u} = \boldsymbol{G}_1 = \nabla F_1(\theta), \qquad \boldsymbol{v} = \lambda \boldsymbol{G}_2 = \lambda \nabla F_2(\theta), \qquad \boldsymbol{g}_{\mathrm{sum}} = \boldsymbol{u} + \boldsymbol{v}. \tag{4}$$

Our agent instantiation uses $F_1 = \mathcal{L}_{\mathrm{GRPO}}$ and $F_2 = \mathcal{L}_{\mathrm{OPD}}$. The same combination applies to any two differentiable losses with specified aggregation. A small step along $-\boldsymbol{g}_{\mathrm{sum}}$ changes F_1 by

$$F_1(\theta - \eta \boldsymbol{g}_{\mathrm{sum}}) - F_1(\theta) = -\eta\big(\|\boldsymbol{u}\|^2 + \langle \boldsymbol{u}, \boldsymbol{v}\rangle\big) + o(\eta). \tag{5}$$

For λF_2, the corresponding expression replaces $\|\boldsymbol{u}\|^2$ by $\|\boldsymbol{v}\|^2$. A negative cross term therefore reduces each objective's first-order progress relative to its own gradient. We call this *gradient conflict*. Scaling $\lambda > 0$ changes its magnitude but not its sign. CORD addresses this interaction by choosing which weighted gradient to preserve and minimally modifying the other.

3 Method

CORD combines two objective gradients by making the protected direction and projection gate explicit. It keeps one weighted gradient intact and makes the smallest change to the other that satisfies the chosen geometric constraint. The same construction accommodates either protection direction. The loss functions and their relative weight specify the signals supplied to it. Figure 2 illustrates the combination for GRPO and OPD, with an analytic loss surface showing example update paths.

3.1 Choosing a protected direction

Using the weighted gradients in Equation 4, CORD selects a protected gradient $\boldsymbol{a}$ and a candidate gradient $\boldsymbol{b}$ from $(\boldsymbol{a}, \boldsymbol{b}) \in \{(\boldsymbol{u}, \boldsymbol{v}), (\boldsymbol{v}, \boldsymbol{u})\}$. The protected loss F_a is correspondingly F_1 or λF_2, so that $\nabla F_a = \boldsymbol{a}$. The loss definitions and training goal inform this protection choice. Swapping roles alters the combined direction without redefining losses or fixing priority across problems.

3.2 Projection direction and gate

Under the *conflict-only* gate, CORD combines gradients as

$$\boldsymbol{g}_{\mathrm{final}} = \boldsymbol{a} + \tilde{\boldsymbol{b}}, \qquad \tilde{\boldsymbol{b}} = \begin{cases} \boldsymbol{b} - \dfrac{\langle \boldsymbol{a}, \boldsymbol{b}\rangle}{\|\boldsymbol{a}\|^2}\boldsymbol{a}, & \langle \boldsymbol{a}, \boldsymbol{b}\rangle < 0, \\ \boldsymbol{b}, & \langle \boldsymbol{a}, \boldsymbol{b}\rangle \geq 0. \end{cases} \tag{6}$$

A zero protected gradient leaves the candidate unchanged.

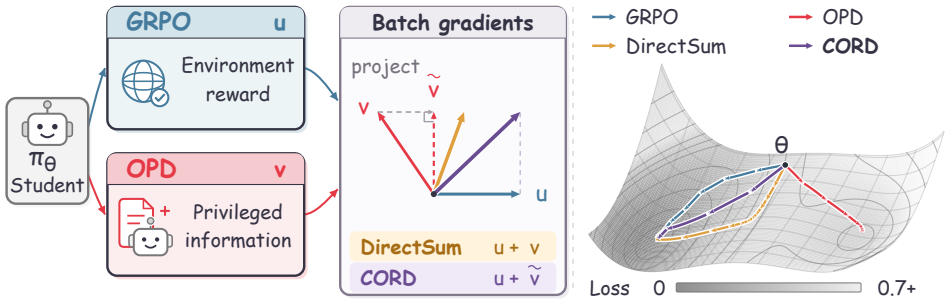


Figure 2: **CORD combines two learning signals through gradient projection.** Left: GRPO uses environment rewards, while OPD uses a teacher conditioned on privileged skill context, with teacher gradients disabled. Middle: accumulated gradients $\boldsymbol{u} = \nabla F_1$ and $\boldsymbol{v} = \lambda \nabla F_2$ give the DirectSum $\boldsymbol{u} + \boldsymbol{v}$. For the conflicting pair shown, CORD protects $\boldsymbol{u}$ and removes the opposing component of $\boldsymbol{v}$, giving $\boldsymbol{u} + \tilde{\boldsymbol{v}}$. Either objective can be protected. Right: illustrative paths in an analytic two-loss construction with GRPO protection. Height and shading encode its GRPO loss. Arrows denote updates of step size 0.7, stopping within 0.10 of either local minimum.

Lemma 1 (Closest feasible replacement)**.** *For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$, the candidate $\tilde{\boldsymbol{b}}$ in Equation 6 is the unique solution of*

$$\min_{\boldsymbol{z} \in \mathbb{R}^d} \tfrac{1}{2}\|\boldsymbol{z} - \boldsymbol{b}\|^2 \quad \textit{subject to} \quad \langle \boldsymbol{a}, \boldsymbol{z} \rangle \geq 0. \tag{7}$$

Proof. For $\boldsymbol{a} = \boldsymbol{0}$ every vector is feasible, and the minimizer is $\boldsymbol{b}$. For $\boldsymbol{a} \neq \boldsymbol{0}$, the Lagrangian is $\frac{1}{2}\|\boldsymbol{z} - \boldsymbol{b}\|^2 - \mu\langle \boldsymbol{a}, \boldsymbol{z} \rangle$. Stationarity gives $\boldsymbol{z} = \boldsymbol{b} + \mu\boldsymbol{a}$. Feasibility, $\mu \geq 0$, and complementary slackness give $\mu = 0$ when $\langle \boldsymbol{a}, \boldsymbol{b} \rangle \geq 0$, and $\mu = -\langle \boldsymbol{a}, \boldsymbol{b} \rangle / \|\boldsymbol{a}\|^2$ otherwise. These conditions are sufficient for this convex problem, and strict convexity makes the minimizer unique. □

The *always* gate instead requires $\langle \boldsymbol{a}, \tilde{\boldsymbol{b}} \rangle = 0$: for nonzero $\boldsymbol{a}$ it applies the projection in the first branch at every update, including aligned gradients. Disabling projection sets $\tilde{\boldsymbol{b}} = \boldsymbol{b}$ and restores weighted summation. CORD uses the single-constraint geometry of A-GEM (Chaudhry et al., 2019), with two current objective gradients and an explicit choice of protection direction and gate. PCGrad (Yu et al., 2020) modifies both gradients on conflict, whereas CORD keeps the protected gradient intact.

Proposition 1 (First-order effect on the protected loss)**.** *Fix θ, λ, and the mini-batch losses, and suppose F_a is differentiable at θ. With conflict-only projection,*

$$\begin{aligned} \langle \boldsymbol{a}, \boldsymbol{g}_{\text{final}} \rangle &= \|\boldsymbol{a}\|^2 + \max\{\langle \boldsymbol{a}, \boldsymbol{b} \rangle, 0\}, \\ F_a(\theta - \eta \boldsymbol{g}_{\text{final}}) - F_a(\theta) &= -\eta \langle \boldsymbol{a}, \boldsymbol{g}_{\text{final}} \rangle + o(\eta). \end{aligned} \tag{8}$$

for $\eta \to 0^+$. Whenever projection is applied, $\langle \boldsymbol{a}, \boldsymbol{g}_{\text{final}} \rangle = \|\boldsymbol{a}\|^2$. This equality also holds under the always gate.

Proof. The projected candidate is orthogonal to $\boldsymbol{a}$, giving the equality on projection. Otherwise $\tilde{\boldsymbol{b}} = \boldsymbol{b}$ and the inner product is nonnegative. Differentiability gives the stated expansion for this fixed direction. The zero-reference case has both inner products equal to zero. □

Projection preserves the protected gradient's first-order descent rate. When it applies with $\boldsymbol{a} \neq \boldsymbol{0}$, orthogonal decomposition of $\boldsymbol{b}$ gives

$$\langle \boldsymbol{b}, \tilde{\boldsymbol{b}} \rangle = \|\tilde{\boldsymbol{b}}\|^2 = \|\boldsymbol{b}\|^2 - \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle^2}{\|\boldsymbol{a}\|^2}. \tag{9}$$

This identity describes the surviving candidate alone. The full direction also includes $\boldsymbol{a}$, so the unprotected loss need not decrease. These properties concern the ideal combined gradient. Global norm clipping rescales it, whereas AdamW's momentum, adaptive scaling, and weight decay can change the update direction (Loshchilov & Hutter, 2019). Appendix B.3 relates the first-order term to the actual parameter displacement and finite-step loss change.

Algorithm 1: CORD gradient combination. At $\lambda = 0$, use $\boldsymbol{G}_1$ directly.

Input: mini-batch B; losses F_1, F_2 with specified aggregation; coefficient λ; protection direction; gate

Accumulate $\boldsymbol{G}_1 = \nabla F_1$ and $\boldsymbol{G}_2 = \nabla F_2$ separately over B

$\boldsymbol{u} \leftarrow \boldsymbol{G}_1, \quad \boldsymbol{v} \leftarrow \lambda \boldsymbol{G}_2$

Select $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow (\boldsymbol{u}, \boldsymbol{v})$ or $(\boldsymbol{v}, \boldsymbol{u})$

Reduce the inner-product and norm statistics across gradient shards

$\tilde{\boldsymbol{b}} \leftarrow \boldsymbol{b}$

if $\|\boldsymbol{a}\|^2 > 0$ *and (gate is always or (gate is conflict-only and* $\langle \boldsymbol{a}, \boldsymbol{b} \rangle < 0$*))* **then**

  $\tilde{\boldsymbol{b}} \leftarrow \boldsymbol{b} - (\langle \boldsymbol{a}, \boldsymbol{b} \rangle / \|\boldsymbol{a}\|^2)\boldsymbol{a}$

end

Pass $\boldsymbol{g}_{\text{final}} \leftarrow \boldsymbol{a} + \tilde{\boldsymbol{b}}$ to the optimizer

3.3 How protection interacts with the mixing weight

The two protection directions share a conflict test but respond differently to the coefficient. Write $\boldsymbol{G}_1 = \nabla F_1$ and $\boldsymbol{G}_2 = \nabla F_2$, and fix a conflicting pair with $c = \langle \boldsymbol{G}_1, \boldsymbol{G}_2 \rangle < 0$. Equation 6 becomes

$$\boldsymbol{g}_{\text{protect 1}} = \boldsymbol{G}_1 + \lambda \left(\boldsymbol{G}_2 - \frac{c}{\|\boldsymbol{G}_1\|^2} \boldsymbol{G}_1 \right), \tag{10}$$

$$\boldsymbol{g}_{\text{protect 2}} = \lambda \boldsymbol{G}_2 + \left(\boldsymbol{G}_1 - \frac{c}{\|\boldsymbol{G}_2\|^2} \boldsymbol{G}_2 \right). \tag{11}$$

In the first case, λ scales the candidate and its correction together. In the second, it scales the protected gradient, while the correction to $\boldsymbol{G}_1$ is independent of λ. If ϕ is the angle between the unweighted gradients, the corresponding correction norms are $\lambda \|\boldsymbol{G}_2\| |\cos \phi|$ and $\|\boldsymbol{G}_1\| |\cos \phi|$. Swapping protection therefore changes which gradient component λ controls.

For a fixed pair and $\lambda > 0$, scaling does not change the conflict decision. Conflict-only projection retains aligned candidates. The always gate removes their parallel component as well. The coefficient controls scale, while the gate controls whether a helpful component is retained. Appendix C.3 isolates these effects at fixed gradients, while RQ2 compares the resulting trained policies.

3.4 Accumulation and agent training

For an aggregation rule $F_j = \sum_m \alpha_m F_j^{(m)}$, with fixed weights α_m, define $\boldsymbol{g}_j^{(m)} = \nabla F_j^{(m)}$ and $\boldsymbol{G}_j = \sum_m \alpha_m \boldsymbol{g}_j^{(m)}$. CORD acts once on these accumulated mini-batch gradients. Projection is nonlinear in the gradient pair, so projecting each micro-batch separately usually produces a different update. Accumulating first defines the projection at the optimizer's update scale. The loss aggregation rule sets the weights before accumulation. Keeping those weights fixed makes repartitioning a computational choice, whereas changing from micro-batch to global-token or trajectory averaging changes the gradients themselves. Appendix C.4 compares these choices.

For agents, Equation 4 becomes $\boldsymbol{u} = \boldsymbol{G}_r$ and $\boldsymbol{v} = \lambda \boldsymbol{G}_o$: protecting GRPO projects OPD, while protecting OPD projects GRPO. Skill-SD (Wang et al., 2026) uses k_3. GRPO+OPSD and SDAR (Lu et al., 2026) retain their host k_1 objectives and weighting conventions. The loss analysis also examines ρ_{d} from Equation 3. Appendix B.2 specifies the recipes. Main host comparisons protect RL with conflict-only projection. The direction and gate analysis tests both protection choices.

Computational complexity. Given the two objective gradients, CORD uses $O(d)$ arithmetic and $O(d)$ additional gradient storage for d trainable parameters. With K PPO mini-batches per training iteration, the gradient-combination cost is $O(Kd)$. When the objectives depend on the same student outputs, their gradient evaluations can share the forward computation. They still require distinct objective-gradient information, followed by inner products, a projection, and vector addition. Shared computation and fused vector operations can reduce redundant work without changing the projection rule. For a fixed workload, actor overhead differs from full-iteration overhead. Rollout, teacher scoring, and environment interaction are shared stages, so their share of each iteration determines

how additional gradient work affects total time. The component model in Appendix C.6 holds the workload fixed and separates measured actor timings from estimated iteration overhead.

4 EXPERIMENTS

We test whether changing the gradient-combination rule adds value across joint training methods, then use the analyses to guide its configuration. The experiments answer three research questions: **(RQ1) Effectiveness:** How much does CORD improve existing joint training recipes? **(RQ2) Projection choices:** How do the protected objective, projection gate, and loss construction guide practical configuration? **(RQ3) Agent behavior:** How do update choices appear in agent interactions?

4.1 EXPERIMENTAL SETUP

Tasks and training. We use the SDAR agent suite (Lu et al., 2026) with Qwen2.5-3B-Instruct (Yang et al., 2024). ALFWorld (Shridhar et al., 2021) evaluates household tasks over at most 50 actions. We report success across six categories and the pooled average. WebShop (Yao et al., 2022) evaluates instruction-following shopping. We cap episodes at 15 actions and report graded task score and strict success rate on a 0–100 scale. Our shared trainer uses 150 training iterations, eight rollouts per prompt, AdamW at 10^{-6}, reference-KL coefficient 0.01, and responses capped at 512 tokens on eight NVIDIA A100 40 GB GPUs. Full settings are given in Appendix B.

Methods. Table 1 includes Vanilla, Skill-Prompt, and OPSD values reported by Lu et al. (2026). GRPO and RLSD results are our own runs. RLSD incorporates the self-distillation signal into the policy optimization objective (Yang et al., 2026a). Following SDAR, the joint recipes share a predefined skill bank whose contents remain fixed throughout training. Each Base–CORD pair uses the same bank and skill-selection rule to isolate gradient combination. Household tasks select task-specific manipulation skills, while shopping instructions select relevant skills by keyword alongside general strategies. This context guides teacher scoring of student-generated actions and is absent from the student's rollout and evaluation inputs. We apply CORD to three joint training recipes in this shared interface: Skill-SD (Wang et al., 2026) (k_3, $\lambda = 0.001$), GRPO+OPSD (host k_1, $\lambda = 0.005$), and SDAR (Lu et al., 2026) (confidence-weighted host k_1, $\lambda = 0.01$). Each recipe without CORD uses naive addition. + CORD retains its coefficient, protects the RL gradient, and projects only on conflict. RQ2 then varies the protected objective and projection gate.

Evaluation. We fix validation tasks to track progress on the same problems during training and hold the final evaluation tasks constant across paired methods. For the main comparison and full-training analyses in RQ2, including PCGrad (Yu et al., 2020), we evaluate the final policy four times after 150 updates at temperature 0.4. Each run uses one training seed, and the displayed $\pm$ is the population standard deviation across these four passes. Our WebShop evaluation uses 500 tasks, whereas the published baseline values in Table 1 use the 128-task protocol of Lu et al. (2026). Appendix B.3 details the protocol.

4.2 RQ1: INCREMENTAL EFFECTIVENESS

Table 1 measures the increment from adding CORD to each joint training recipe. On ALFWorld, CORD improves all three recipes: pooled success increases from 76.4% to 77.9% for Skill-SD, from 65.0% to 80.9% for GRPO+OPSD, and from 73.8% to 77.7% for SDAR. These are gains of 1.5, 15.9, and 3.9 percentage points, respectively. The largest gain occurs for GRPO+OPSD, which reaches the highest pooled success in the main comparison.

On WebShop, CORD improves Skill-SD success by 3.6 points (55.8% to 59.4%) and SDAR success by 3.9 points (60.7% to 64.6%). Their task scores also increase, from 72.1 to 78.1 and from 76.6 to 78.2, respectively. For GRPO+OPSD, success changes from 64.8% to 64.5% and task score from 80.5 to 77.5. Thus, the same projection configuration improves all three recipes on ALFWorld and two on WebShop. Appendix C.7 reports the corresponding statistical tests. Appendix B.4 adds 7B results with gains on ALFWorld and mixed outcomes on WebShop. These differences motivate examining how protection and gating interact with the loss, coefficient, and gradient aggregation.

Table 1: **Final results on Qwen2.5-3B-Instruct.** ALFWorld success and WebShop score/success are on a 0–100 scale. †: Lu et al. (2026). ∗: skills at inference. Our step-150 results use one training seed and four evaluation passes. ± denotes decode standard deviation. CORD protects RL with conflict-only projection. Δ = CORD − Base uses displayed means: success differences are percentage points and score differences are score points, with red/green indicating increases/decreases. Deltas have no assigned standard deviation and are excluded from ranking. Bold/underline mark the largest/second-largest distinct method means. Ties share the mark.

	ALFWorld							WebShop	
Method	Pick	Look	Clean	Heat	Cool	Pick2	**Avg.**	Score	Success Rate
Vanilla†	44.4	11.1	6.2	15.4	28.6	12.5	21.9	6.7	0.8
Skill-Prompt*†	51.7	66.7	48.4	0.0	4.3	10.0	28.9	0.2	0.8
OPSD†	48.8	41.7	16.7	0.0	15.8	16.7	28.1	11.3	3.1
GRPO	93.5±3.1	72.9±12.3	67.0±3.0	62.5±5.9	59.5±2.4	66.3±8.3	71.3±1.9	71.1±1.1	59.2±1.5
RLSD	95.4±1.6	68.8±3.6	83.0±3.9	**87.5**±7.8	56.0±3.9	53.8±6.7	74.4±2.2	69.8±0.7	54.4±1.0
Skill-SD	89.3±2.4	**78.8**±6.4	85.2±3.7	65.6±7.0	60.0±2.8	**70.8**±7.8	76.4±1.7	72.1±0.9	55.8±0.9
+ CORD	91.4±3.5	67.3±6.4	87.0±4.1	67.2±5.2	**78.0**±2.0	60.4±2.1	77.9±1.5	78.1±0.2	59.4±0.8
Δ	+2.1	−11.5	+1.8	+1.6	+18.0	−10.4	+1.5	+6.0	+3.6
GRPO+OPSD	94.4±6.1	60.4±12.3	75.0±2.5	58.9±5.9	50.0±5.3	41.3±4.2	65.0±2.6	**80.5**±0.6	**64.8**±0.2
+ CORD	**98.6**±1.4	75.0±6.4	88.0±4.0	75.0±6.3	66.0±2.0	69.8±4.5	**80.9**±1.6	77.5±0.2	64.5±1.0
Δ	+4.2	+14.6	+13.0	+16.1	+16.0	+28.5	+15.9	−3.0	−0.3
SDAR	96.3±2.6	64.6±3.6	81.2±3.0	67.9±3.6	59.5±7.9	61.5±4.7	73.8±1.4	76.6±1.0	60.7±0.7
+ CORD	88.6±6.4	76.9±0.0	**89.8**±1.6	79.7±8.1	74.0±3.5	51.0±3.5	77.7±3.1	78.2±1.1	64.6±1.1
Δ	−7.7	+12.3	+8.6	+11.8	+14.5	−10.5	+3.9	+1.6	+3.9

4.3 RQ2: The roles of projection and gradient choices

Table 2: **Protection direction and gating on WebShop.** Qwen2.5-3B-Instruct with k_3, reporting success (%). Values are means and population standard deviations across four evaluation passes at temperature 0.4 after 150 updates, with one training seed. Protect RL projects the distillation gradient. Protect distillation projects the RL gradient. PCGrad projects both on conflict. Bold identifies the largest observed mean at each coefficient.

Projection rule	Gate	$\lambda = 0.001$	$\lambda = 0.005$
No projection	—	55.75 ±0.91	56.75 ±0.89
Protect RL	Conflict-only	59.35 ±0.79	61.85 ±1.59
Protect RL	Always	62.65 ±0.79	61.25 ±0.92
Protect distillation	Conflict-only	58.35 ±0.74	61.90 ±1.08
Protect distillation	Always	38.90 ±0.95	**62.10** ±1.54
PCGrad (Yu et al., 2020)	Conflict-only	**63.10** ±0.91	58.85 ±0.95

Choose protection direction and trigger together. The protected objective determines which gradient remains intact, while the gate determines whether projection also removes aligned components. Table 2 compares these choices on WebShop. At $\lambda = 0.001$, conflict-only projection improves success over no projection for both RL and distillation protection, from 55.75% to 59.35% and 58.35%, respectively. Switching to always projection raises success to 62.65% when protecting RL but lowers it to 38.90% when protecting distillation. At $\lambda = 0.005$, all four CORD configurations instead lie between 61.25% and 62.10%, above the 56.75% unprojected result. Direction and gating therefore interact with the coefficient: removing the aligned component has a different effect depending on which objective is protected and the relative gradient scales.

Table 3: **Token objectives at $\lambda = 0.005$.** ALFWorld success and WebShop score/success are on a 0–100 scale. CORD protects RL with conflict-only projection. Values are mean ± standard deviation across four decoding passes of step-150 Qwen2.5-3B-Instruct checkpoints at temperature 0.4, with one training seed. Bold marks the largest mean per column.

	ALFWorld							WebShop	
Objective	Pick	Look	Clean	Heat	Cool	Pick2	**Avg.**	Score	Success
k_1	**95.71** ±4.74	71.15 ±6.38	71.30 ±4.81	56.25 ±4.42	67.00 ±9.54	66.67 ±5.10	74.11 ±3.00	73.68 ±0.79	**59.95** ±0.77
ρ_{d}	92.14 ±3.11	65.38 ±3.85	85.19 ±2.62	78.13 ±3.13	70.00 ±2.00	**70.83** ±2.95	79.11 ±1.28	69.13 ±0.73	58.95 ±0.38
k_3	93.57 ±1.24	**80.77** ±8.60	**91.67** ±3.07	**79.69** ±6.81	**85.00** ±1.73	63.54 ±1.80	**83.75** ±1.06	**74.66** ±0.44	55.55 ±0.79

CORD keeps one gradient intact, whereas the two-objective PCGrad adaptation (Yu et al., 2020) modifies both when they conflict (Appendix B.3). At $\lambda = 0.001$, PCGrad has the largest observed mean of 63.10%, but at $\lambda = 0.005$ its 58.85% is below all four CORD configurations.

The fixed-gradient analysis in Appendix C.3 clarifies how CORD's protection direction interacts with λ. Protecting RL changes the weighted distillation gradient, so the correction scales with λ. Protecting distillation changes the RL gradient, whose correction is unchanged when only λ varies. *A small auxiliary coefficient therefore does not make distillation protection a small intervention.* Because conflict-only projection retains aligned contributions, the effect of always projection depends on the selected protection direction.

Tune the loss and its coefficient together. The token objective determines the distillation gradient supplied to CORD. Table 3 compares k_3, the importance-weighted log-ratio variant ρ_{d}, and the host k_1 implementation at $\lambda = 0.005$, with RL protection and conflict-only projection. The relative performance of the objectives changes across environments: ρ_{d} exceeds k_3 on WebShop (58.95% versus 55.55%), while k_3 exceeds ρ_{d} on ALFWorld (83.75% versus 79.11%). The host k_1 reaches 59.95% and 74.11%, respectively. The two importance-weighted losses supply different token gradients: away from numerical caps, their derivatives with respect to the student log-probability are $\rho_{\mathrm{on},t}\ell_t$ for k_3 and $\rho_{\mathrm{on},t}(\ell_t + 1)$ for ρ_{d} (Equation 15). Changing the loss therefore changes what CORD projects, even at the same coefficient. On WebShop, k_3 also has the highest graded score despite lower strict success, so loss selection must follow the intended metric.

Figure 3 shows how the effect of λ depends on the token objective. Increasing it from 0.001 to 0.005 lowers k_3 success from 59.35% to 55.55%, but raises ρ_{d} success from 56.20% to 58.95%. *Projection therefore does not make a coefficient transferable between losses.* These loss and coefficient analyses use a separate checkpoint collection from the direction and gate analysis.

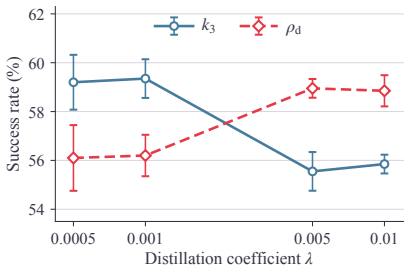


Figure 3: **Effect of λ on WebShop success.** Error bars show standard deviation across decoding passes.

Fix aggregation before comparing projection choices. The mini-batch size sets how many response sequences contribute to each gradient pair. Five-round ALFWorld continuations from the same checkpoint and AdamW state vary this size from 32 to 128 at a fixed episode budget (Table 4). Success under RL protection varies by 0.78 percentage points, while distillation protection peaks at size 64. Larger mini-batches change both gradient estimation and the number of optimizer updates. These continuations update the policy and collect new trajectories, testing training consequences beyond the geometry of a fixed gradient pair.

Aggregation also determines how tokens and trajectories contribute to each gradient. At size 64, the protection directions nearly tie under microbatch weighting. Distillation protection has the higher mean under global-token weighting, whereas RL protection has the higher mean under trajectory

weighting. All three paired intervals include zero, so these orderings remain descriptive. Appendix C.4 defines the weights and protocol. Repartitioning fixed weights changes the computation, whereas changing the weights changes the objectives being protected.

Table 4: **Effect of gradient aggregation and mini-batch size.** ALFWorld task-macro success (%) after five continuation rounds from the same Qwen2.5-3B step-90 checkpoint and AdamW state. B counts response rows per optimizer mini-batch. Appendix C.4 specifies the single-seed evaluation protocol and aggregation weights. Bold marks row maxima.

Aggregation	B	No projection	Protect RL	Protect distillation
Microbatch	64	53.98	**57.55**	57.42
Global token	32	52.52	**57.46**	49.22
Global token	64	59.11	57.33	**61.03**
Global token	128	**61.81**	58.10	55.86
Trajectory	64	60.99	**61.77**	57.42

4.4 RQ3: Case analysis of agent behavior

Success rates do not show which attributes an agent missed or how many actions it repeated. Figure 4 compares three policies that open the same product to buy a youth garment in dark heather, size 4t. The unprojected policy selects the size and age group but omits the color, receiving a partial score of 0.90. Both projected policies complete all requested options and receive 1.00. With RL protection, the agent selects each option once and purchases in six steps. With distillation protection, it repeats four selections and purchases in ten. *Attribute completion and action efficiency are thus separate aspects of behavior*: both directions succeed here, but their repetitions differ. Further examples are provided in Appendices C.8 and C.9.



Figure 4: **Option completion and repetition in WebShop.** Qwen2.5-3B-Instruct, k_3, $\lambda = 0.005$, step 150, conflict-only projection, and greedy decoding. Boxes show selected options and the purchase action, with original step indices. Green highlights selection of the requested color. Dashed amber boxes mark repeated selections. Scores are on $[0, 1]$. Success requires 1.00.

5 Conclusion

Joint RL and distillation gives agents complementary feedback, but opposing gradient components can reduce its local optimization benefit. CORD applies a simple projection rule to this interaction, allowing either objective to be protected and making the trigger explicit. The first-order analysis explains the correction on fixed surrogates before optimizer transformations, and the agent experiments establish incremental gains across three host recipes on ALFWorld and two of three on WebShop. The analyses also yield concrete configuration guidance: choose protection and triggering together, tune the mixing weight for the selected distillation loss, and specify aggregation before comparing gradient directions. CORD thus makes gradient combination a reusable design choice.

AI USE STATEMENT

Generative AI tools assisted with language editing, literature search, and refining the presentation of theoretical arguments and proofs. The authors take responsibility for the final content.

ETHICS STATEMENT

This work studies gradient combination in benchmark agent training. CORD's projection rule does not impose a safety constraint, and benchmark improvements do not establish safe behavior in real-world interactions. Trained agents may retain harmful outputs or biases from their underlying models, feedback, and task data. Deployment requires application-specific evaluation and safeguards.

REPRODUCIBILITY STATEMENT

Section 3 specifies the update rule and provides its proofs. Appendix B documents the tasks, fixed skill bank, host recipes, training settings, and evaluation protocol. Appendix B.3 states the assumptions for the optimizer-step analysis, and Appendix C describes the additional analyses and their scope.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 794–803. PMLR, 2018. URL http://proceedings.mlr.press/v80/chen18a.html.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 2020. URL http://proceedings.mlr.press/v108/farajtabar20a.html.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00781. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Kendall_Multi-Task_Learning_Using_CVPR_2018_paper.html.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. 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*, SOSP '23, pp. 611–626. Association for Computing Machinery, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 18878–18890, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d27fdf2477ffbff837d73ef7ae23db9-Abstract.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *CoRR*, abs/2605.15155, 2026. doi: 10.48550/ARXIV.2605.15155. URL https://doi.org/10.48550/arXiv.2605.15155.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-sd: Skill-conditioned self-distillation for multi-turn LLM agents. *CoRR*, abs/2604.10674, 2026. doi: 10.48550/ARXIV.2604.10674. URL https://doi.org/10.48550/arXiv.2604.10674.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026. doi: 10.48550/ARXIV.2607.14777. URL https://arxiv.org/abs/2607.14777.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *CoRR*, abs/2602.08234, 2026. doi: 10.48550/ARXIV.2602.08234. URL https://doi.org/10.48550/arXiv.2602.08234.

An Yang et al. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled RLVR. *CoRR*, abs/2604.03128, 2026a. doi: 10.48550/ARXIV.2604.03128. URL https://doi.org/10.48550/arXiv.2604.03128.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. OPID: on-policy skill distillation for agentic reinforcement learning. *CoRR*, abs/2606.26790, 2026b. doi: 10.48550/ARXIV.2606.26790. URL https://doi.org/10.48550/arXiv.2606.26790.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *CoRR*, abs/2603.08561, 2026. doi: 10.48550/ARXIV.2603.08561. URL https://doi.org/10.48550/arXiv.2603.08561.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/64784.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860, 2023. doi: 10.14778/3611540.3611569. URL https://www.vldb.org/pvldb/vol16/p3848-huang.pdf.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-reflective self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.28076*, 2026. doi: 10.48550/ARXIV.2607.28076. URL https://arxiv.org/abs/2607.28076.

Appendix Contents

A Additional Related Work 13

A.1 On-policy and skill-conditioned distillation 13

A.2 Evolving skills and reflection memory 14

A.3 Conflicting gradients and projection 14

A.4 Agent RL and GRPO-style policy optimization 14

B Experimental Details and Additional Results 15

B.1 Datasets and Tasks 15

B.2 Baselines 15

B.3 Implementation and Evaluation Details 17

B.4 Additional Experiments Across Model Scales 20

C Supplementary Analyses 20

C.1 Motivation Diagnostics 20

C.2 Token Objectives and Mixing Coefficients 21

C.3 Task-level Projection Intervention 21

C.4 Sensitivity to Optimization Choices 22

C.5 Training Dynamics 24

C.6 Computational cost 25

C.7 Statistical Significance 26

C.8 Additional Case Analyses 27

C.9 Token-Level Effects of Projection 30

A Additional Related Work

A.1 On-policy and skill-conditioned distillation

On-policy distillation trains a student on its own generated sequences while a teacher supplies token-level targets, reducing the distribution mismatch between training and inference. Agarwal et al. (2024) develop generalized knowledge distillation with a separate teacher. On-policy self-distillation instead uses different contextual views of a model: Zhao et al. (2026) condition the teacher on privileged information, such as verified reasoning traces, while the student sees the original question. For multi-turn agents, Skill-SD (Wang et al., 2026) derives dynamic skills from completed trajectories and uses an importance-weighted reverse-KL objective with a periodically synchronized teacher. These methods supply dense supervision on student-generated trajectories.

SDAR (Lu et al., 2026) combines GRPO with skill-conditioned distillation and modulates token-level supervision using a detached sigmoid gate. Its experimental protocol draws on the SkillBank of SkillRL (Xia et al., 2026), which learns a hierarchical skill library and evolves it with the policy. SDAR also evaluates retrieved-skill adaptations of Skill-SD and OPSD, together with prompt-based skill injection. Our three host recipes—Skill-SD, GRPO+OPSD, and SDAR—follow the agent-training implementations described in Section 3 and Appendix B. In particular, the retrieved-skill k_3 host follows the adaptation in Lu et al. (2026), which differs from the original Skill-SD dynamic skill-generation protocol. A token-level distillation gate changes the input gradient, whereas CORD's projection gate decides when to modify the relationship between the two accumulated gradients. The two operations can therefore be composed.

Another approach incorporates teacher information directly into the advantage or policy-loss signal. RLSD (Yang et al., 2026a) uses detached teacher–student log-probability differences to reweight token advantages while preserving their signs. This route modifies the policy update internally, without a separate distillation loss to combine.

A.2 Evolving skills and reflection memory

The guidance available to an agent can evolve with its experience. SkillRL expands and refines a hierarchical skill library using failure analysis (Xia et al., 2026). RetroAgent stores retrospective lessons from successful and failed attempts in a growing reflection memory, and combines retrieved lessons with progress-based intrinsic rewards (Zhang et al., 2026). Its RL-trained variant also jointly optimizes decision making and self-reflection. These approaches improve what the agent retains from its interactions and retrieves for later tasks.

Evolving guidance need not reside in a persistent library. SEED trains the policy to generate hindsight skills from its current trajectories and internalizes their guidance through joint RL and on-policy distillation (Wu et al., 2026). OPID constructs episode- and step-level skills whose induced log-probability shifts enter the policy update (Yang et al., 2026b). GRSD uses reflections from successful and failed rollouts to form group-level guidance and modulate advantages (Zheng et al., 2026). CORD operates at the subsequent combination of objective gradients, so its use is orthogonal to whether guidance comes from a fixed bank or evolves with the policy. Potential applications include combining reward and distillation gradients in SEED, or decision-making and reflection gradients in the RL-trained RetroAgent variant, while preserving their respective guidance mechanisms.

A.3 Conflicting gradients and projection

Multi-task loss balancing adjusts objective scales using gradient magnitudes or learned uncertainty (Chen et al., 2018; Kendall et al., 2018). Gradient projection instead changes directions and has a long history in multi-task and continual learning. For two conflicting task gradients, PCGrad (Yu et al., 2020) removes each gradient's component along the other before combining them. A-GEM (Chaudhry et al., 2019) uses an asymmetric projection: it modifies the current-task gradient to satisfy a single half-space constraint defined by the gradient estimated from a sampled episodic-memory batch. This constraint prevents an increase in that sampled memory loss under a first-order approximation. Orthogonal gradient descent (Farajtabar et al., 2020) projects updates away from a stored subspace of gradients of previous network outputs. CAGrad (Liu et al., 2021) balances progress on the mean task loss with the worst local improvement among tasks.

CORD reuses the single-half-space projection underlying A-GEM (Lemma 1). The protected reference in CORD comes from one of the two jointly trained objectives, and the projected candidate is added to that reference. Either objective can be protected. Conflict-only projection uses a half-space constraint, while the always-project variant imposes an orthogonality constraint even when the gradients agree. The agent experiments study how protection, triggering, loss construction, and aggregation affect learning. The main host comparisons protect GRPO with conflict-only projection, while the direction and gate analyses test both choices.

A.4 Agent RL and GRPO-style policy optimization

GRPO (Shao et al., 2024) estimates advantages relative to groups of sampled responses, avoiding a separate learned critic. PPO (Schulman et al., 2017) supplies the clipped policy surrogate used by these host implementations. The truncated importance weighting in our k_3 and ρ_{d} distillation objectives is related to the correction used in V-trace/IMPALA (Espeholt et al., 2018), although the distillation weight does not implement the full V-trace return estimator. GiGPO (Feng et al., 2025) improves multi-turn credit assignment by combining episode-level advantages with step-level comparisons among actions grouped by shared anchor states. These choices determine the reward-derived gradient. CORD operates after the two objective gradients have been accumulated for a mini-batch, so credit assignment and gradient combination address different parts of the update.

B Experimental Details and Additional Results

This section provides the task definitions, baseline configurations, and evaluation procedures used to study CORD. Our implementation extends the SDAR codebase (Lu et al., 2026), which builds on the multi-turn training framework *verl-agent* released with GiGPO (Feng et al., 2025). We retain this environment and skill interface for the three host recipes.

B.1 Datasets and Tasks

Table 5 summarizes the two environments. ALFWorld and WebShop require an agent to execute actions whose consequences determine which observations become available next. Both tasks connect token-level distillation to rewards obtained after several decisions.

Table 5: Task and environment configuration. A turn denotes one generated response followed by an environment interaction. Evaluation sizes describe the source task pools or episode assignments. Section B.3 specifies the fixed evaluation protocol. The WebShop catalogue size counts products, rather than training instructions.

Environment	Training source	Evaluation source	Turns	Metric
ALFWorld	3,553 training games	140 in-distribution games	50	Success rate
WebShop	Goal pool from a 1,000-product catalogue	500 worker–goal assignments	15	Score; success

ALFWorld. ALFWorld (Shridhar et al., 2021) provides text-based household tasks in which the agent navigates rooms, locates objects, and performs the operations specified by a goal. We use the TextWorld interface and the training split shared by SDAR and GiGPO. Evaluation uses the in-distribution split, whose games contain environments seen during training. The six task categories are Pick and Place (Pick), Examine in Light (Look), Clean and Place (Clean), Heat and Place (Heat), Cool and Place (Cool), and Pick Two and Place (Pick2). Success requires completing the full goal within 50 turns. We report the success rate for each category and the pooled success rate over evaluated episodes, so the overall score weights categories by their number of evaluated tasks.

WebShop. WebShop (Yao et al., 2022) asks the agent to find and purchase a product that satisfies an instruction, including constraints such as attributes, options, and price. We use the text interface and the small catalogue selected by the inherited environment configuration. The agent searches the catalogue, opens product pages, selects options, and purchases an item within 15 turns. The environment constructs goals from this catalogue: the wrapper reserves goal indices 0–499 for evaluation and samples training goals from the remaining indices. This index partition is interpreted within each worker's seeded goal ordering. We retain both the graded task score, which measures how closely the purchase satisfies the instruction, and strict success, which requires a task score of 1. Both are reported on a 0–100 scale.

B.2 Baselines

We instantiate Skill-SD, GRPO+OPSD, and SDAR in a common trainer. Each recipe combines the GRPO objective with a token-level auxiliary loss, and uses the same skill bank and teacher-input construction. Their differences can be expressed through three choices: the distillation coefficient λ, whether teacher-confidence weighting is enabled, and the OPD token objective. Table 6 states these choices explicitly.

Token objectives and confidence weights. For a sampled response token y_t and its interaction history x_t, define

$$s_t = \log \pi_\theta(y_t \mid x_t), \qquad q_t = \mathrm{sg}[\log \pi_\theta(y_t \mid x_t \oplus S(x_t))], \qquad \ell_t = s_t - q_t. \tag{12}$$

These log-probabilities follow Equation 2. The teacher receives skill context $S(x_t)$. Its scores q_t and the rollout log-probabilities s_t^{old} are detached. The capped on-policy ratio is

$$\rho_{\mathrm{on},t} = \min\left\{\exp\left(\min\{s_t - s_t^{\mathrm{old}}, 10\}\right), 2\right\}. \tag{13}$$

The token objectives used by the host recipes and loss analysis are

$$\psi_t = \begin{cases} \rho_{\mathrm{on},t}\big(\exp(\min\{-\ell_t, 20\}) - 1 + \ell_t\big), & k_3, \\ \rho_{\mathrm{on},t}\ell_t, & \rho_{\mathrm{d}}, \\ -\ell_t, & \text{SDAR-style } k_1. \end{cases} \tag{14}$$

The k_3 and ρ_{d} objectives retain the student-side gradient through $\rho_{\mathrm{on},t}$. For fixed sampled tokens, with neither ratio nor exponential cap active,

$$\frac{\partial \psi_t^{k_3}}{\partial s_t} = \rho_{\mathrm{on},t}\ell_t, \qquad \frac{\partial \psi_t^{\rho_{\mathrm{d}}}}{\partial s_t} = \rho_{\mathrm{on},t}(\ell_t + 1). \tag{15}$$

Thus k_3 has zero token derivative at matched teacher and student log-probabilities, while ρ_{d} retains an offset. The objectives can change the direction of the aggregate gradient, rather than only its scale. The loss and coefficient analyses examine the ρ_{d} log-ratio objective as an alternative to the host objectives in Table 6, which remain unchanged. The k_1 term follows the SDAR and GRPO+OPSD implementations. With fixed sampled tokens and detached q_t, its gradient is $-\nabla s_t$. Teacher scores affect this host variant through confidence weights when those weights are enabled. Confidence weights are $w_t = 1$ when disabled and $w_t = \mathrm{sg}[\sigma(-\beta\ell_t)]$ when enabled, where σ is the sigmoid function. These weights act before gradient accumulation, whereas the projection gate acts on the resulting gradient pair.

Loss aggregation. Let m_t mask padding and inactive response tokens. The standard host path forms a valid-token mean within each physical micro-batch B^m and applies its accumulation and distributed scaling α_m:

$$\mathcal{L}_{\mathrm{OPD}}(B) = \mathcal{A}_B[w_t\psi_t] = \sum_m \alpha_m \frac{\sum_{t\in B^m} m_t w_t \psi_t}{\sum_{t\in B^m} m_t}. \tag{16}$$

The sum is over nonempty micro-batches. For a complete mini-batch with M equally weighted micro-batches, $\alpha_m = 1/M$. Thus a token's effective weight also depends on its micro-batch's valid-token count. A global-token mean instead uses one denominator $\sum_{t\in B} m_t$ for the complete optimizer mini-batch. Appendix C.4 defines and compares micro-batch, global-token, and trajectory aggregation. The coefficient λ multiplies the aggregated loss separately from these weights.

Importance weighting of the host k_1 term would give $\psi_t = -\rho_{\mathrm{on},t}\ell_t$, which differs in sign from the ρ_{d} objective and is not used by the Base recipes in Table 6.

Table 6: Base recipes in the shared implementation. The OPD column specifies the token term, including importance weighting. Every Base configuration disables projection.

Base recipe	λ	Confidence weighting w_t	OPD token term ψ_t
Skill-SD	10^{-3}	1 (off)	Importance-weighted k_3
GRPO+OPSD	5×10^{-3}	1 (off)	k_1, without $\rho_{\mathrm{on},t}$
SDAR	10^{-2}	$\mathrm{sg}[\sigma(-5\ell_t)]$	k_1, without $\rho_{\mathrm{on},t}$

Skill-SD. Our shared implementation of the Skill-SD recipe uses the importance-weighted k_3 surrogate and a coefficient of 10^{-3}. It retains the SDAR skill interface. In this implementation, the student-to-rollout importance ratio is capped at 2, and the exponential argument in the k_3 term is upper-clamped at 20 for numerical stability. These safeguards are applied identically to the Base and + CORD configurations. The auxiliary loss uses Equation 16 without teacher-confidence weighting.

GRPO+OPSD. GRPO+OPSD uses the plain teacher-minus-student log-probability term and uniform token weights. Its coefficient in our unified implementation is $\lambda = 0.005$. This matches the effective scale of the SDAR implementation's GRPO+OPSD configuration: the latter sets the outer coefficient to 0.01 and the sigmoid sharpness to zero, yielding a constant weight $\sigma(0) = 0.5$. Our gate-off branch uses $w_t = 1$, so absorbing this constant into λ preserves the auxiliary loss and its gradient. The numerical difference in coefficients therefore reflects the gate convention rather than a separate hyperparameter search.

SDAR. SDAR uses the same k_1 token term with $\lambda = 0.01$ and $\beta = 5$. Its detached sigmoid weight assigns greater weight to sampled tokens for which the skill-conditioned teacher has a higher log-probability than the student. This confidence weighting acts before differentiation. CORD subsequently acts on the accumulated auxiliary gradient, using its alignment with the GRPO actor gradient to decide whether to project. This separates token weighting from the subsequent choice of gradient combination.

Additional baselines. Our GRPO runs disable the distillation contribution. RLSD uses the skill-conditioned self-teacher with coefficient 0.5, a 100-step warmdown, and clipping parameter 0.2. We evaluate its final policies after 150 updates in four passes using the ordered ALFWorld assignments and the full WebShop task set described in Appendix B.3. The reference Vanilla, Skill-Prompt, and OPSD entries in Table 1 are reported values from the SDAR paper.

Fixed skill bank and teacher context. Following SDAR, we use the predefined SkillBank adopted from SkillRL (Lu et al., 2026; Xia et al., 2026). The bank contains general strategies and task-specific skills distilled from successful experience and failure analysis. We keep its contents fixed throughout training. Thus, we use SkillRL's bank as a source of guidance without its recursive bank-evolution procedure. In ALFWorld, the household task determines the task-specific skills. In WebShop, instruction keywords select relevant skills alongside the general strategies. Table 7 illustrates representative content.

The teacher is a skill-conditioned view of the current policy, not a separately trained model. It scores the student's generated tokens with gradients disabled. The student receives no additional skill context during rollout or evaluation. Each Base–CORD pair shares the bank, skill-selection rule, and auxiliary coefficient, so its comparison isolates the gradient-combination rule. CORD does not prescribe how the skill bank is selected or constructed.

Table 7: Representative content from the fixed skill bank, paraphrased for brevity. These entries guide the teacher without entering the student's rollout or evaluation input.

Skill context	Guidance
ALFWorld: cleaning	Take the object, clean it at a sink, move to the target location, and place it there.
WebShop: apparel	Select the required size and color, then check the price of the chosen variant before purchasing.
WebShop: search recovery	After two or three unproductive variant changes, return to search instead of continuing the same loop.

CORD configuration. The main CORD configuration protects the GRPO actor gradient, uses the Euclidean inner product, and projects once per complete mini-batch only when the two gradients conflict. The coefficient is constant, and the projection reference is the current protected gradient without EMA smoothing.

B.3 Implementation and Evaluation Details

Training configuration. Table 8 summarizes the common settings. We use Qwen2.5-3B-Instruct for the comparisons currently reported in Table 1, with FSDP (Zhao et al., 2023) for model training and vLLM (Kwon et al., 2023) for rollout generation. Each iteration samples 16 task groups with eight trajectories per group. Group members share the initial task and environment state, while their actions are sampled. The resulting trajectories are converted into step-level prompt–response records for optimization. Accordingly, the PPO mini-batch sizes below count these records rather than complete multi-turn episodes.

The group-relative advantage in Section 2 uses the host's response-record returns, with group statistics pooled across the records from all attempts at the same task. Each record's normalized advantage is broadcast over its valid response tokens. The policy-loss implementation additionally uses dual clipping for negative advantages: if $c_t = -\min(r_t\widehat{A}_t, \mathrm{clip}(r_t, 1-\varepsilon, 1+\varepsilon)\widehat{A}_t)$ is the token loss in Equation 1, it replaces c_t by $\min(c_t, -3\widehat{A}_t)$ when $\widehat{A}_t < 0$. The full minimized actor loss is

Table 8: Common training settings for the three base recipes and their CORD variants. Task groups and trajectories are counted before trajectories are expanded into step-level training records. Learning rates are constant after the indicated warm-up.

Setting	ALFWorld	WebShop
Training iterations	150	150
Task groups per rollout	16	16
Trajectories per group	8	8
Maximum prompt tokens	2,048	4,096
Maximum response tokens per turn	512	512
History retained (previous turns)	2	2
PPO mini-batch size (step records)	256	64
PPO epochs per rollout batch	1	1
Learning rate	10^{-6}	10^{-6}
Learning-rate warm-up fraction	0	0
Reference-policy KL coefficient	0.01	0.01
Invalid-action penalty	0.1	0.1
Training rollout temperature	1.0	1.0

$\mathcal{L}_{\mathrm{GRPO}} = \mathcal{L}_{\mathrm{pg}} + \beta_{\mathrm{ref}}\mathcal{K}_{\mathrm{ref}} - \beta_{\mathrm{H}}\mathcal{H}$, where $\mathcal{L}_{\mathrm{pg}}$ includes this safeguard and the regularization terms use the same valid-token aggregation. The frozen rollout probabilities and advantages carry no gradient.

We optimize with AdamW, using momentum coefficients $(0.9, 0.999)$, weight decay 0.01, a PPO clip ratio of 0.2, and gradient-norm clipping at 1. The frozen initial policy supplies the reference for the low-variance KL regularizer, with the same coefficient across recipes. The GRPO leg accumulated by CORD differentiates the complete host actor loss: the clipped policy surrogate plus reference-policy KL, and the entropy term when enabled. Protection therefore refers to this actor surrogate. The skill-conditioned auxiliary loss is computed separately. The interactive environments assign reward 10 for completing the goal and 0 otherwise. Invalid actions receive the additional penalties in Table 8. The graded WebShop task score is retained as an evaluation metric even though its training reward uses strict success.

Multi-turn inputs and gradient accumulation. Following *verl-agent*, each decision uses the current observation, the task description, and a bounded history of previous interactions. ALFWorld and WebShop retain the two most recent turns. Losses are computed on the generated responses. Environment observations and padding do not provide target tokens. We accumulate the GRPO actor and auxiliary gradients over the full PPO mini-batch with the selected aggregation weights before computing their inner product and applying the projection. In the fixed-coefficient implementation, the auxiliary backward already includes λ, so its buffer stores $\lambda \boldsymbol{G}_o$. After distributed gradient reduction and accumulation, a packed reduction supplies inner products and norms across parameter shards. The reported CORD configurations use Euclidean statistics. Numerical protection skips projection when the squared reference norm is at most ϵ and otherwise divides by $\|\boldsymbol{a}\|^2 + \epsilon$, with $\epsilon = 10^{-12}$. The ideal projection identities in Proposition 1 omit this numerical adjustment. Writing $A = \|\boldsymbol{a}\|^2$ and $c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle$, the regularized combined gradient $\boldsymbol{g}_\epsilon$ satisfies $\langle \boldsymbol{a}, \boldsymbol{g}_\epsilon \rangle = A + c\epsilon/(A + \epsilon)$ before floating-point error. For $c < 0$, this is below A and need not be positive. Skipping projection instead gives $A + c$. The combined gradient is clipped before the AdamW optimizer step. Task sampling and data order use recorded seeds, and the rollout implementation separately exposes the generation seed so that sampled decoding can be reproduced with a fixed worker configuration.

PCGrad two-objective adaptation. We apply PCGrad (Yu et al., 2020) to the host's fully accumulated actor gradient $\boldsymbol{u}$ and λ-weighted distillation gradient $\boldsymbol{v}$. When they conflict, each is projected against the other's original gradient, and the two projected gradients are summed. Aligned pairs are added unchanged. Each projection adds $\epsilon = 10^{-12}$ to its squared-reference-norm denominator and is skipped when that squared norm is at most ϵ. The runs in Table 2 use k_3 and the final-evaluation protocol below.

From the combined gradient to the parameter step. At an anchor θ, fix the sampled inputs and responses, masks, rollout and reference-policy probabilities, advantages, teacher scores, detached confidence values, aggregation weights, the stochastic realization, and λ. These define a local surrogate F_a with $\boldsymbol{a} = \nabla F_a(\theta)$. For the actual optimizer displacement Δ, write $p = -\boldsymbol{a}^\top \Delta$.

If F_a has an L-Lipschitz gradient on a neighborhood of the segment from θ to $\theta + \Delta$, then

$$F_a(\theta + \Delta) - F_a(\theta) \le -p + \frac{L}{2}\|\Delta\|^2. \tag{17}$$

Indeed, the fundamental theorem of calculus expresses the remainder as $\int_0^1 [\nabla F_a(\theta + t\Delta) - \boldsymbol{a}]^\top \Delta \, dt$. Cauchy–Schwarz and the Lipschitz condition bound its absolute value by $L\|\Delta\|^2/2$. For ideal SGD, $\Delta = -\eta \boldsymbol{g}_{\text{final}}$, strict descent follows when $L > 0$, $\boldsymbol{a}^\top \boldsymbol{g}_{\text{final}} > 0$, and

$$0 < \eta < \frac{2\boldsymbol{a}^\top \boldsymbol{g}_{\text{final}}}{L\|\boldsymbol{g}_{\text{final}}\|^2}. \tag{18}$$

This is a sufficient local condition, without an estimate of L. PPO or ratio-cap kinks can invalidate the smoothness assumption. Scalar gradient clipping preserves the sign of $\boldsymbol{a}^\top \boldsymbol{g}_{\text{final}}$ but changes the effective step size. AdamW's momentum, coordinate scaling, and weight decay change Δ, so the Euclidean projection identity alone does not establish $p > 0$. These statements concern the fixed surrogate and imply neither monotonic return improvement nor training convergence.

Fixed validation and final-evaluation tasks. We fix the validation tasks so that training progress is measured on the same problems at each evaluation. In the inherited ALFWorld protocol, we preserve an ordered list of 128 episode assignments drawn from the in-distribution evaluation pool. The list contains 91 distinct games. Retaining repeated assignments preserves their evaluation weights. Analyses identified as full-game evaluations use all 140 in-distribution ALFWorld games with fixed assignments. For WebShop, final evaluation uses 500 predetermined task assignments. Each comparison uses the same tasks, ordering, and decoding settings for Base and CORD, separating changes in the policy from changes in task composition.

Monitoring and final evaluation. We separate training-time monitoring from the measurements reported as final performance. When enabled, monitoring uses greedy decoding on a fixed set every five training iterations. Final evaluation uses the policy after 150 updates, without selecting the checkpoint with the highest intermediate validation score. This follows the *verl-agent* authors' clarification that final scores are obtained by repeatedly evaluating the step-150 checkpoint and averaging the results.[1]

The final-evaluation protocol performs four evaluation passes, generating one trajectory per task in each pass. Sampled evaluation uses temperature 0.4 and top-$p = 1$, without top-k truncation. If s_r is a reported metric in pass r, we report

$$\bar{s} = \frac{1}{K}\sum_{r=1}^{K} s_r, \qquad \sigma_{\text{eval}} = \sqrt{\frac{1}{K}\sum_{r=1}^{K}(s_r - \bar{s})^2}, \qquad K = 4. \tag{19}$$

Thus, $\bar{s} \pm \sigma_{\text{eval}}$ reports the mean and population standard deviation across four evaluation passes of the final policy. It does not estimate variation across training seeds, and averaging four passes does not change the metric into a best-of-four success rate. The final-evaluation comparisons use one training seed.

Relation to the reported SDAR scores. SDAR reports WebShop results on 128 fixed validation tasks (Lu et al., 2026). The paper does not specify the number of independent final-evaluation passes or report variability across such passes. We use 500 fixed assignments and report the mean and population standard deviation over four independently sampled passes of the final policy. These protocol differences limit direct comparison with the published scores, identified separately in Table 1. Our Base–CORD comparisons share task coverage and evaluation settings.

[1] *verl-agent*, issue #246, author clarification (June 7, 2026).

B.4 Additional Experiments Across Model Scales

Table 9 provides supplementary results with Qwen2.5-7B-Instruct for GRPO, RLSD, and the Skill-SD and SDAR host pairs. On ALFWorld, CORD raises Skill-SD success from 69.5% to 77.0% and SDAR from 68.9% to 76.8%, with displayed-mean gains of 7.5 and 7.9 percentage points. WebShop outcomes depend on the host: SDAR gains 4.7 score points and 2.9 success percentage points, whereas Skill-SD loses 2.9 score points and 1.8 success percentage points. These single-seed 7B results supplement the 3B comparison and show that the benefit depends on both task and host.

Table 9: **Final results on Qwen2.5-7B-Instruct.** ALFWorld success and WebShop score/success are on a 0–100 scale. Results use one training seed and four evaluation passes at temperature 0.4 after update 150. ALFWorld uses 128 fixed assignments and WebShop uses 500 fixed tasks. ± denotes population standard deviation across the four decoding passes. CORD protects RL with conflict-only projection. $\Delta = \text{CORD} - \text{Base}$ uses displayed means, in percentage points for success and score points for score. Red/green indicate increases/decreases. Bold/underline mark the largest/second-largest method means.

	ALFWorld							WebShop	
Method	Pick	Look	Clean	Heat	Cool	Pick2	**Avg.**	Score	Success Rate
GRPO	95.4±3.1	64.6±3.6	**83.0**±5.3	71.4±5.1	45.2±5.3	65.4±2.7	72.9±1.2	80.1±0.8	69.3±1.1
RLSD	95.4±3.1	72.9±9.1	67.0±3.9	42.9±5.1	32.1±6.2	69.2±3.8	65.6±1.8	**84.4**±0.3	71.0±0.6
Skill-SD	96.3±2.6	68.8±9.1	80.4±4.0	**75.0**±6.2	48.8±3.9	44.2±8.4	69.5±3.3	84.2±0.5	**71.6**±0.7
+ CORD	**99.1**±1.6	83.3±5.9	76.8±4.0	73.2±5.9	**51.2**±3.9	74.0±3.2	**77.0**±1.3	81.3±1.3	69.8±1.5
Δ	+2.8	+14.5	−3.6	−1.8	+2.4	+29.8	+7.5	−2.9	−1.8
SDAR	90.7±5.6	75.0±0.0	59.8±4.6	58.9±11.7	45.2±5.3	**77.9**±5.7	68.9±3.5	79.0±1.0	65.8±0.4
+ CORD	93.5±4.8	**87.5**±9.3	80.4±3.1	71.4±5.1	48.8±3.9	76.0±6.9	76.8±3.5	83.7±0.5	68.7±0.4
Δ	+2.8	+12.5	+20.6	+12.5	+3.6	−1.9	+7.9	+4.7	+2.9

We omit GRPO+OPSD from this 7B comparison because, for fixed sampled tokens, its k_1 auxiliary term uses constant token weights and a detached teacher. Its auxiliary gradient therefore does not vary with teacher log-probabilities and cannot express teacher–student mismatch in this implementation. SDAR's detached confidence weights still depend on this mismatch.

C Supplementary Analyses

C.1 Motivation Diagnostics

Gradient diagnostics. Figure 1 uses 2,741 WebShop/Qwen3-1.7B mini-batch gradient records collected with projection disabled and $\lambda = 0.001$. Negative RL–distillation cosine occurs in 1,322 records (48.2%). The left panel shows the 5th, 25th, 50th, 75th, and 95th percentiles of this subset, using linear interpolation. It compares the original cosines with those after analytic conflict-only projection of the same pairs before the optimizer. Aligned records are omitted from this panel. The middle panel reweights distillation to $\lambda = 0.005$ and compares DirectSum with conflict-only RL protection. Negative and nonnegative cosines define the conflict and aligned groups. RL-surrogate progress is normalized by the RL-gradient-alone value. The plotted coordinates are recovered from the archived vector figure, with tails clipped to the displayed range.

Training comparison. The right panel uses WebShop/Qwen2.5-3B-Instruct, k_3, $\lambda = 0.001$, and conflict-only distillation protection. It shows success on on-policy training trajectories over iterations 75–148 after 21-step smoothing, with limits 0.35–0.65. Curves average training seeds 0 and 1, and bands span the two seed curves. Two unprojected seed-0 runs are averaged before combining seeds. GRPO-only is trained separately at seeds 0 and 1.

C.2 Token Objectives and Mixing Coefficients

Table 3 reports the category-level results for the token-objective comparison in RQ2. Figure 5 adds the WebShop task score to the success-rate curves in Figure 3. These experiments use RL protection and conflict-only projection. Each point summarizes four evaluations of the step-150 checkpoint at temperature 0.4. The ALFWorld comparison uses the full-game protocol described in Appendix B.3.

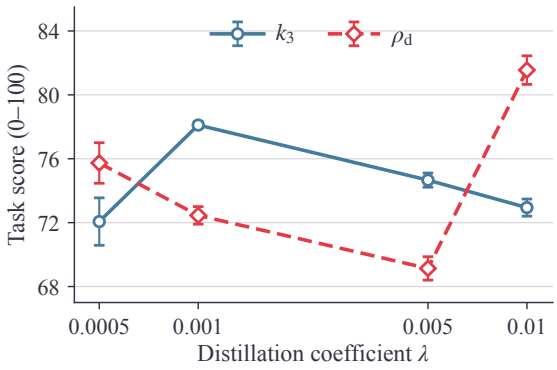


Figure 5: **Effect of the mixing coefficient on WebShop score.** The checkpoints and evaluation protocol are the same as in Figure 3. Error bars show standard deviation across decoding passes.

C.3 Task-level Projection Intervention

Measurement protocol. Figure 6 compares the two protection directions under local coefficient reweighting. For this diagnostic, $\boldsymbol{G}_r$ and $\boldsymbol{G}_o$ denote the two objective gradients aggregated over eight complete attempts at one task, before the outer coefficient λ is applied. We measure

$$100\,\frac{\|\boldsymbol{d}_{\mathrm{CORD}}(\lambda)-\boldsymbol{d}_{\mathrm{sum}}(\lambda)\|}{\|\boldsymbol{d}_{\mathrm{sum}}(\lambda)\|},\qquad \boldsymbol{d}_{\mathrm{sum}}(\lambda)=-(\boldsymbol{u}+\boldsymbol{v}(\lambda)),\quad \boldsymbol{d}_{\mathrm{CORD}}(\lambda)=-\boldsymbol{g}_{\mathrm{final}}.$$

Here $\boldsymbol{u}=\boldsymbol{G}_r$ and $\boldsymbol{v}(\lambda)=\lambda\boldsymbol{G}_o$, matching the weighting in Equation 4. The bold $\boldsymbol{d}$ symbols denote descent directions, hence the minus signs. The ratio requires a nonzero direct-sum direction. Within each environment, both columns reuse the same checkpoint, trajectories, and unweighted gradients, with $\boldsymbol{v}(\lambda)=(\lambda/\lambda_0)\boldsymbol{v}(\lambda_0)$. ALFWorld uses checkpoint 90, confidence-weighted k_1 with $\beta=5$, and native $\lambda_0=0.01$. WebShop uses checkpoint 75, k_3, and $\lambda_0=0.001$. We use the eight ALFWorld source tasks whose direct gradients were available, out of 32 source tasks, and all 16 WebShop source tasks. The ALFWorld subset follows a failed task-to-block gradient reconstruction check. Task indices retain the same order in both columns. These interventions compare changes to fixed gradient pairs, without training a new policy at each coefficient.

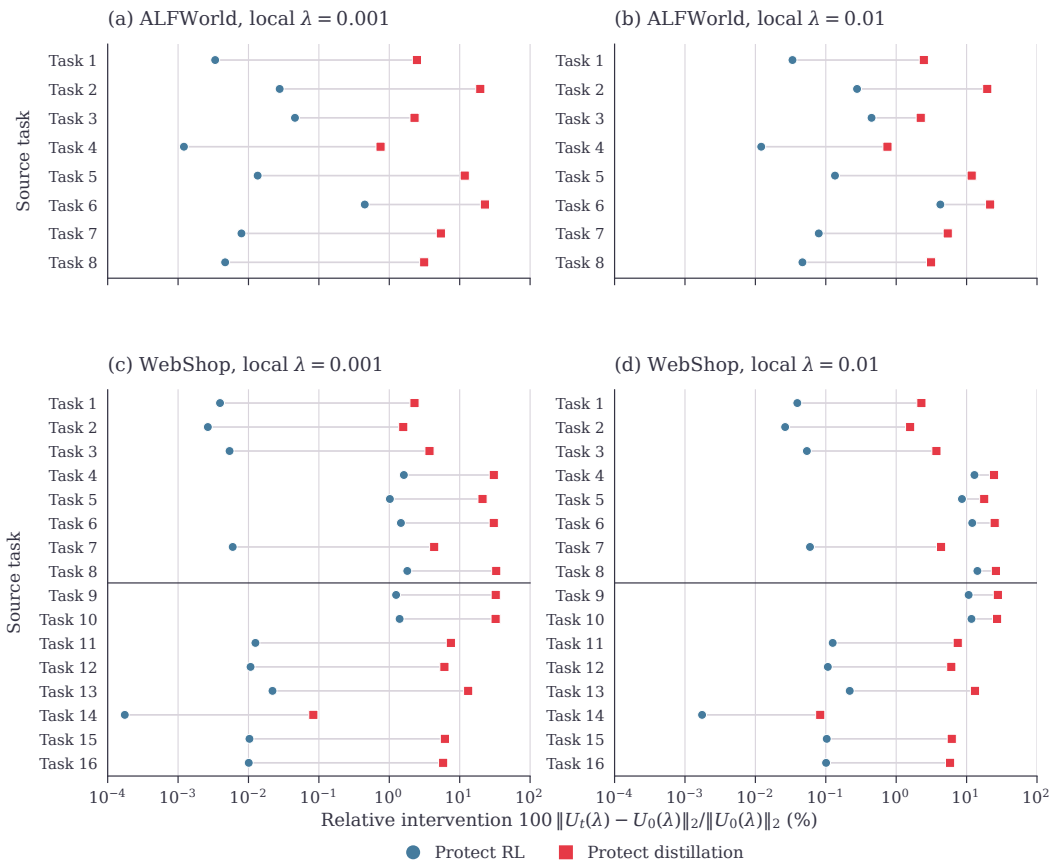


Figure 6: **Task-level projection intervention under local coefficient reweighting.** Each line connects the two protection directions for one source task under always projection. Columns reuse the same gradients at different coefficients. The horizontal axis reports the relative change in the combined update. Its labels U_0 and U_t denote $\boldsymbol{d}_{\text{sum}}$ and $\boldsymbol{d}_{\text{CORD}}$, respectively.

Dependence on the coefficient. For fixed nonzero loss gradients $\boldsymbol{G}_r$ and $\boldsymbol{G}_o$, write $\boldsymbol{v} = \lambda \boldsymbol{G}_o$ with $\lambda > 0$, and let ϕ be the angle between $\boldsymbol{G}_r$ and $\boldsymbol{G}_o$. Under the always gate used here, the projection correction has norm $\lambda \|\boldsymbol{G}_o\| |\cos \phi|$ when protecting RL and $\|\boldsymbol{G}_r\| |\cos \phi|$ when protecting distillation, by Equation 6. Thus the second correction is independent of λ at this fixed gradient pair, whereas the kept distillation term still scales with λ. These identities also hold on the conflicting branch of the conflict-only gate, whose aligned branch has zero correction.

C.4 Sensitivity to Optimization Choices

Table 4 reports the endpoints of the aggregation and mini-batch study in Section 4.3. Here we specify the continuation protocol and token weights, and examine the corresponding uncertainty in Figure 7.

Continuation protocol. Every configuration starts from the same ALFWorld/Qwen2.5-3B checkpoint at training step 90 and its AdamW state. We continue for five rounds with one training seed, gated k_1, $\lambda = .01$, $\beta = 5$, and conflict-only projection. Each run uses 640 episodes, including the same initial 128 episodes. Four response rows per GPU on eight GPUs give $1/2/4$ accumulation steps for optimizer mini-batch sizes $B = 32/64/128$. Evaluation uses 512 attempts over 91 in-distribution tasks at temperature 0.4, with success averaged across tasks. The 95% task-bootstrap intervals describe evaluation uncertainty for this seed. Paired contrasts are descriptive. The figure labels *Protect GRPO* and *Protect OPD* denote RL and distillation protection, respectively.

Aggregation weights. Let S be an optimizer mini-batch containing b valid response rows, where B is the configured mini-batch size. For objective j and token t in response row i, let $f_{it}^{(j)}$ include its token loss and any confidence weight: in particular, $f_{it}^{(2)} = w_{it}\psi_{it}$ for OPD. Then $F_j(S) = \sum_{(i,t)\in S} \omega_{it} f_{it}^{(j)}$ before projection. The three rules assign token weights

$$\omega_{it} = \begin{cases} 1/(MT_m), & \text{microbatch averaging } (L_1), \\ 1/T, & \text{global-token averaging } (L_2), \\ R/(NbL_{\tau(i)}), & \text{trajectory averaging } (L_3). \end{cases}$$

Here T_m and T count valid tokens in microbatch m and S, and M counts nonempty microbatches across GPUs. For trajectory averaging, R and N count response rows and complete trajectories in the rollout. Let $\tau(i)$ identify the complete trajectory containing row i, and let L_k count valid response tokens across all rows of trajectory k. When $b = R$, the trajectory-averaged objective is

$$F_{j,3} = \frac{1}{N}\sum_{k=1}^{N} \frac{1}{L_k} \sum_{(i,t):\,\tau(i)=k} f_{it}^{(j)},$$

so every complete trajectory has equal total weight. Mini-batches of response rows retain these complete-trajectory length weights.

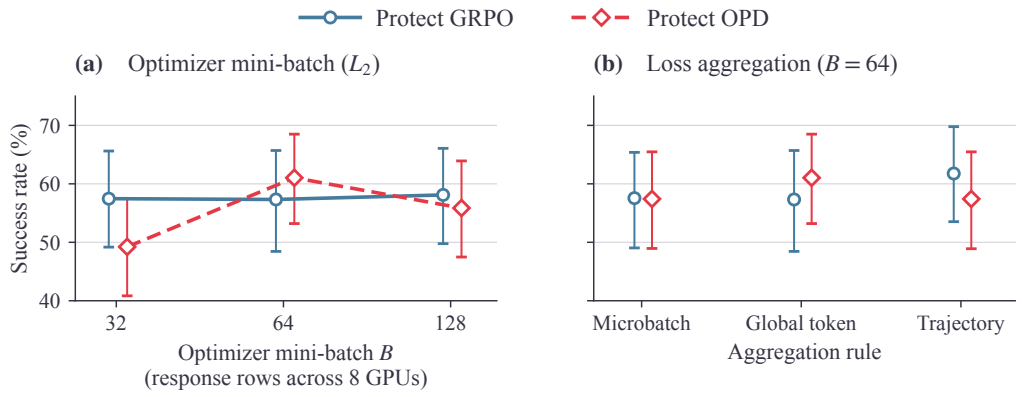


Figure 7: **ALFWorld continuation results with task-bootstrap uncertainty.** Panel (a) varies mini-batch size under global-token averaging (L_2), while panel (b) varies aggregation at $B = 64$. Error bars show 95% intervals over tasks under the protocol above.

Optimizer mini-batch size. At fixed L_2, RL-protected success spans 0.78 percentage points over $B \in \{32, 64, 128\}$, whereas distillation protection peaks at $B = 64$ (Figure 7a). At $B = 128$, distillation protection is 5.95 percentage points below no projection, with a paired 95% interval of $[-11.13, -0.96]$. Using $\boldsymbol{u} = \boldsymbol{G}_r$ and $\boldsymbol{v} = \lambda \boldsymbol{G}_o$ as in Equation 4, during conflict, correction norms are $\lambda\|\boldsymbol{G}_o\||\cos\phi|$ under RL protection and $\|\boldsymbol{G}_r\||\cos\phi|$ under distillation protection (Appendix C.3), where ϕ is the angle between $\boldsymbol{G}_r$ and $\boldsymbol{G}_o$. Small λ thus leaves the reverse correction unattenuated at fixed gradients. Logged median ratios $\|\boldsymbol{v}\|/\|\boldsymbol{u}\|$ of 0.009–0.013 describe the scale regime in which this contrast occurs. At fixed episode budget and learning rate 10^{-6}, increasing B reduces the number of updates from 616 to 300 to 150 under distillation protection. This sweep therefore changes both gradient estimation and the number of optimizer updates.

Loss aggregation. At $B = 64$, the directions nearly tie under microbatch averaging. Distillation protection leads by 3.71 points under global-token averaging, and RL protection leads by 4.35 under trajectory averaging (Figure 7b). All three paired intervals include zero, so this ordering remains descriptive. Aggregation reweights gradients before projection, changing the conflict test and protected objective. Global-token averaging (L_2) weights each valid token equally, while trajectory averaging (L_3) gives each complete trajectory equal total weight when all rollout rows are included.

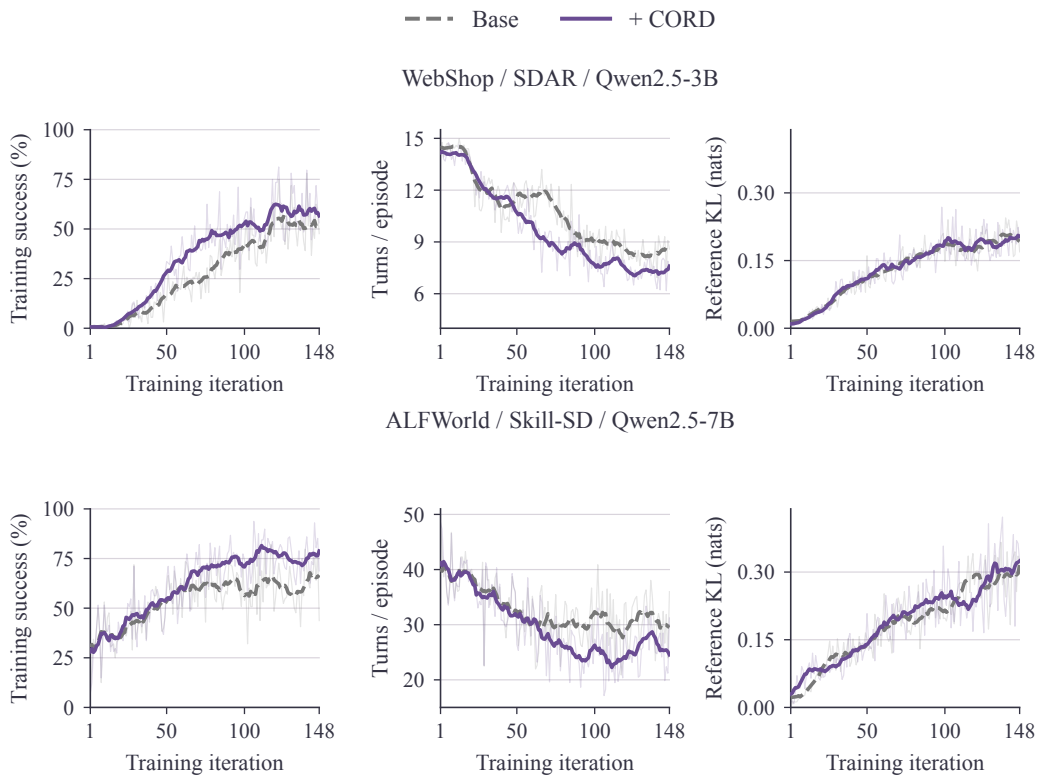


Figure 8: **Training success, interaction length, and reference KL.** Top: SDAR on WebShop with Qwen2.5-3B ($\lambda = 0.01$). Bottom: Skill-SD on ALFWorld with Qwen2.5-7B ($\lambda = 0.001$). Columns show rollout success, mean environment turns per episode, and the logged sample-based KL estimate to the frozen reference policy. Faint curves show raw observations, while darker curves show 11-iteration centered moving averages with shorter windows at the boundaries. Each method has one training seed, shown over the common range of iterations 1–148.

C.5 Training Dynamics

We examine how gradient combination affects rollout success, interaction length, and policy statistics during training. Each Base/CORD pair below shares its code revision, active training configuration, and training seed, with conflict-only RL protection enabled only for CORD. The plots include 3B settings and an additional 7B training setting. All measurements describe single-seed on-policy training trajectories and are separate from the final evaluations.

Reward progress and interaction length. Both settings show higher late-training success and shorter episodes with CORD (Figure 8). Using unsmoothed means over iterations 101–148, WebShop success increases from 50.0% to 57.1% while episode length decreases from 8.58 to 7.48 turns. In the 7B ALFWorld setting, success increases from 61.0% to 76.0% and episode length decreases from 30.67 to 25.34 turns. Mean reference-KL estimates are close within each pair: 0.188 versus 0.189 on WebShop and 0.274 versus 0.271 on ALFWorld. Thus, the shorter episodes accompany higher training success at a similar logged reference-KL level. Within these runs, the success and episode-length differences retain their signs over the alternative windows 76–148 and 124–148.

Token entropy and response length. CORD increases token entropy while slightly reducing mean response length (Figure 9). Over iterations 101–148, entropy increases from 0.780 to 0.925 nats, while mean response length decreases from 137.1 to 133.3 tokens. The entropy difference remains positive in both alternative windows. These observations distinguish changes in the token distribution from changes in response verbosity. Entropy is measured on each policy's own visited states, so it does not by itself measure exploration quality or establish a final-evaluation gain.

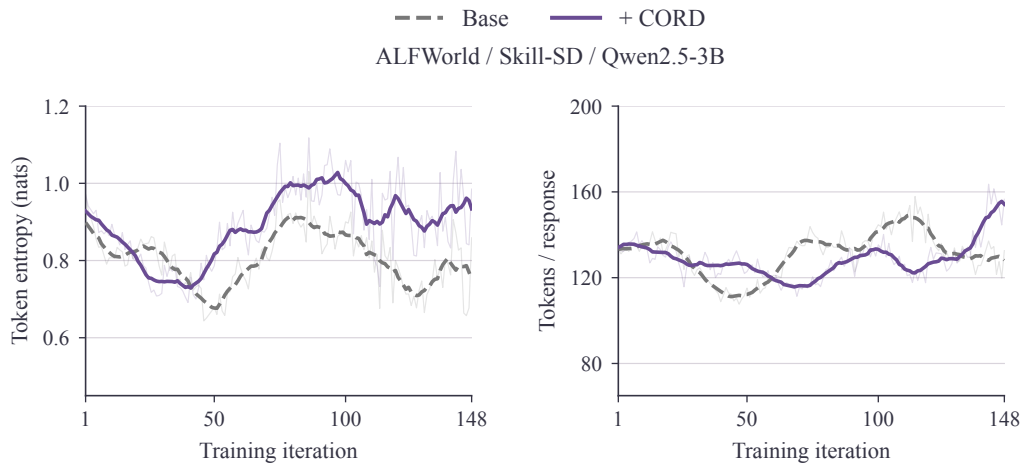


Figure 9: **Token entropy and response length on ALFWorld.** Skill-SD with Qwen2.5-3B at $\lambda = 0.001$. Left: token-mean policy entropy on the rollout responses, measured before the actor update. Right: mean response-token count, rather than tokens per complete episode. Each method uses one training seed. The raw overlays, 11-iteration smoothing, and common range of iterations 1–148 follow Figure 8.

C.6 Computational cost

Let d be the number of trainable parameters and let a training iteration contain K PPO mini-batches, each partitioned into M micro-batches. CORD accumulates the two objective gradients over each complete PPO mini-batch before applying the projection. Computing their inner products, norms, and combined update requires $O(d)$ arithmetic per mini-batch, or $O(Kd)$ per iteration. The additional gradient buffers require $O(d)$ storage, which is distributed across workers when gradients are sharded. The scalar inner products and norms can be communicated in a packed reduction whose message size does not grow with d.

For gradient evaluation, write F for the cost of a student forward computation and B_1, B_2 for the reverse-mode computations of the two objectives on a fixed micro-batch. If both objectives use the same student outputs, a shared-forward schedule has actor-update cost

$$T^{\text{actor}}_{\text{CORD}} = KM(F + B_1 + B_2) + O(Kd) + T_{\text{opt}},$$

where T_{opt} includes gradient clipping and optimizer updates. A fused-loss Base instead has cost

$$T^{\text{actor}}_{\text{Base}} = KM(F + B_{\text{sum}}) + T_{\text{opt}},$$

with B_{sum} denoting the reverse-mode computation of the weighted loss sum. These expressions use equal-sized micro-batches to simplify notation. For variable workloads, the corresponding costs are summed over micro-batches. Parameter and gradient communication belong to the gradient evaluation costs, separately from the packed reduction for projection.

Sharing the forward computation avoids evaluating identical student outputs twice. It does not make two objective-gradient evaluations equivalent to one backward pass on their summed loss: the latter supplies only the combined gradient, which generally does not determine the inner products needed by CORD. Both schedules have the same asymptotic order when the reverse-mode costs are comparable, while their constant factors depend on graph reuse, gradient-buffer handling, sharding, and hardware. Forward sharing, fused vector operations, and selective diagnostic collection are implementation opportunities rather than a fixed wall-clock guarantee. Teacher scores already computed for the auxiliary objective are reused.

Cost of a complete training iteration. Let A denote the actor-update time of an ordinary fused-loss Base, and let S include rollout and environment interaction, teacher, reference, and old-policy scoring,

reward and advantage computation, and the remaining work within a training iteration. For a fixed workload, if these shared stages remain unchanged and the actor-update multiplier is r_{actor}, then

$$R_{\text{iter}} = \frac{S + r_{\text{actor}} A}{S + A} = 1 + (r_{\text{actor}} - 1) f_{\text{actor}}, \qquad f_{\text{actor}} = \frac{A}{S + A}.$$

Thus, the actor-update multiplier is weighted by its share of Base iteration time. This comparison excludes initialization, periodic evaluation, checkpoint saving, and time outside the training iteration.

Component-based estimate. We evaluate an isolated shared-forward implementation on Qwen2.5-3B with eight A100-40GB GPUs, FSDP, activation checkpointing, a global PPO mini-batch size of 64, and a per-GPU micro-batch size of four. We replay the first, middle, and last complete mini-batches saved from one ALFWorld iteration. All replays use the same checkpoint, optimizer, and random states, with fixed $\lambda = 0.01$ and conflict-only projection with GRPO protection. After two warmup calls, each setting has seven timed repeats per mini-batch. Timings include the student forward and backward computations, communication, gradient combination, clipping, and AdamW, with GPU synchronization and the maximum elapsed time across workers. The ratio of summed per-mini-batch median times is $1.78\times$ relative to an ordinary Base that backpropagates the summed loss once.

To estimate the iteration-level impact, we use two recorded WebShop workload profiles with the same model, GPU type and count, and actor configuration. Both profiles cover the same 131 iterations between iterations 11 and 148 after excluding evaluation and checkpoint-saving iterations. The projection-off profile already collects both objective gradients for diagnostics. We therefore reconstruct ordinary Base actor time by dividing the logged actor time by its separately measured calibration ratio: 2.00 for the diagnostic path and 2.01 for the original CORD path. Applying the shared-forward multiplier to these reconstructed actor times gives Table 10. The estimated Base actor share is 25–27%, yielding an estimated iteration multiplier of 1.20–$1.21\times$. Applying the same calculation within the three consecutive windows 11–50, 51–100, and 101–148 gives 1.17–$1.22\times$.

Table 10: Component-based estimates for two WebShop workload profiles. Each row holds the recorded workload fixed for both methods. Shared-stage times are observed means, whereas actor times and iteration ratios are estimates for the ordinary Base and the shared-forward CORD implementation.

Recorded workload	Shared stages (s)	Base actor (s, est.)	CORD actor (s, est.)	Iteration ratio (est.)
Projection-off run	150.2	54.8	97.2	$1.21\times$
CORD run	147.7	50.0	88.8	$1.20\times$

These estimates assume that the actor ratios measured on the ALFWorld replay inputs transfer to the WebShop workloads and approximate the logged actor stage, which also includes worker and data-transfer overhead. The historical profiles and replay use different physical machines of the same GPU type. The estimates describe the recorded workloads with shared-stage costs held fixed and exclude policy-dependent changes in subsequent trajectories. They are not measurements of a complete training run with the optimized implementation.

C.7 Statistical Significance

The fixed-task, repeated-evaluation protocol in Appendix B.3 separates decoding variability from changes in task composition. We compare Base and CORD using two-sided Welch tests on the four evaluation-pass means for each checkpoint (Table 11). These tests summarize variation across decoding passes at one training seed. Holm correction is applied jointly to all 27 ALFWorld and WebShop comparisons.

Table 11: **Significance tests for Table 1.** Raw two-sided Welch p-values are reported to three significant digits. Bold † entries have Holm-adjusted $p < 0.05$.

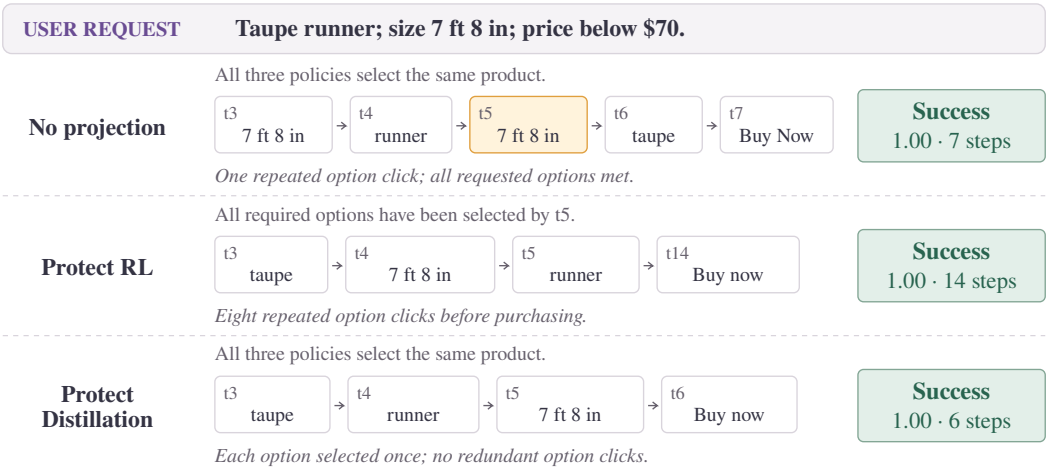
Metric	Skill-SD	GRPO+OPSD	SDAR
ALFWorld			
Pick	4.18×10^{-1}	3.32×10^{-1}	1.25×10^{-1}
Look	6.86×10^{-2}	1.35×10^{-1}	9.62×10^{-3}
Clean	5.85×10^{-1}	5.16×10^{-3}	8.39×10^{-3}
Heat	7.67×10^{-1}	1.79×10^{-2}	8.01×10^{-2}
Cool	$\mathbf{1.88\times10^{-4}}^{\dagger}$	9.15×10^{-3}	4.23×10^{-2}
Pick2	1.00×10^{-1}	$\mathbf{2.14\times10^{-4}}^{\dagger}$	2.33×10^{-2}
Average	3.16×10^{-1}	$\mathbf{2.88\times10^{-4}}^{\dagger}$	1.18×10^{-1}
WebShop			
Score	$\mathbf{9.09\times10^{-4}}^{\dagger}$	$\mathbf{1.05\times10^{-3}}^{\dagger}$	1.28×10^{-1}
Success rate	$\mathbf{2.20\times10^{-3}}^{\dagger}$	6.95×10^{-1}	2.84×10^{-3}

C.8 Additional Case Analyses

These four cases extend Section 4.4 using Qwen2.5-3B-Instruct, k_3, step-150 checkpoints and greedy decoding, with $\lambda = 0.005$ for WebShop and 0.001 for ALFWorld. The unprojected variant adds the gradients directly, while both protected variants use conflict-only projection. The ALFWorld tasks and decoding settings match, but training used eight GPUs for distillation protection and sixteen for the other checkpoints.

WebShop: repetition and attribute completion. In the rug case (Figure 10a), all three policies buy the same product and options with score 1.00, but the policy with RL protection makes eight repeated option clicks and uses 14 steps, compared with six under distillation protection. This reverses the ordering by total steps in Figure 4. For the garment (Figure 10b), all three finish in five steps. The policy with RL protection omits the requested size, scoring 0.80 compared with 1.00 for the other two. Thus, reaching a purchase promptly and completing its required attributes are separate aspects of behavior.

(a) Same purchase, different repetition



(b) Equal step counts, different attribute completion

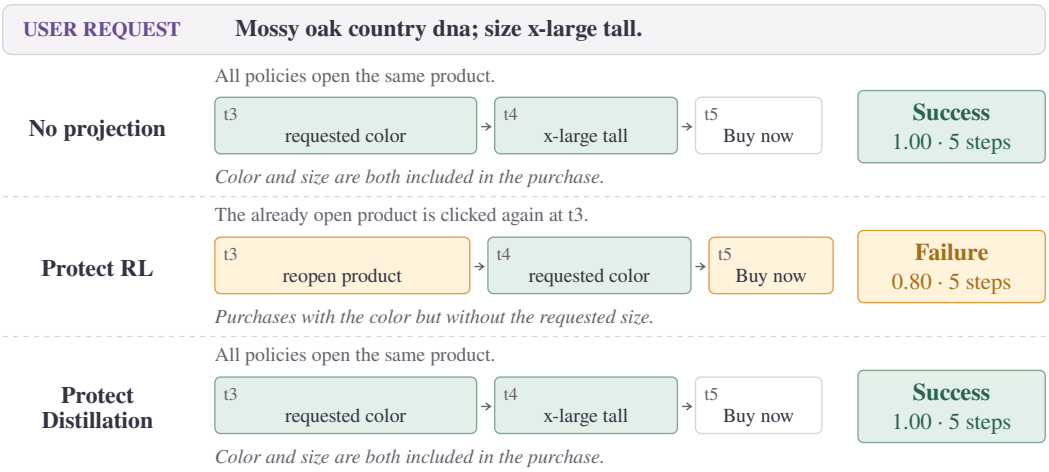


Figure 10: **Additional WebShop trajectories.** Trajectories retain the original step indices and use context lines to summarize omitted actions. Green marks requested attributes and amber marks repeated or incomplete actions. Outcomes report the final score and total number of steps, with score 1.00 required for success.

ALFWorld: object state and completed subgoals. In Figure 11a, the unprojected policy repeatedly cools eggs and never heats one within 50 steps. Both projected policies finish with the same four-action heating-and-placement sequence, taking eight steps under RL protection and sixteen under distillation protection. The extra actions occur before this shared suffix, including five ineffective actions under distillation protection. Figure 11b reverses the length ordering. With distillation protection, the agent keeps the first clock on the dresser and completes the second delivery in 11 steps, compared with 25 steps after backtracking under RL protection. The unprojected policy attempts placement with the wrong held object or at the wrong location and ends with only one delivered clock. The shorter successful trajectory occurs under different protection directions in these two cases.

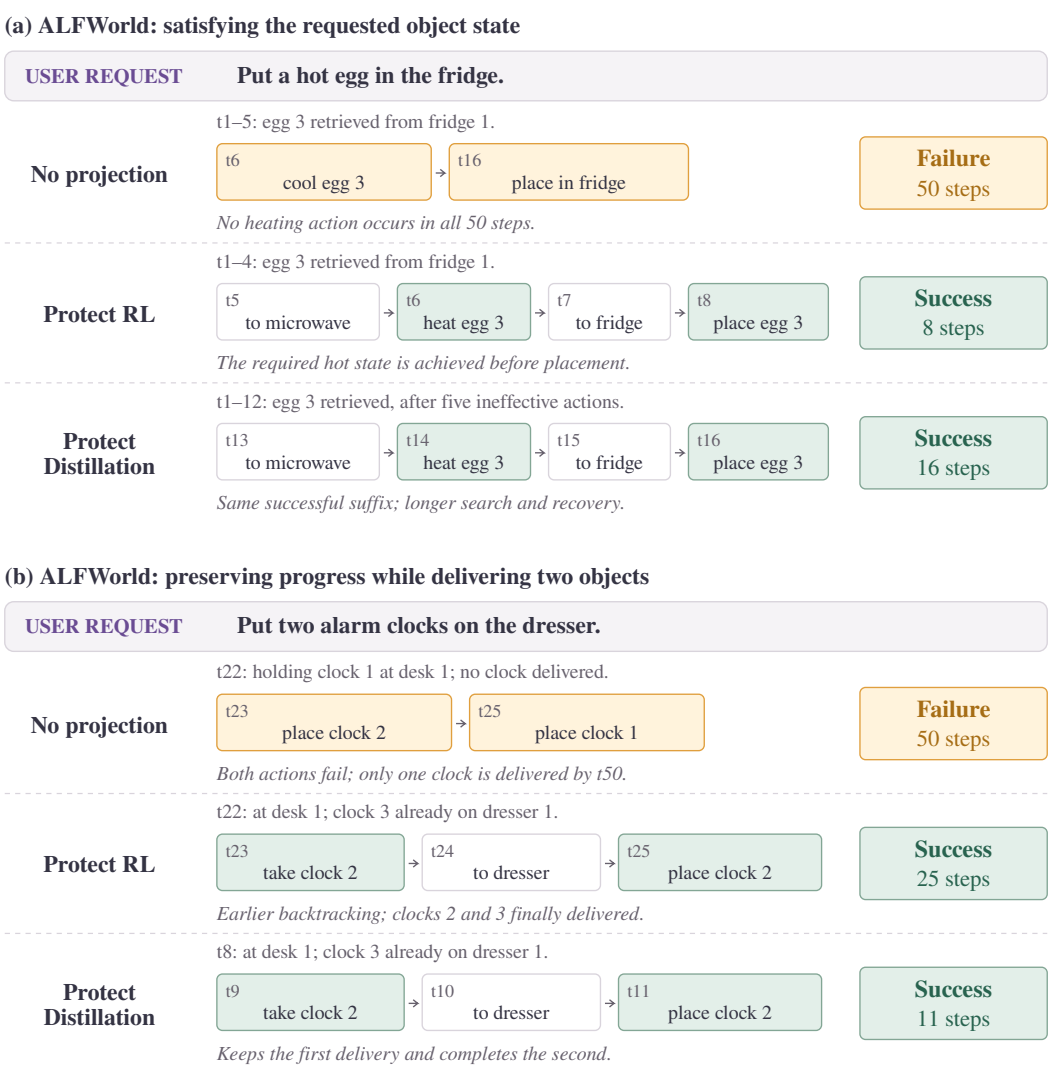


Figure 11: **Additional ALFWorld trajectories.** (a) Achieving the requested object state. (b) Completing two deliveries. Action labels are abbreviated, with original step indices and context for omitted intervals. "Clock" denotes an alarm clock and "place" denotes the environment's move action. Green marks goal-relevant manipulation, and amber marks incorrect or ineffective actions.

C.9 Token-Level Effects of Projection

Figure 12 follows a complete 12-step WebShop trajectory. From the Qwen2.5-3B step-75 checkpoint and identical AdamW states, we apply one update per rule on the same conflicting batch (OPD weight $\lambda = 0.001$), then rescore tokens at their saved prefixes. In the query, CORD with OPD protection retains more probability on the token `he` in `heather` and the fit term `men` than DirectSum. For `3`, it reduces suppression at the first size selection (step 5), but reverses DirectSum's increase when the choice repeats (step 7). This parameter-space projection produces context-dependent token changes, including an adverse increase for the unlisted `[next` token.



Figure 12: **Token probability changes along a complete trajectory.** Legend keys mark row positions, while outlines identify tabulated tokens, not update quality. CORD protects OPD. GRPO protection matches DirectSum at measured precision and is omitted. Dots denote spaces, and tag-boundary pieces are retained. The recorded outcome predates these updates.