UNDERSTANDING AND MITIGATING THE FRAGILITY OF SELF-REWARDING OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-rewarding optimization (SRO) promises to improve language models by enabling them to generate their own reward signals. Yet when and how SRO succeeds or fails remains poorly understood. We reveal that SRO is fundamentally fragile: even seemingly benign perturbations to training prompts can corrupt the self-generated preference signal and trigger iterative collapse. We demonstrate this through two novel, prompt-level attacks: a Re-Ranking attack that concentrates training on prompts where self-reward diverges from an oracle, and a Preamble attack that learns short prefixes inflating self-reward without any gain in oracle quality. To explain why these attacks succeed, we introduce self-reward fidelity (SRF), a metric quantifying on-policy agreement between self-evaluated and oracle rewards. We show that SRF is tightly coupled to answer quality through shared model parameters, and that controlled SRF perturbations induce a phase-transition behavior: SRO improves when SRF remains sufficiently high, but collapses once SRF drops below a model- and task-dependent threshold. This analysis motivates Projection Decoupling Defense, which suppresses SRO update components along directions to which the reward generator is locally sensitive, improving SRO robustness under adversarial conditions. This work sheds light on the fragility of SRO and charts promising directions toward making it more robust. The artifacts are available: `https://anonymous.4open.science/r/SRO-dilemma-5ED7`

1 INTRODUCTION

Aligning large language models (LLMs) with human intent is typically framed as an optimization problem (Rafailov et al., 2023; Bai et al., 2022a; Shao et al., 2024), where the objective is to produce outputs that score highly on quality, correctness, or safety. These alignment pipelines rely on external supervision, such as human preference labels or reward models trained on curated feedback, and optimize the policy via reinforcement learning or preference optimization. While effective, such approaches are expensive to scale and fundamentally limited by the availability, consistency, and coverage of external supervision.

Self-rewarding optimization (SRO) (Huang et al., 2023; Liu et al., 2024; Wu et al., 2025; Zhou et al., 2025; Wang et al., 2025b;a; Yuan et al., 2024) offers an appealing alternative. Rather than relying on external reward models, SRO repurposes the LLM itself as both an answer generator and a reward generator, creating a self-contained loop that reduces dependence on human annotations and enables continual self-improvement. Yet empirical results are mixed: the same training recipe can improve performance in some settings while causing sharp degradation or outright collapse in others (Zhang et al., 2025; Yuan et al., 2024; Herel & Mikolov, 2024). Despite growing interest, the factors governing SRO's effectiveness remain poorly understood.

We begin from this failure mode. If SRO is to serve as a scalable alignment mechanism, it should remain stable under variations in training data, such as reordering, filtering, or adversarial modification of prompts. We show this is not always the case. We introduce two prompt-level adversarial attacks that preserve the SRO training recipe while perturbing only the prompts seen by the self-rewarding loop. The first, a Re-Ranking attack, exploits dataset knowledge to prioritize questions where the reward generator is most unreliable, concentrating training on high-disagreement regions. The second, a Preamble attack, learns a short prompt prefix that systematically inflates self-reward scores without any corresponding oracle improvement, thereby poisoning the preference signal. Across

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.

SRO methods (Huang et al., 2025; Wang et al., 2025b), LLMs, and tasks, both attacks substantially degrade performance.

These attacks raise a critical question: what property of the SRO pipeline makes such collapses possible? We answer this by analyzing self-reward fidelity (SRF), the on-policy agreement between the model's self-evaluated and oracle rewards. SRF is not an isolated reward-model metric: because the same parameters implement both answer generation and reward generation, answer quality and self-reward fidelity are inherently coupled. By explicitly controlling SRF through targeted reward perturbations while holding the base model and training recipe fixed, we observe a phase-transition behavior: SRO improves when on-policy reward agreement remains sufficiently high, but degrades or collapses once fidelity falls below a model- and task-dependent threshold.

In response, we propose Projection Decoupling Defense to decoupling the effect of answer-policy updates on self-evaluation. We hypothesize that collapse is exacerbated when the same parameters are updated in correlated directions for both answering and judging. We realize this idea using verdict-score gradients on a small probe set to identify directions to which the reward generator is locally sensitive. We then suppresses the components of SRO updates along those directions. This targeted intervention breaks harmful feedback loops and substantially improves SRO robustness under adversarial reward conditions, while preserving its benefits in benign settings.

To the best of our knowledge, this is the first work to explore the fragility of SRO through prompt-level adversarial attacks and to chart principled directions for improving its robustness. Our contributions are as follows:

- We expose the fragility of SRO through Re-Ranking and Preamble attacks that manipulate only the prompt distribution yet reliably drive iterative collapse.
- We introduce self-reward fidelity (SRF) as a critical indicator, and show that controlled SRF perturbations induce a phase-transition-like behavior governing when SRO improves or collapses.
- We propose Projection Decoupling Defense, which suppresses SRO update components along directions to which the reward generator is locally sensitive, improving SRO robustness under adversarial conditions.

2 Related Work

Generative Reward Model. A generative reward model (GRM) treats evaluation as a generation task rather than outputting scalar rewards. Recent work explores both training and evaluation methods for GRMs. For training, Bai et al. (2022b); Gunjal et al. (2025); Yuan et al. (2025) replace opaque scalar rewards with principles or checklists that guide reasoning before producing rewards, while Lee et al. (2023); Yuan et al. (2024); Chen et al. (2025) use strong LLMs as judges to create preference signals or employ self-rewarding approaches. For evaluation, Shi et al. (2024); Wataoka et al. (2024); Chen et al. (2024); Bavaresco et al. (2024); Chehbouni et al. (2025) identify systematic biases, including position bias, verbosity bias, self-preference, and misinformation oversight, that distort judgments. Raina et al. (2024); Maloyan & Namiot (2025); Tian et al. (2025); Bodhwani et al. (2025) further examine the manipulability of GRMs and the reliability of their confidence estimates.

Preference Optimization. Reinforcement learning from human feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022; Bradley & Terry, 1952; Schulman et al., 2017) is a standard approach for aligning language models with human preferences: a reward model is trained from pairwise comparisons and the base model is then optimized against this proxy reward using policy-gradient methods such as Proximal Policy Optimization (PPO). However, proxy rewards can be brittle and costly to curate at scale, motivating direct preference optimization methods (Rafailov et al., 2023) that bypass explicit reward modeling and optimize policies directly from comparisons. Recent work (Azar et al., 2024; Xiao et al., 2024; Lee et al., 2025) provides unified and calibrated perspectives on these objectives, enabling more stable training. In parallel, AI-feedback approaches (Liu et al., 2023; Hong et al., 2024; Ethayarajh et al., 2024; Dong et al., 2023), such as Constitutional AI (Bai et al., 2022b) and RLAIF (Lee et al., 2023), reduce reliance on human annotations while preserving the benefits of preference-driven learning.

Self-Rewarding Optimization. Self-rewarding optimization (SRO) replaces human-labeled preferences with model-generated ones and can yield iterative alignment gains (Yuan et al., 2024). Recent work improves SRO through alternative self-supervision strategies, such as high-confidence self-

training and contrastive prompt distillation, and reliability mechanisms including meta-judging, reward-consistency regularization, and temporal decoupling of chosen/rejected samples (Huang et al., 2023; Liu et al., 2024; Wu et al., 2025; Zhou et al., 2025; Wang et al., 2025b;a). However, SRO often fails when self-rewards become exploitable through verbosity biases, reward hacking, or coarse answer-level judging on multi-step reasoning (Zhang et al., 2025; Yuan et al., 2024; Herel & Mikolov, 2024). In this work, we study the root cause of SRO's inconsistent effectiveness and explore how to stabilize SRO under adversarial reward conditions.

3 Exposing SRO Fragility

3.1 Self-Rewarding Optimization

As illustrated in Figure 1, the LLM $\mathcal{M}_\theta$ (parameterized by θ) serves both as the answer generator (policy) π_θ and the reward generator r_θ. SRO iteratively uses preference signals produced by r_θ to update π_θ. Specifically, at each iteration t, for sampled questions $\{x_n\}_{n=1}^N$ drawn from the data distribution $\mathcal{D}$, the current policy π_{θ_t} generates K candidate answers $\{y_{n,k}\}_{k=1}^K \sim \pi_{\theta_t}(\cdot|x_n)$ for each question; the current reward generator r_{θ_t} produces self-evaluated rewards $r_{\theta_t}(x_n, y_{n,k})$, which are converted into pairwise preferences $(y^+_{n,k}, y^-_{n,k})$; the shared parameters are then updated according to the SRO loss $\mathcal{L}_{SRO}$ with gradient as g_{SRO}.

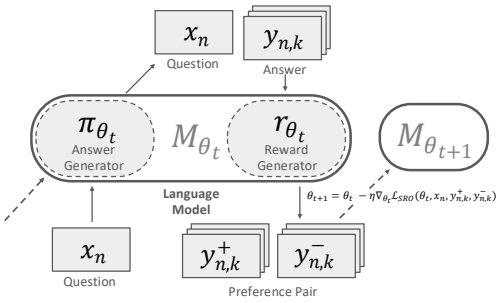


Figure 1: Self-rewarding optimization (SRO).

3.2 Adversarial Attacks on SRO

To understand the fragility of SRO, we examine how vulnerable SRO is to adversarial manipulations. We consider a threat model wherein the adversaries can only manipulate the prompts (i.e., questions) in the training set. This setting captures practical deployments where externally collected prompts, filtered data pools, or prompt templates can affect which examples enter the self-rewarding loop. We introduce two novel attack strategies that drive SRO into collapse by manipulating which questions enter the self-rewarding loop (detailed implementation deferred to §E.2).

Re-Ranking Attack. We first discuss how to leverage prior knowledge about the prompt dataset to trigger SRO collapse. For a given question x, we assess its difficulty for the target LLM π_θ as follows. We sample K answers $\{y_i\}_{i=1}^K$ from π_θ and collect rewards from both the oracle o and the reward generator r_θ for each question-answer pair (x, y_i). By comparing these rewards, we quantify x's difficulty as:

$$\kappa = \frac{1}{K}\sum_{i=1}^{K} \|o(x, y_i) - r_\theta(x, y_i)\|_2, \tag{1}$$

where higher κ indicates x is more challenging for r_θ.

By measuring question difficulty, we construct training batches that can trigger SRO collapse. Specifically, given a question set $\mathcal{D}_q = \{x_n\}_{n=1}^N$, for each question $x_n \in \mathcal{D}_q$, we first estimate its difficulty score κ_n following Eq. 1. We then construct a re-ranked dataset by adding Gaussian noise to prevent an overly rigid curriculum. For each x_n, we compute a perturbed score $s_n = \kappa_n + \sigma\epsilon_n$ with $\epsilon_n \sim \mathcal{N}(0, 1)$ and $\sigma = 0.1$. Sorting $\mathcal{D}_q$ in descending order of s_n yields an ordering that concentrates training on questions with large reward-oracle mismatch. Finally, we construct each training batch by sampling N questions from $\mathcal{D}_q$ without replacement according to s_n.

Preamble Attack. We further present a Preamble attack that learns short preambles to systematically inflate self-reward scores without improving oracle rewards. To this end, we train an attacker model to generate misleading preambles for each training question. Using DPO, we optimize the attacker model to induce the reward generator to assign inflated scores.

Specifically, we train a preamble generator f that prepends short text (preamble) to each input prompt to increase the self-reward score while maintaining oracle safety. For each prompt x_n in

the question set $\mathcal{D}_q$, we sample K candidate preambles $p_{n,k} \leftarrow f(\mathcal{I}, x_n)$ (where $\mathcal{I}$ is the preamble instruction) and generate responses from the fixed policy π_θ: a baseline response $y_n \sim \pi(\cdot \mid x_n)$ and preamble-conditioned responses $y^p_{n,k} \sim \pi(\cdot \mid p_{n,k}, x_n)$. We then evaluate their scores using both the reward generator r_θ and the oracle o. The learning signal is the preamble effect on the model's self-reward as

$$A_{n,k} = (r_\theta(p_{n,k}, x_n, y^p_{n,k}) - r_\theta(x_n, y_n)), \quad \text{s.t.} \quad k \in \mathcal{F}_n, \tag{2}$$

where $\mathcal{F}_n$ is a feasible set as $\mathcal{F}_n = \{k \in [K] : o(x_n, y^p_{n,k}) = o(x_n, y_n)\}$. We update f to maximize the expected advantage under prompts, sampled preambles, and policy rollouts, yielding a trained attacker $f^\star$. We construct the preamble dataset by sampling one preamble per prompt from $f^\star$ and prepending it to the original prompt: $\mathcal{D}_{(p,x)} = \{(p_n, x_n)\}_{n=1}^N$. We use $\mathcal{D}_{(p,x)}$ as the input distribution for SRO, disrupting training with questions augmented by preambles designed to exploit the reward model.

3.3 Experimental Setting

We empirically evaluate the effectiveness of both attacks, with the following setting.

LLMs. We primarily use four models from two families: `Llama-3.2-3B` and `Llama-3.1-8B` from the `Llama` family, and `Qwen-3-4B` and `Qwen-2.5-7B` from the `Qwen` family.

Datasets. We consider three distinct tasks: general knowledge on MMLU-Pro (Wang et al., 2024), logic reasoning on ARC (Clark et al., 2018), and safety alignment on SafeRLHF (Ji et al., 2024). From each dataset, we only use questions for SRO. We use `gpt-5` as the oracle evaluator to assess both model-generated answers and rewards.

SRO Methods. We evaluate two representative SRO methods: (i) Self-Rewarding Language Model (SRLM) (Yuan et al., 2024), which improves the LLM by generating answers and training on the model's own judgments iteratively, and (ii) CREAM (Wang et al., 2025b), which uses cross-iteration ranking agreement to weight DPO and reverse-DPO, stabilizing SRO.

Metrics. We measure answer quality (AQ), the expected oracle reward for model-generated answers:

$$\text{AQ}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)} o(x, y), \tag{3}$$

During training, we set temperature as 0.7 and top-p = 0.9 for the answer generator π_θ and temperature as 0.1 and top-p = 0.9 for the reward generator r_θ. We set the learning rate to $1e$-6 and use the Adam optimizer with a cosine scheduler and warm-up ratio of 0.1. For evaluation, we set temperature as 0.1 and top-p = 0.9 for both π_θ and r_θ.

3.4 Attack Results

Figure 2 shows that SRO can easily fail under prompt-level adversarial attacks. On safety alignment, clean SRLM is stable for larger models, but this stability disappears once the prompts are perturbed; similar trends on general knowledge and logical reasoning appear in §B.2. Re-Ranking and Preamble change only the prompt distribution, yet they drive large AQ drops among models and SRO methods. For SRLM, after three iterations, Llama-3.2-3B's AQ decreases from 88.6% to 39.9% under Re-Ranking and to 32.1% under Preamble.

4 Explaining SRO Fragility

4.1 Self-Rewarding Fidelity

We first introduce the metric of self-reward fidelity (SRF) as a critical success indicator for SRO. Intuitively, SRF measures the on-policy agreement between rewards from the reward generator r_θ of $\mathcal{M}_\theta$ and rewards from the oracle reward generator o. Formally,

$$\text{SRF}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)}[1 - |o(x, y) - r_\theta(x, y)|], \tag{4}$$

where $o(x, y), r_\theta(x, y) \in [0, 1]$ denote the normalized rewards from the oracle reward generator and the reward generator of $\mathcal{M}_\theta$, respectively. In the binary case, Eq. 4 can be simplified as:

$$\text{SRF}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)} \mathbb{1}_{[o(x,y)=r_\theta(x,y)]}, \tag{5}$$

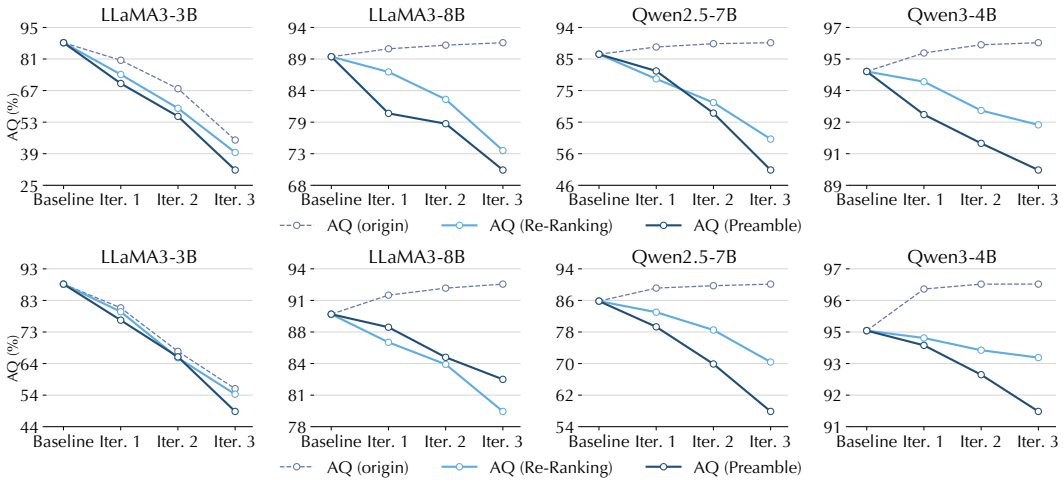


Figure 2: Answer quality (AQ) of varied LLMs on the safety alignment task under the Re-Ranking and Preamble attacks (upper: SRLM (Yuan et al., 2024); lower: CREAM (Wang et al., 2025b)). Notably, SRO, using either SRLM or CREAM, collapses under prompt-level adversarial attacks.

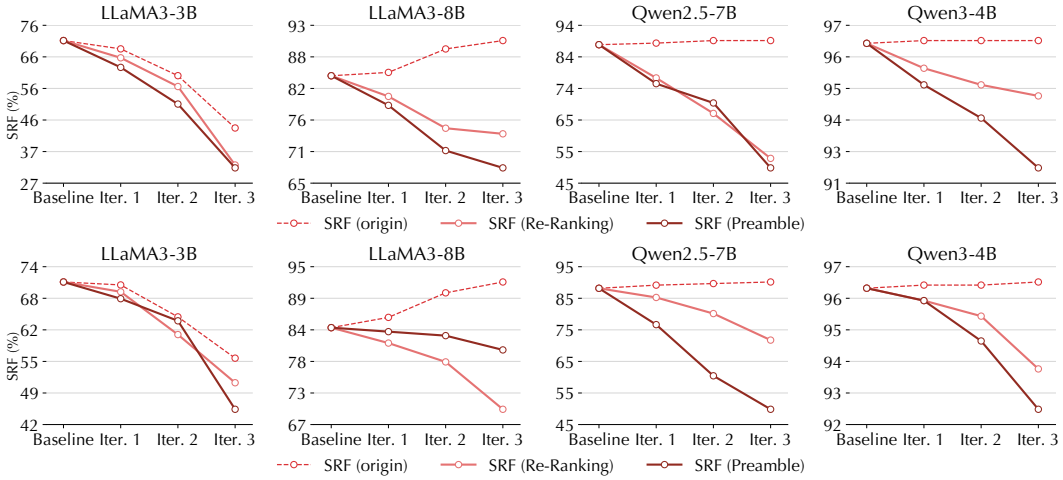


Figure 3: SRF closely tracks the (un)success of SRO. SRF of varied LLMs on the safety alignment task under the Re-Ranking and Preamble attacks (upper: SRLM (Yuan et al., 2024); lower: CREAM (Wang et al., 2025b)).

where $\mathbb{1}_p$ is an indicator function, which returns 1 if predicate p is true and 0 otherwise. Intuitively, SRF measures the on-policy agreement between the reward generator of $\mathcal{M}_\theta$ and the oracle reward generator o. Because SRF is evaluated on samples $y \sim \pi_\theta(\cdot|x)$, it can be seen as an on-policy measure of self-evaluation reliability. It measures whether the model can reliably judge the kinds of answers its current policy produces. Thus, SRF also reflects the compatibility between the answer distribution induced by π_θ and the reward judgments produced by r_θ.

4.1.1 SRO Collapse through SRF Lens

We first show that SRF serves as an effective indicator of attack success (§3.4). Figure 3 reports SRF trajectories corresponding to the attack results in Figure 2. SRF trajectories closely track AQ behavior: clean SRO succeeds when SRF is maintained or improved, while attacked or fragile runs exhibit progressive SRF degradation. This pattern confirms SRF as a reliable success indicator for SRO: high self-reward fidelity permits iterative self-improvement to proceed, whereas declining SRF signals corruption of the self-generated preference signal and portends imminent collapse.

Next, using SRF as a control mechanism, we explore the possible cause of SRO's fragility. To isolate the impact of SRF, we fix the underlying LLM and manipulate SRF by perturbing reward labels

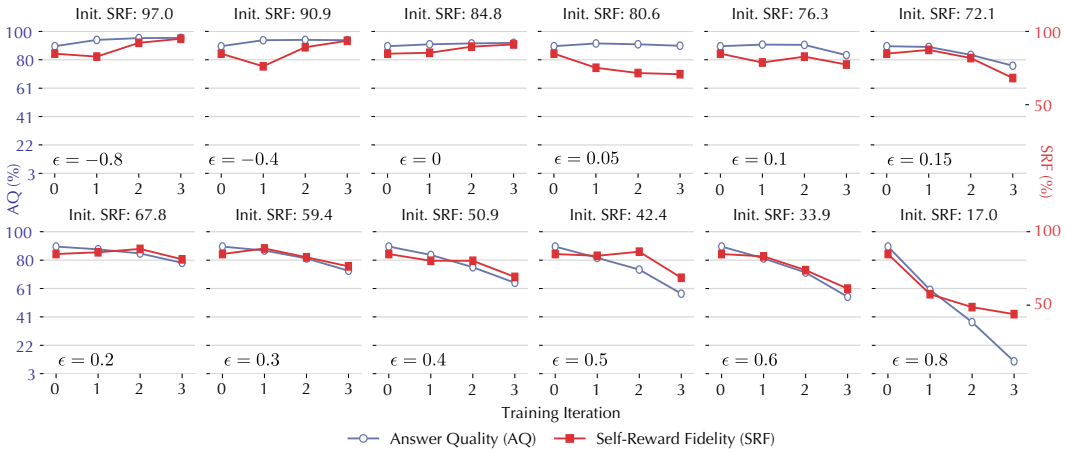


Figure 4: Impact of SRF perturbation on SRO dynamics: a phase transition phenomenon is observed for training `Llama-3.1-8B` on safety alignment, where SRO transits from performance improvement to degradation as initial SRF drops.

at a specified noise ratio ϵ. To this end, we construct a perturbed training set by flipping a subset of self-reward labels. Specifically, we sample $z \sim \mathcal{U}(0,1)$ and flip each reward according to the event $z < |\epsilon|$, yielding an expected flip rate $\mathbb{P}[z < |\epsilon|] = |\epsilon|$. In particular, for $\epsilon > 0$, we inject errors by selecting a sample where the self-reward agrees with the oracle ($r_\theta(x,y) = o(x,y)$) and flipping the label to its complement $1 - r_\theta(x,y)$. For $\epsilon \leq 0$, we instead target a mismatched example ($r_\theta(x,y) \neq o(x,y)$) and flip its label; for binary rewards, this acts as a purification operator that corrects a controllable fraction of incorrect labels. Formally, for $z \sim \mathcal{U}(0,1)$, we have:

$$\begin{cases} \{r_\theta(x,y)\} & \text{if } z \geq |\epsilon| \\ \{1 - r_\theta(x,y) \mid r_\theta(x,y) = o(x,y)\} & \text{if } \epsilon > 0 \wedge z < |\epsilon| \\ \{1 - r_\theta(x,y) \mid r_\theta(x,y) \neq o(x,y)\} & \text{if } \epsilon \leq 0 \wedge z < |\epsilon| \end{cases} \tag{6}$$

We measure the impact of SRF perturbation on SRO in Figure 4, training `Llama-3.1-8B` for safety alignment. We report the initial SRF at each noise level after perturbation. Notably, when $\epsilon < 0$ (denoising regime), SRO improves both metrics: AQ rises from 89.9% to 95.0%, and SRF increases from 84.8% to 95.0%. At $\epsilon = 0$ (no noise), both metrics remain high across iterations, with AQ increasing from 89.9% to 92.2% and SRF from 84.8% to 91.1%. In contrast, even mild noise ($\epsilon > 0$) degrades performance and halts improvement. At $\epsilon = 0.6$, AQ decreases to 55.6% and SRF to 61.2%. At $\epsilon = 0.8$, collapse occurs: AQ falls from 89.9% to 11.6% and SRF from 84.8% to 43.6%.

By manipulating SRF, we uncover a transition-phase phenomenon: above an empirical threshold range (approximately 76.3% in this setting with $\epsilon = 0.1$), SRO consistently improves model performance; below this range, SRO leads to performance degradation or even collapse. This pattern holds across other models (details in §B.1), though specific thresholds vary. These findings establish SRF as both a predictive indicator and a potential control mechanism for SRO effectiveness.

4.2 SRO Collapse through Analytical Lens

We now analyze SRO collapse by separating the two roles played by the shared model: the answer generator π_θ and the reward generator r_θ. Let L_{AQ} be the answer-quality loss and L_{SRF} a differentiable surrogate loss for self-reward fidelity, constructed from oracle-labeled reward-generation preferences. Details of these two losses are defered to §E.1. Since SRF is an on-policy agreement metric evaluated on samples from π_θ, whereas L_{SRF} is a training surrogate, we interpret local increases in L_{SRF} as local degradation of the reward generator's fidelity. We have the following proposition (detailed proofs and analysis in §A.1).

Proposition 4.1 (Transfer under Aligned Gradients)**.** *Assume L_{SRF} is differentiable at θ_t and consider one model update $\theta_{t+1} = \theta_t - \eta \nabla_\theta L_{\text{AQ}}$. If the answer-quality and self-reward gradients are positively aligned, $\langle \nabla_\theta L_{\text{AQ}}, \nabla_\theta L_{\text{SRF}} \rangle > 0$, then there exists $\delta > 0$ such that, for every step size $\eta \in (0, \delta)$, $L_{\text{SRF}}(\theta_{t+1}) < L_{\text{SRF}}(\theta_t)$.*

Through this fixed-batch, local gradient-transfer view, we observe that shared parameters couple the two roles: when $\nabla_\theta L_{\mathrm{AQ}}$ and $\nabla_\theta L_{\mathrm{SRF}}$ are positively aligned, an update that improves answer quality simultaneously reduces the SRF surrogate loss. This explains why SRO can initially improve both AQ and SRF when self-rewards remain reliable.

The same coupling, however, creates a conditional failure mode under noisy or adversarial self-rewards. Low SRF indicates that self-reward labels disagree more frequently with oracle labels, but does not by itself imply that the noisy AQ gradient reverses direction. However, if unreliable rewards corrupt the AQ preference signal sufficiently to make the noisy update locally misaligned with the SRF descent direction, shared parameters transfer this corruption into the reward-generator role. Formally:

Proposition 4.2 (Transfer under Misaligned Gradients)**.** *Let $\tilde{L}_{\mathrm{AQ}} = L_{\mathrm{AQ}} + \Delta_{\mathrm{AQ}}$ be the noisy AQ objective induced by unreliable self-rewards, where Δ_{AQ} captures the corruption. Consider one update $\theta_{t+1} = \theta_t - \eta \nabla_\theta \tilde{L}_{\mathrm{AQ}}$. Assume L_{SRF} has β_{SRF}-Lipschitz gradient in a neighborhood of θ_t. If the noisy AQ gradient misaligns with the SRF gradient, $\langle \nabla_\theta L_{\mathrm{SRF}}, \nabla_\theta \tilde{L}_{\mathrm{AQ}} \rangle \le -\gamma$ for some $\gamma > 0$, then for all $0 < \eta < \frac{2\gamma}{\beta_{\mathrm{SRF}} \|\nabla_\theta \tilde{L}_{\mathrm{AQ}}\|^2}$, $L_{\mathrm{SRF}}(\tilde{\theta}_{t+1}) > L_{\mathrm{SRF}}(\theta_t)$.*

This result exposes a one-step mechanism for iterative collapse: low-fidelity rewards corrupt the answer update, which in turn degrades the reward generator and renders future self-evaluations less reliable. A complete multi-step collapse theorem would further require linking SRF to gradient corruption across rounds and analyzing the resulting recurrence, which we leave for future work.

5 Mitigating SRO Fragility

5.1 Answer-Reward Decoupling

The analysis above suggests that corrupted answer updates can alter the reward generator through shared parameters, reinforcing errors over successive SRO iterations. This motivates a defense that limits the effect of policy updates on reward generation. We propose Projection Decoupling Defense, which attenuates update directions that affect verdict scores under a linear approximation constructed at the start of each SRO round, while retaining directions in its null space.

Specifically, let $\theta \in \mathbb{R}^d$ denote the shared parameters of the answer generator π_θ and reward generator r_θ. We fix a small probe set $\mathcal{C} = \{(x_i, y_i)\}_{i=1}^{N_c}$ of question–answer pairs, with answers sampled from the initial answer generator. For each probe (x_i, y_i), let $\ell_i(\theta) \in \mathbb{R}$ denote its differentiable verdict score: the reward generator's log-probability difference between positive and negative verdicts. We give the full definition in §C.1.

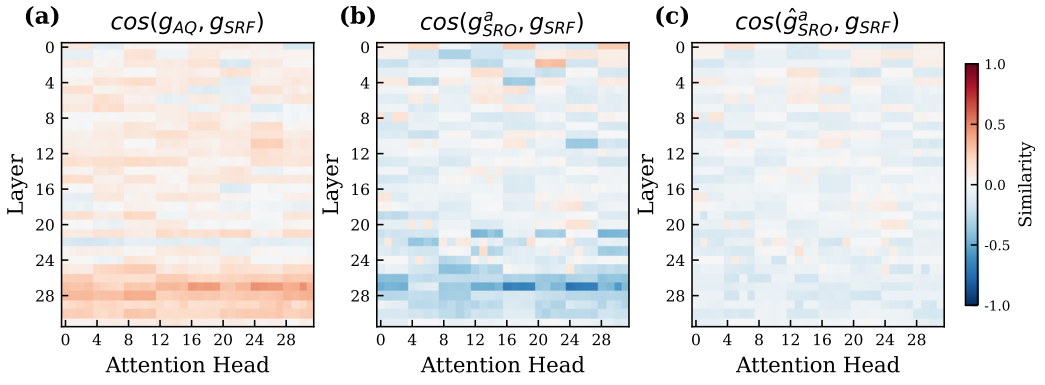


Figure 5: Cosine similarity with the self-reward fidelity gradient g_{SRF} across layers and attention heads: (a) answer-quality gradient g_{AQ}; (b) SRO gradient under the Re-Ranking attack g^a_{SRO}; (c) defended gradient $\hat{g}^a_{\mathrm{SRO}}$. Projection Decoupling Defense reduces negative alignment under attack.

At the start of each SRO round t, we evaluate the verdict-score gradients at θ_t and define their span:

$$g_i = \nabla_\theta \ell_i(\theta)|_{\theta=\theta_t}, \qquad \mathcal{S} = \mathrm{span}\{g_i\}_{i=1}^{N_c}. \tag{7}$$

Let Q have orthonormal columns spanning $\mathcal{S}$. Intuitively, $QQ^\top$ extracts the component of a parameter update that can change the verdict scores under a local linear approximation; the orthogonal component leaves them unchanged to first order. The full basis construction is given in §C.2.

During optimization of $\mathcal{L}_{\text{SRO}}$, we recompute the update gradient g_{SRO}. The verdict-score gradients determine which directions to protect, and g_{SRO} supplies the answer-training update. For exposition, we consider gradient descent with learning rate η and defended gradient as

$$\widehat{g}_{\text{SRO}} = g_{\text{SRO}} - \alpha Q(Q^\top g_{\text{SRO}}), \qquad \theta \leftarrow \theta - \eta \widehat{g}_{\text{SRO}}, \tag{8}$$

where $\alpha \in [0, 1]$ controls protection strength. Figure 5 visualizes gradient alignment across layers and attention heads. The answer-quality gradient is generally positively aligned with the SRF surrogate-loss gradient $g_{\text{SRF}} = \nabla_\theta L_{\text{SRF}}$ (panel a), whereas SRO gradients under the Re-Ranking attack exhibit negative alignment (panel b). Applying Projection Decoupling Defense weakens this negative alignment (panel c), consistent with reduced local interference with reward generation.

Algorithm 1: SRO with Projection Decoupling Defense

Input: Initial shared parameters θ_1; training questions $\mathcal{D}_q$; SRO loss $\mathcal{L}_{\text{SRO}}$; candidates per question K; SRO rounds T; probe count $N_{\mathcal{C}}$; protection strength $\alpha \in [0, 1]$; learning rate η

Sample $N_{\mathcal{C}}$ questions x_i from $\mathcal{D}_q$ and answers $y_i \sim \pi_{\theta_1}(\cdot \mid x_i)$; set $\mathcal{C} \leftarrow \{(x_i, y_i)\}_{i=1}^{N_{\mathcal{C}}}$;

for $t = 1$ **to** T **do**

- Sample K candidate answers per question in $\mathcal{D}_q$ from π_{θ_t};
- Score the candidates with r_{θ_t} and construct preference data $\mathcal{D}_t$ following the chosen SRO method;
- $g_i \leftarrow \nabla_\theta \ell_i(\theta)|_{\theta=\theta_t}$ for each fixed probe $(x_i, y_i) \in \mathcal{C}$, using Eq. 10;
- $J \leftarrow [g_1, \ldots, g_{N_{\mathcal{C}}}]^\top$, $G \leftarrow JJ^\top$;
- Eigendecompose $G = U\Lambda U^\top$, retaining positive eigenvalues in Λ and orthonormal eigenvectors in U;
- $Q \leftarrow J^\top U \Lambda^{-1/2}$;
- Optimize $\mathcal{L}_{\text{SRO}}$ on $\mathcal{D}_t$ from θ_t to obtain θ_{t+1}, computing g_{SRO} at each training step and filtering updates with the fixed Q as in Eq. 8;

Output: Defended shared parameters θ_{T+1}

Algorithm 1 integrates this update rule into iterative SRO. We first construct the probe set $\mathcal{C}$. In each round t, the current answer and reward generators produce preference data $\mathcal{D}_t$, and the verdict-score gradients at θ_t determine the basis Q. We then optimize $\mathcal{L}_{\text{SRO}}$ on $\mathcal{D}_t$, holding Q fixed and filtering each training update as in Eq. 8, to obtain the shared parameters θ_{t+1} for the next round.

5.2 Defense Results

We evaluate Projection Decoupling Defense on safety alignment across the four models in §3.4. All defended runs use $N_{\mathcal{C}} = 16$ probes and $\alpha = 0.5$. We examine both robustness to Re-Ranking and Preamble attacks and the effect of the defense on attack-free training.

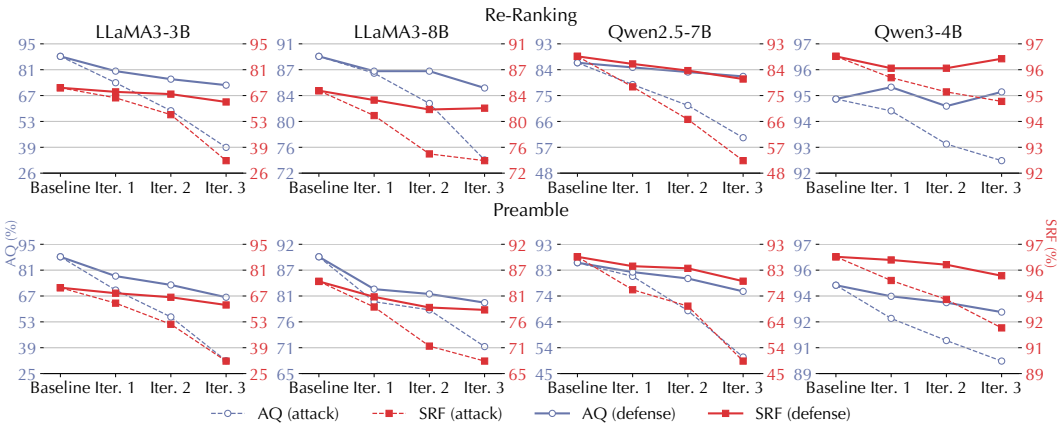


Figure 6: Defense against two attacks on SRLM (upper: ReRanking attack; lower: Preamble attack).

Robustness under attack. Figure 6 shows that the defense improves AQ under SRLM by 2.9–34.5 percentage points at iteration 3. Under Preamble, it raises `Llama-3.2-3B`'s AQ from 32.1% to

66.6% and SRF from 31.9% to 62.4%. These joint gains suggest that limiting the effect of policy updates on reward generation improves SRO robustness under attack. The defense also improves AQ under CREAM by 1.6–24.7 percentage points at iteration 3 (Appendix B.3, Figure 19).

Performance without an attack. Figure 7 shows that the defense largely retains the gains of clean SRO for the three stronger models. After three iterations, defended AQ reaches 91.9%, 89.9%, and 96.4% for `Llama-3.1-8B`, `Qwen-2.5-7B`, and `Qwen-3-4B`, respectively, only 0.2–0.3 percentage points below undefended training. For `Qwen-2.5-7B`, AQ starts at 86.6% and reaches 88.8%, 89.8%, and 90.1% without defense, versus 87.4%, 88.5%, and 89.9% with defense over three iterations. As undefended AQ approaches saturation, the defense continues improving and nearly matches its final AQ despite a smaller initial gain. For `Llama-3.2-3B`, the defense raises final AQ from 45.4% to 84.4% and SRF from 44.4% to 67.9%, though both remain below their initial values of 88.6% and 71.8%, respectively. These results suggest that the defense can also mitigate degradation during clean SRO while largely preserving gains in stable models.

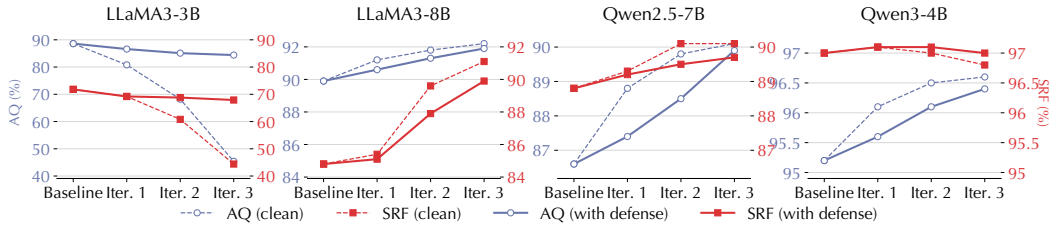


Figure 7: SRO iterations under attack-free conditions, with defense (solid) and without defense (dashed)

Defense ablations. We compare our defense with three baselines: normalized updates, locally norm-matched random projection, and Frozen Judge under Re-Ranking with SRLM (Figure 8). On `Qwen-2.5-7B`, our defense retains the highest AQ across all three iterations. Frozen Judge, the strongest baseline at iteration 3, drops sharply from 86.6% to 79.1% AQ at iteration 1 and recovers only slightly to 79.4% by iteration 3, remaining below our defense throughout training. Our defense finishes at 81.7% AQ, compared with 67.3% for normalization and 66.2% for random projection. Control definitions and additional results are in Appendix D.

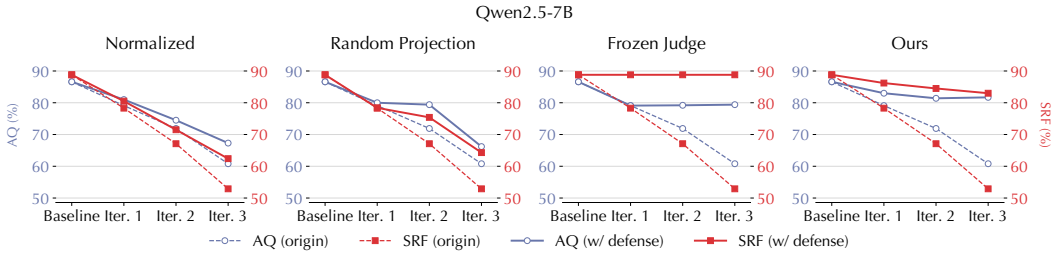


Figure 8: Defense ablations on `Qwen-2.5-7B` under Re-Ranking with SRLM.

6 Conclusion

This paper exposes the fragility of self-rewarding optimization (SRO) via Re-Ranking and Preamble attacks that manipulate only the training-question distribution yet drive iterative collapse. We then analyze the failure mechanism, identifying self-reward fidelity (SRF), the on-policy agreement between self-evaluated and oracle rewards, as a critical success indicator, and showing that shared answer/reward parameters amplify low-fidelity updates. This analysis motivates Projection Decoupling Defense, which uses verdict-score gradients to identify and attenuate SRO updates along directions that affect self-evaluation, substantially improving robustness under adversarial conditions. Our work opens three directions for future research: (i) developing adaptive defenses, (ii) understanding how SRF dynamics scale with model capability, and (iii) exploring whether similar phase transitions govern other forms of self-supervised training.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, drafting portions of the manuscript, and literature search and summarization. We reviewed all AI-assisted content, revised the generated text, and checked literature summaries and related claims against the original sources. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies vulnerabilities in self-rewarding optimization to improve model safety. We acknowledge that these findings could be misused to undermine model safeguards. Our defense is evaluated under specific experimental settings and does not guarantee safety beyond those settings.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed experimental settings and hyperparameters in §E.3. Our implementation is available at `https://anonymous.4open.science/r/SRO-dilemma-5ED7`.

REFERENCES

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, et al. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *arXiv preprint arXiv:2406.18403*, 2024.

Umesh Bodhwani, Yuan Ling, Shujing Dong, Yarong Feng, and Hongfei Li. A calibrated reflection approach for enhancing confidence estimation in llms. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pp. 399–411, 2025.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Khaoula Chehbouni, Mohammed Haddou, Jackie Chi Kit Cheung, and Golnoosh Farnadi. Neither valid nor reliable? investigating the use of llms as judges. *arXiv preprint arXiv:2508.18076*, 2025.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or llms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*, 2024.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

David Herel and Tomas Mikolov. Collapse of self-trained language models. *arXiv preprint arXiv:2404.02305*, 2024.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, 2024.

Chenghua Huang, Zhizhen Fan, Lu Wang, Fangkai Yang, Pu Zhao, Zeqi Lin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Self-Evolved Reward Learning for LLMs. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2025.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 1051–1068, 2023.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Qiu, Boxun Li, and Yaodong Yang. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. *arXiv preprint arXiv:2406.15513*, 2024.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Sangkyu Lee, Janghoon Han, Hosung Song, Stanley Jungkyu Choi, Honglak Lee, and Youngjae Yu. Kl penalty control via perturbation for direct preference optimization. *arXiv preprint arXiv:2502.13177*, 2025.

Aiwei Liu, Haoping Bai, Zhiyun Lu, Xiang Kong, Xiaoming Wang, Jiulong Shan, Meng Cao, and Lijie Wen. Direct large language model alignment through self-rewarding contrastive prompt distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9688–9712, 2024.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.

Narek Maloyan and Dmitry Namiot. Adversarial attacks on llm-as-a-judge systems: Insights from prompt injections. *arXiv preprint arXiv:2504.18333*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Vyas Raina, Adian Liusie, and Mark Gales. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7499–7517, 2024.

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

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

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. *arXiv preprint arXiv:2406.07791*, 2024.

Zailong Tian, Zhuoheng Han, Yanzhe Chen, Haozhe Xu, Xi Yang, Richeng Xuan, Houfeng Wang, and Lizi Liao. Overconfidence in llm-as-a-judge: Diagnosis and confidence-driven solution. *arXiv preprint arXiv:2508.06225*, 2025.

Yidong Wang, Xin Wang, Cunxiang Wang, Junfeng Fang, Qiufeng Wang, Jianing Chu, Xuran Meng, Shuxun Yang, Libo Qin, Yue Zhang, et al. Temporal self-rewarding language models: Decoupling chosen-rejected via past-future. *arXiv preprint arXiv:2508.06026*, 2025a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Zhaoyang Wang, Weilei He, Zhiyuan Liang, Xuchao Zhang, Chetan Bansal, Ying Wei, Weitong Zhang, and Huaxiu Yao. Cream: Consistency regularized self-rewarding language models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2025b.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in llm-as-a-judge. *arXiv preprint arXiv:2410.21819*, 2024.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11548–11565, 2025.

Teng Xiao, Yige Yuan, Huaisheng Zhu, Mingxiao Li, and Vasant G Honavar. Cal-dpo: Calibrated direct preference optimization for language model alignment. *Advances in Neural Information Processing Systems*, 37:114289–114320, 2024.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024.

Youliang Yuan, Qiuyang Mang, Jingbang Chen, Hong Wan, Xiaoyuan Liu, Junjielong Xu, Jen-tse Huang, Wenxuan Wang, Wenxiang Jiao, and Pinjia He. Curing miracle steps in llm mathematical reasoning with rubric rewards. *arXiv preprint arXiv:2510.07774*, 2025.

Shimao Zhang, Xiao Liu, Xin Zhang, Junxiao Liu, Zheheng Luo, Shujian Huang, and Yeyun Gong. Process-based self-rewarding language models. *arXiv preprint arXiv:2503.03746*, 2025.

Xin Zhou, Yiwen Guo, Ruotian Ma, Tao Gui, Qi Zhang, and Xuanjing Huang. Self-consistency of the internal reward models improves self-rewarding language models. *arXiv preprint arXiv:2502.08922*, 2025.

A PROOFS

A.1 DETAILED GRADIENT-TRANSFER ANALYSIS

We start from a separated view of the answer generator π_θ and the reward generator r_θ. The former is the policy being trained, while the latter is the internal reward function that guides training. Let L_{AQ} denote the loss used to optimize answer quality, and let L_{SRF} denote a differentiable surrogate loss for self-reward fidelity. The SRF metric is an on-policy agreement between r_θ and the oracle on samples from π_θ, whereas L_{SRF} is the DPO-style reward-generation loss used for optimization. In this appendix, we analyze $L_{\mathrm{SRF}}(\theta; B_t)$ on a fixed batch B_t sampled at the current iterate θ_t and treat local increases in this surrogate as local degradation of reward-generation fidelity. We omit the change in the on-policy sampling distribution induced by future updates.

Proof. (Proposition 4.1) Since L_{SRF} is differentiable at θ_t, for any sufficiently small perturbation h we have the first-order expansion

$$L_{SRF}(\theta_t + h) = L_{SRF}(\theta_t) + \langle g_{SRF}, h\rangle + r(h),$$

where the remainder satisfies $r(h)/\|h\| \to 0$ as $\|h\| \to 0$. Set $h = -\eta g_{AQ}$ and let $c \coloneqq \langle g_{AQ}, g_{SRF}\rangle$. By assumption, $c > 0$. Then

$$\begin{aligned} L_{SRF}(\theta_{t+1}) - L_{SRF}(\theta_t) &= L_{SRF}(\theta_t - \eta g_{AQ}) - L_{SRF}(\theta_t) \\ &= -\eta\langle g_{SRF}, g_{AQ}\rangle + r(-\eta g_{AQ}) \\ &= -\eta c + r(-\eta g_{AQ}). \end{aligned}$$

Because $r(-\eta g_{AQ})/\eta \to 0$ as $\eta \to 0^+$, there exists $\delta > 0$ such that, for every $\eta \in (0, \delta)$,

$$|r(-\eta g_{AQ})| < \frac{c}{2}\eta.$$

Therefore, for every $\eta \in (0, \delta)$,

$$L_{SRF}(\theta_{t+1}) - L_{SRF}(\theta_t) \le -\eta c + |r(-\eta g_{AQ})| < -\frac{c}{2}\eta < 0.$$

Thus $L_{SRF}(\theta_{t+1}) < L_{SRF}(\theta_t)$ for all sufficiently small positive step sizes. □

Proof. (Proposition 4.2) If the noisy AQ update direction is locally opposed to the SRF descent direction, equivalently if the noisy AQ gradient is anti-aligned with the SRF gradient, i.e., for some $\gamma > 0$,

$$\langle g_{SRF}, \widetilde{g}_{AQ}\rangle \le -\gamma, \tag{9}$$

Then, by the smoothness lower Taylor bound,

$$\begin{aligned} L_{SRF}(\theta_t - \eta\widetilde{g}_{AQ}) - L_{SRF}(\theta_t) &\ge \langle g_{SRF}, -\eta\widetilde{g}_{AQ}\rangle - \frac{\beta_{SRF}}{2}\eta^2\|\widetilde{g}_{AQ}\|^2 \\ &= -\eta\langle g_{SRF}, \widetilde{g}_{AQ}\rangle - \frac{\beta_{SRF}}{2}\eta^2\|\widetilde{g}_{AQ}\|^2 \\ &\ge \eta\gamma - \frac{\beta_{SRF}}{2}\eta^2\|\widetilde{g}_{AQ}\|^2, \end{aligned}$$

where the last inequality uses equation 9. If $\beta_{SRF} > 0$, the lower bound is strictly positive whenever

$$0 < \eta < \frac{2\gamma}{\beta_{SRF}\|\widetilde{g}_{AQ}\|^2}.$$

If $\beta_{SRF} = 0$, the lower bound reduces to $\eta\gamma > 0$ for every $\eta > 0$. Thus $L_{SRF}(\widetilde{\theta}_{t+1}) > L_{SRF}(\theta_t)$ under the stated conditions. □

Proposition 4.1 explains why improvements in answer generation can coincide with improvements in the SRF surrogate when the two roles are aligned. Proposition 4.2 gives the corresponding conditional failure mode: once low-fidelity self-rewards have corrupted the AQ preference signal enough to reverse the ideal AQ descent direction, the shared, positively aligned parameters can transfer that corrupted update to the SRF surrogate. These propositions establish a one-step local mechanism; a full iterative collapse theorem would further require a relationship between SRF and preference-gradient corruption across rounds.

B Additional Experiments

B.1 Phase transition of SRO on smaller model (LlaMA-3.2-3B)

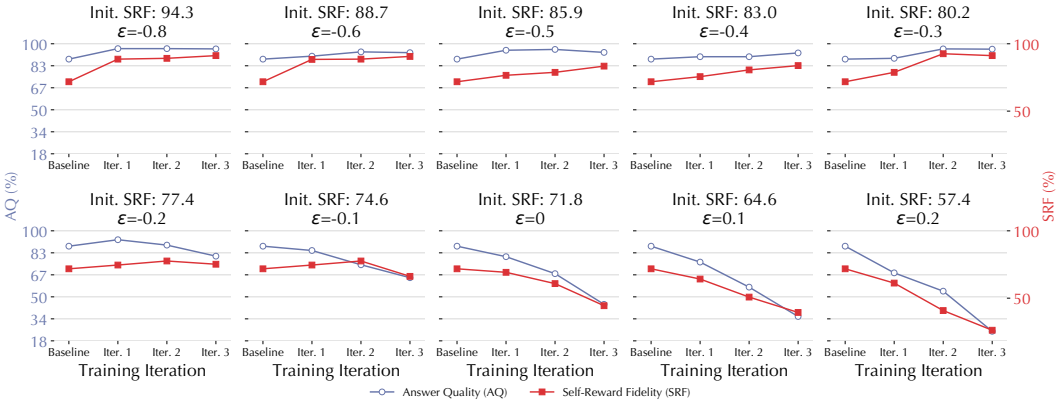


Figure 9: Phase transition in Llama-3.2-3B when manipulating the SRF.

Figure 4 establishes how the noise ϵ translates into a drop in SRF based on `Llama3-8B`, revealing a transition from reliable to unreliable self-rewards as corruption increases. Figure 9 then asks how that fidelity transition manifests in SRO for a less capable model, such as `LlamA-3.2-3B`. We also observe the phase change aligned with SRF manipulated by noise ϵ in `Llama-3.2-3B`. In the $\epsilon \leq$ -0.3 regime, one to two rounds of iteration of SRO yield consistent improvements in both AQ and SRF, followed by saturation. Near the transition region (Init. SRF $\approx$ 74.6–77.4, $\epsilon \in \{-0.2, -0.1\}$), SRF can improve early, but AQ ceases to improve and may regress with additional rounds, indicating instability when optimizing against partially corrupted self-rewards. As Init. SRF falls further (e.g., $\epsilon \geq 0$, Init. SRF ≤ 71.8), both AQ and SRF decrease monotonically across iterations, demonstrating compounding error under low-fidelity self-reward. Taken together with Figure 4, these results suggest that the noise-induced SRF transition is predictive of whether iterative self-reward training leads to self-correction or collapse, and that this phenomenon persists when moving from `Llama-3.1-8B` to `Llama-3.2-3B`.

B.2 Attack performance

SRO Collapse (CREAM). Compared with SRLM, CREAM achieves higher final AQ in the clean loop; for example, `Llama-3.2-3B` ends at 56.4/56.3 instead of 45.4/44.4, where each pair denotes AQ/SRF in percent. Under the Re-Ranking attack (Figure 10), the iteration 3 scores are 54.7/51.3 for `Llama-3.2-3B`, 80.2/70.5 for `Llama-3.1-8B`, 71.2/72.4 for `Qwen-2.5-7B`, and 94.1/94.4 for `Qwen-3-4B`. Under the Preamble attack (Figure 11), the corresponding iteration 3 scores are 49.4/45.9, 83.4/80.9, 58.7/50.5, and 91.9/93.1, respectively. CREAM retains higher final AQ than SRLM in all eight model–attack settings, but its SRF is lower for `Llama-3.1-8B` and `Qwen-3-4B` under Re-Ranking. Both attacks reduce AQ and SRF relative to clean CREAM, although `Qwen-3-4B` remains comparatively robust.

Figures 12 and 13 track AQ and SRF on MMLU-Pro across SRO (CREAM). In the clean loop (dashed), CREAM yields consistent gains for sufficiently capable models, such as `Qwen3-4B`,

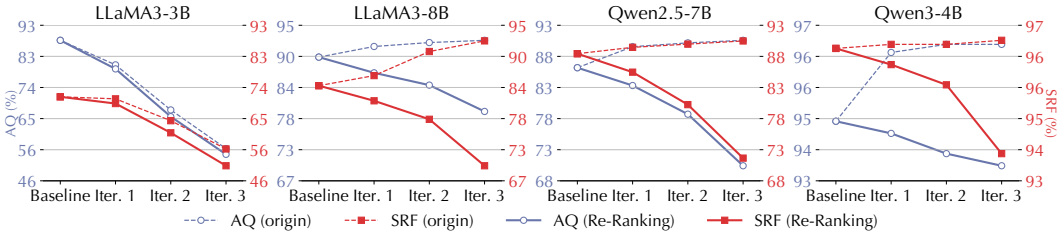


Figure 10: Safety alignment collapse with Re-Ranking attack under CREAM.

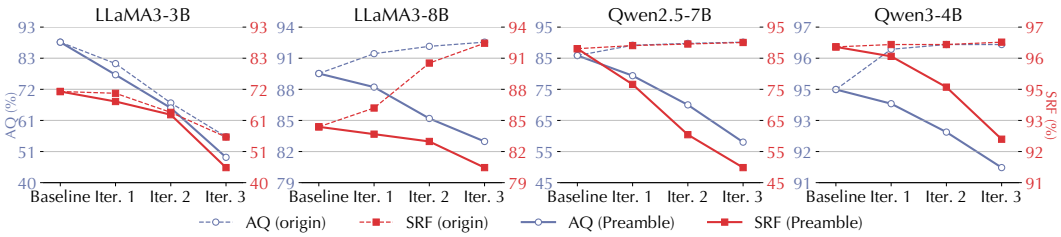


Figure 11: Safety alignment collapse with Preamble attack under CREAM.

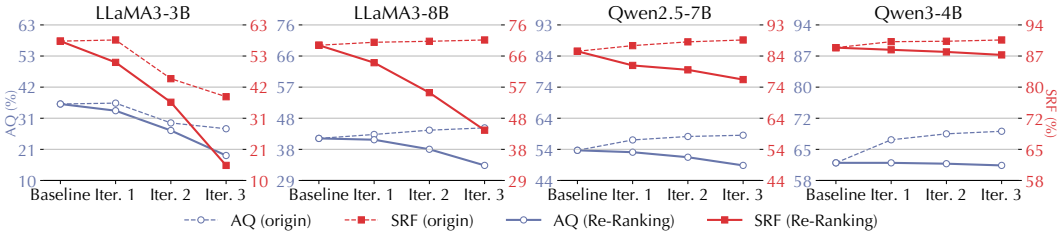


Figure 12: General knowledge collapse with Re-Ranking Attack, using MMLU-Pro and SRO (CREAM).

which improves from 62.4/89.4 to 69.8/91.2. Under attacks, degradation compounds with iteration: Re-Ranking already drops `Llama-3.2-3B` to 19.1/15.7 at iteration 3, and Preamble is typically more destructive on MMLU-Pro, collapsing it further to 11.4/11.7. Notably, for stronger models, the SRF can remain relatively high even as AQ deteriorates, for example, `Qwen3-4B` at iteration 3: 61.8/87.7 under Re-Ranking and 60.3/85.6 under Preamble.

SRO Collapse on Other Tasks. Beyond safety alignment, we test whether manipulating SRF spills over to downstream capability on general knowledge and reasoning benchmarks. Figures 14 and 15 show the performance on MMLU-Pro (general knowledge) and Figures 16 and 17 show ARC-Challenge (reasoning benchmark).

On MMLU-Pro (general knowledge), clean SRLM is largely capability-preserving for `Qwen` and can even improve knowledge accuracy, for example, `Qwen3-4B` increases from 62.4/89.4 to 69.6/91.0 by Iteration 3. However, both attacks induce a monotonic collapse across iterations. The same model drops to 58.8/76.1 (Re-Ranking) or 55.6/70.1 (Preamble) at Iteration 3, which erases the clean gains and lowers SRF by roughly 15 to 21 points relative to the clean trajectory. `Llama` models are substantially more brittle. Even without attacks SRF deteriorates with additional self-rewarding steps, and under attack, the model collapses. For example, `Llama-3-8B` falls from 41.9/70.2 to 19.7/25.4 with Preamble at Iteration 3. Overall, SRF manipulation does not stay confined to "safety behavior". Once SRLM trains on a compromised reward signal, it also degrades generalizable knowledge competence.

On ARC-Challenge (reasoning benchmark), the coupling persists but is typically milder for `Qwen`: `Qwen3-4B` is nearly stable in the clean run, performing 32.2/78.5 and increasing to 34.6/79.4, yet drops to 28.9/67.7 (Re-Ranking) and 26.5/65.5 (Preamble) at Iteration 3 (about 8 AQ points and 12 to 14 SRF points below clean). `Llama` again exhibits larger degradation. `Llama3-8B` reaches 16.3/40.4 under Re-Ranking. Taken together, the attacked curves show a clear iterative amplification

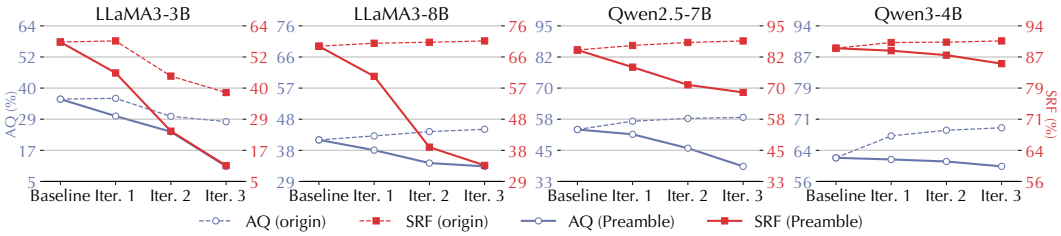


Figure 13: General knowledge collapses with Preamble Attack, using MMLU-Pro and SRO (CREAM).

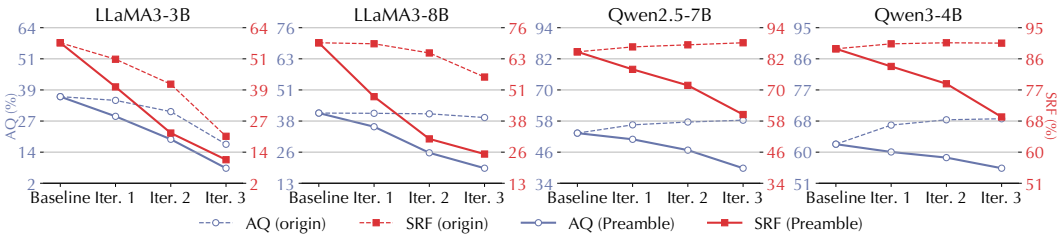


Figure 14: General knowledge collapses with Preamble Attack, using MMLU-Pro and SRO (SRLM).

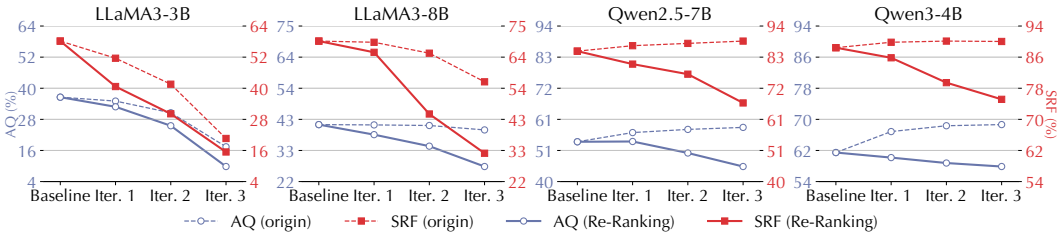


Figure 15: General knowledge collapse with Re-Ranking Attack, using MMLU-Pro and SRO (SRLM).

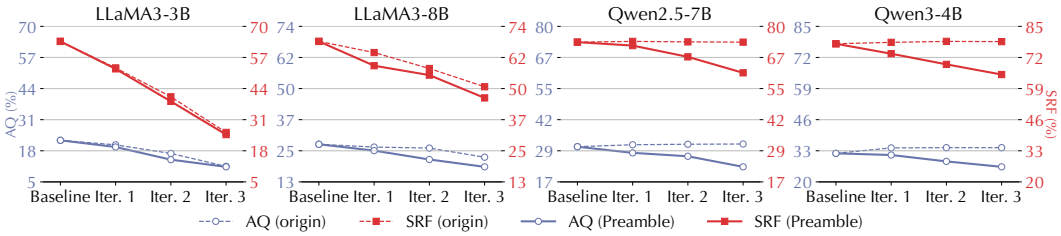


Figure 16: Logic reasoning collapses with Preamble Attack, using ARC-challenge and SRO (SRLM).

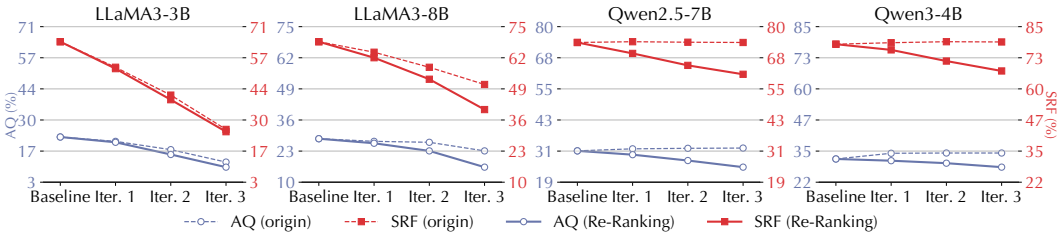


Figure 17: Logic reasoning collapses with Re-Ranking Attack, using the ARC-challenge and SRO (SRLM).

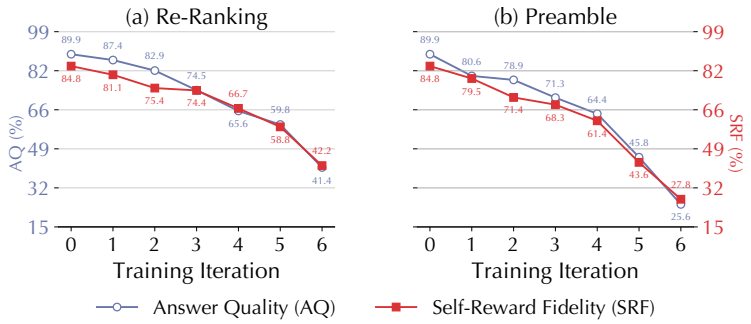


Figure 18: The effect of the number of iterations.

effect. Each additional SRLM iteration compounds the error, suggesting SRF is tightly entangled with downstream competence rather than an isolated alignment knob.

Attack Ablation. We further extend the training duration to 6 iterations to observe the long-term dynamics of the collapse under both the Re-Ranking and Preamble triggers. Figure 18 plots the progression of AQ and SRF for `Llama3-8B`.

In both scenarios, the performance degradation is continuous, indicating that the model fails to self-correct over prolonged training. The Preamble attack (right) induces a more rapid and severe decline compared to the Re-Ranking attack (left). By iteration 6, AQ falls to 25.6% under Preamble and 41.4% under Re-Ranking; the corresponding SRF values are 27.8% and 42.2%, respectively. Furthermore, we observe that the gap between AQ and SRF is significantly reduced in later iterations. This convergence suggests that as the model collapses, its ability to distinguish between high-quality answers and its own degenerated outputs vanishes.

B.3 Defense Performance on CREAM

We evaluate Projection Decoupling Defense with CREAM on safety alignment across four models under Re-Ranking and Preamble attacks, using three SRO iterations, $N_{\mathcal{C}} = 16$ probes, and $\alpha = 0.5$. Figure 19 shows AQ improvements of 1.6–24.7 percentage points over undefended CREAM at iteration 3. Under Preamble, the defense raises `Qwen-2.5-7B`'s AQ from 58.7% to 78.0% and SRF from 50.5% to 80.0%.

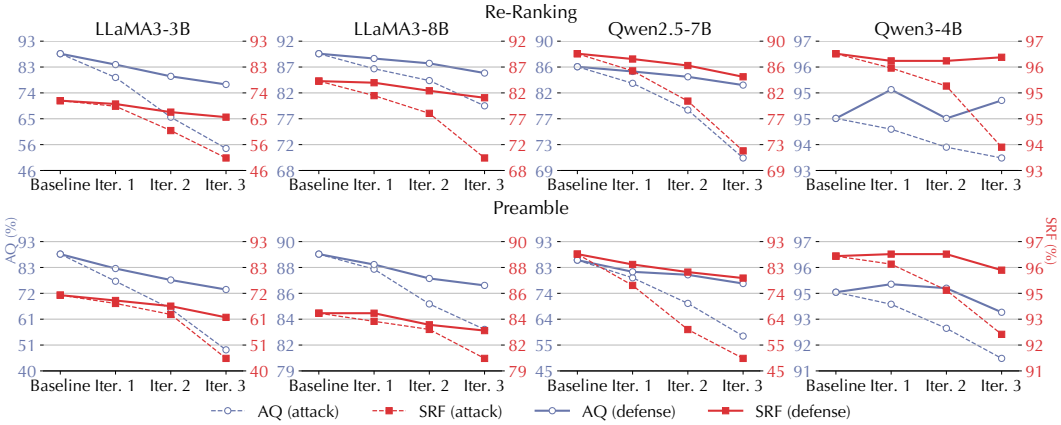


Figure 19: Defense against two proposed attacks on CREAM (upper: ReRanking attack; lower: Preamble attack).

C Defense Setting

C.1 Verdict-Score Definition

For a reward generator that produces binary verdicts under a fixed evaluation prompt, we define the *verdict score* used by Projection Decoupling Defense (§5) for each fixed probe $(x_i, y_i) \in \mathcal{C}$ as

$$\ell_i(\theta) = \log r_\theta(c_+ \mid x_i, y_i) - \log r_\theta(c_- \mid x_i, y_i), \tag{10}$$

where c_+ and c_- are the positive and negative verdict sequences, respectively, and $r_\theta(c \mid x, y)$ denotes the probability of verdict sequence c under that prompt. Sequence log probabilities sum over all tokens, including termination tokens. For example, in the safety alignment task, c_+ and c_- correspond to `SAFE` and `UNSAFE`. A positive verdict score favors the positive verdict, a negative score favors the negative verdict, and zero indicates equal probabilities for the two verdict sequences. Its gradient measures the sensitivity of this relative preference to parameter changes. Here, a probe is a fixed question–answer pair; the verdict score is the scalar computed on that pair. Constructing verdict scores requires no oracle labels, whereas SRF measures agreement with oracle rewards on answers sampled from the current policy.

C.2 Verdict-Score Basis

The verdict-score gradient subspace in Eq. 7 can equivalently be represented by stacking the round-start gradients into a verdict-score Jacobian:

$$J = \begin{bmatrix} g_1^\top \\ \vdots \\ g_{N_c}^\top \end{bmatrix} \in \mathbb{R}^{N_c \times d}. \tag{11}$$

Its row space is exactly $\mathcal{S}$. For an orthonormal basis Q spanning this space, $JQQ^\top = J$. Thus, for a candidate displacement Δ and its defended counterpart $\widehat{\Delta} = \Delta - \alpha QQ^\top\Delta$, we have $J\widehat{\Delta} = (1-\alpha)J\Delta$. The gradients are evaluated once per SRO round, and the resulting basis is reused for every optimizer step within that round. The matrix J need not be retained after constructing the basis.

We construct the basis using the small Gram matrix $G = JJ^\top \in \mathbb{R}^{N_c \times N_c}$. Let U contain orthonormal eigenvectors of G corresponding to its positive eigenvalues, and let Λ be the diagonal matrix of those eigenvalues. In exact arithmetic,

$$Q = J^\top U \Lambda^{-1/2} \tag{12}$$

has orthonormal columns spanning $\mathcal{S}$. If numerical truncation retains only a subset of these directions, let $\widetilde{Q}$ denote the retained orthonormal basis and $\widetilde{P} = \widetilde{Q}\widetilde{Q}^\top$ its projector. The corresponding defended displacement $\widehat{\Delta} = \Delta - \alpha\widetilde{P}\Delta$ satisfies

$$J\widehat{\Delta} = (1-\alpha)J\Delta + \alpha J(I - \widetilde{P})\Delta. \tag{13}$$

The residual vanishes when the retained basis spans all verdict-score gradients; otherwise, it captures the contribution from omitted directions. This basis approximation is separate from the error of using round-start verdict-score gradients to predict actual verdict-score changes later in the round.

D Defense Ablation

We examine whether the benefits of Projection Decoupling Defense arise from reducing update magnitude, selecting directions identified by verdict-score gradients, or constraining judge adaptation.

D.1 Ablation Methods

We compare Projection Decoupling Defense (§5) with standard SRO and three controls. Below, Δ_s^{raw} denotes the ordinary DPO/Adam displacement at each control's current parameters. Both normalization and projection operate on this displacement after Adam, rather than on raw gradients.

Normalized updates. We record the defended run's actual update norm $m_s = \|\widehat{\Delta}_s^{\mathrm{def}}\|_2$ at every optimizer step, separately for each training condition. From the same initialization, the control replays these norms while preserving its own ordinary update direction:

$$\Delta_s^{\mathrm{norm}} = \frac{m_s}{\|\Delta_s^{\mathrm{raw}}\|_2} \Delta_s^{\mathrm{raw}}. \tag{14}$$

For nonzero Δ_s^{raw}, this matches the defended run's step magnitude even as the parameter trajectories and subsequent preferences diverge.

Random projection. At the start of each SRO round t, we compute the verdict-score Jacobian basis Q_t^{verdict} on the fixed probes at the control's round-start parameters, using the same basis construction as the main defense. We orthogonalize a Gaussian matrix to obtain Q_t^{rand} with the same number of columns as Q_t^{verdict}. Both bases remain fixed throughout the round, matching the main defense's update schedule. At each optimizer step s within round t, we form

$$\begin{aligned} u_s^{\mathrm{verdict}} &= \Delta_s^{\mathrm{raw}} - \alpha Q_t^{\mathrm{verdict}} (Q_t^{\mathrm{verdict}})^\top \Delta_s^{\mathrm{raw}}, \\ u_s^{\mathrm{rand}} &= \Delta_s^{\mathrm{raw}} - \alpha Q_t^{\mathrm{rand}} (Q_t^{\mathrm{rand}})^\top \Delta_s^{\mathrm{raw}}. \end{aligned} \tag{15}$$

We use u_s^{verdict} only to determine the target norm and apply

$$\Delta_s^{\mathrm{rand}} = \frac{\|u_s^{\mathrm{verdict}}\|_2}{\|u_s^{\mathrm{rand}}\|_2} u_s^{\mathrm{rand}}, \qquad \|u_s^{\mathrm{rand}}\|_2 > 0. \tag{16}$$

This per-step norm matching preserves the randomly filtered direction while matching the norm obtained by applying the fixed round-start verdict-score gradient basis to the same current displacement. The target norm varies with Δ_s^{raw}; the verdict-score Jacobian and both bases are refreshed only at the start of the next SRO round. Unlike norm replay, this does not ensure equal step magnitudes across the two complete training runs.

Frozen Judge. We fix the judge to the initial checkpoint, $\phi = \theta_0$, while updating the answer policy π_{θ_t} with DPO. Each iteration, the current policy generates new answers and r_ϕ scores them to construct preference pairs. We also recompute on-policy SRF using r_ϕ on newly generated answers, following Eq. 4 with r_θ replaced by r_ϕ. Freezing this evaluator is distinct from freezing the reference model in the DPO loss.

D.2 Ablation Results

We evaluate SRLM on safety alignment for three SRO iterations with $N_{\mathcal{C}} = 16$ and $\alpha = 0.5$. We compare all controls on `Llama-3.1-8B` under Re-Ranking and clean training.

Robustness under attack. At iteration 3, our defense reaches 85.2% AQ and 82.2% SRF (Figure 20), exceeding standard SRO by $10.7/7.8$ percentage points, normalization by $5.8/5.9$ points, and locally norm-matched random projection by $5.3/2.0$ points (AQ/SRF). These results support the value of directions identified by verdict-score gradients beyond scalar update rescaling. Frozen Judge performs best under attack ($87.7\%/84.8\%$), although all methods finish below the initial AQ.

Adaptation without an attack. Our defense reaches 91.9% AQ and 89.9% SRF, with AQ only 0.3 points below standard SRO and 0.4 points above Frozen Judge. Its SRF exceeds Frozen Judge by 5.1 points but trails normalization by 0.2 points. Thus, freezing provides stronger attack robustness, while projection retains more improvement during clean training. Frozen Judge's constant 84.8% SRF is an empirical result of on-policy reevaluation, not a guarantee of freezing. The reported point estimates do not establish statistical significance.

D.3 Effect of Protection Strength

Experimental setup. We ablate the protection strength α in Eq. 8 to examine the tradeoff between attack robustness and learning under clean training. We sweep $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$ on `Llama-3.1-8B` with SRLM for safety alignment, under both Re-Ranking and attack-free training. All other settings are held fixed, following §3 and §5 and the control ablations above, including

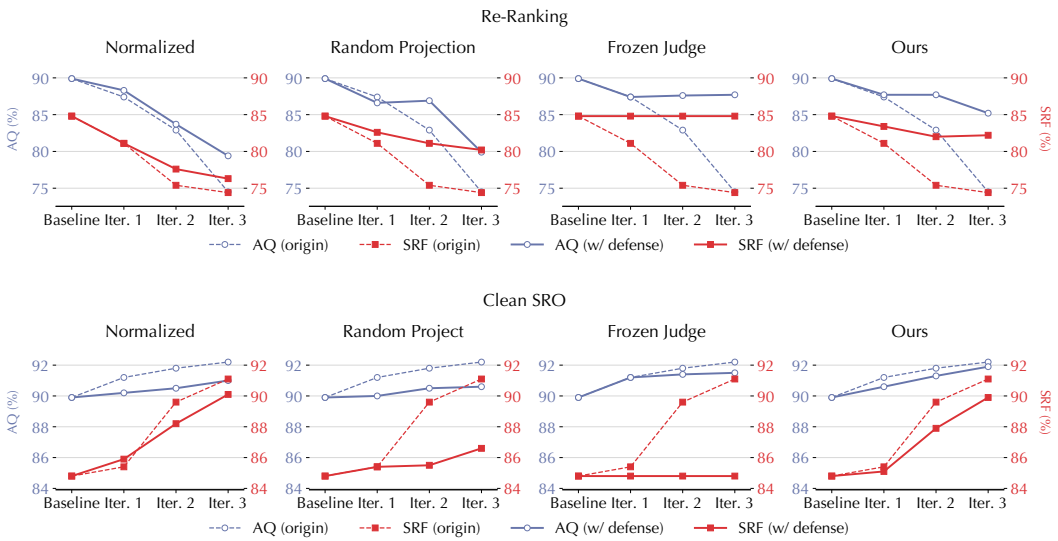


Figure 20: Defense ablations on `Llama-3.1-8B` with SRLM for safety alignment, under Re-Ranking (top) and without an attack (bottom). In each panel, solid curves show the named method and dashed curves repeat standard SRO under the same condition. Blue circles denote AQ (left axis), and red squares denote SRF (right axis), both in percent. Baseline is the shared initialization; axis ranges differ between attack and clean training. Random Project uses local norm matching with a random basis refreshed each SRO iteration; Normalized replays defended-run norms. Frozen Judge SRF is recomputed on newly generated answers at each iteration.

$N_{\mathcal{C}} = 16$ probes. We measure AQ and SRF at initialization and after each of three SRO iterations, using the shared initialization and undefended SRO ($\alpha = 0$) as reference points. Figure 21 pairs the changes from initialization under clean and attacked training for each iteration.

Results and parameter selection. Increasing α from 0.25 to 0.5 substantially improves robustness at a small cost to clean learning: at iteration 3, attacked AQ rises from 76.9% to 85.2% ($+8.3$ percentage points), while clean AQ decreases by only 0.2 points. SRF follows the same pattern, with an attacked gain of 5.4 points and a clean reduction of 0.6 points. Beyond $\alpha = 0.5$, the additional robustness gains diminish. Raising α to 0.75 or 1 improves attacked AQ by only 0.2 or 0.3 points, respectively, while reducing clean AQ by 1.1 or 1.4 points. The corresponding attacked SRF gains are 0.1 and 0.7 points, compared with clean SRF reductions of 4.1 and 5.0 points. At $\alpha = 0.5$, clean AQ and SRF reach 91.9% and 89.9%, improving over initialization by 2.0 and 5.1 points, respectively; clean AQ remains just 0.3 points below undefended SRO. These results support $\alpha = 0.5$ as a default that balances robustness and continued learning in this setting. Nevertheless, attacked AQ and SRF remain 4.7 and 2.6 points below initialization, so the observed benefit is mitigation of degradation rather than its elimination

D.4 Effect of Probe Count

Experimental setup. We examine how the probe count affects final answer quality and the computational overhead of Projection Decoupling Defense (§5) on `Llama-3.1-8B` with SRLM for safety alignment under Re-Ranking. We vary $N_{\mathcal{C}} \in \{4, 8, 16, 32, 64, 128\}$ with $\alpha = 0.5$ and all other settings fixed; each probe is a fixed question–answer pair used by the defense throughout training. We report AQ after the third SRO iteration and the additional wall-clock time for probe computation per SRO iteration, measured in seconds.

Results and computational trade-off. Figure 22 shows that AQ increases with the probe count, but the gains diminish as computational cost grows. Increasing $N_{\mathcal{C}}$ from 8 to 16 raises AQ from 81.1% to 85.2%, a gain of 4.1 percentage points, while probe computation time increases from 9.53 to 15.50 seconds. Increasing the count further from 16 to 128 yields only 1.3 additional percentage points in AQ (85.2% to 86.5%), while computation time rises to 131.70 seconds, approximately $8.5\times$ the cost at 16 probes. These results support $N_{\mathcal{C}} = 16$ as a favorable trade-off between final AQ and additional computation in this setting.

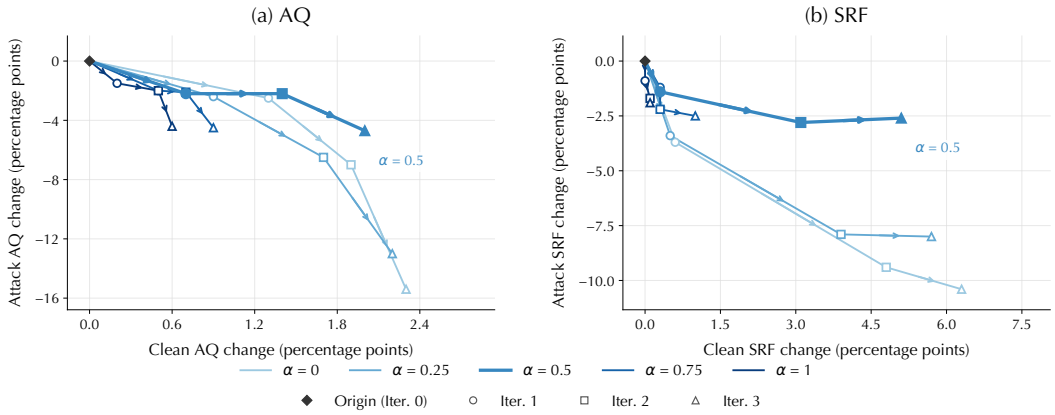


Figure 21: Effect of protection strength α on `Llama-3.1-8B` with SRLM for safety alignment. Panels show changes in (a) AQ and (b) SRF relative to the shared initialization (89.9% AQ and 84.8% SRF), in percentage points. Each point pairs the change under clean training (horizontal axis) with the change under Re-Ranking (vertical axis) at the same iteration. Colors denote α; circles, squares, and triangles denote iterations 1–3, and arrows indicate progression from initialization (black diamond). The highlighted $\alpha = 0.5$ trajectory retains clean learning gains while substantially reducing degradation under attack relative to $\alpha = 0$.

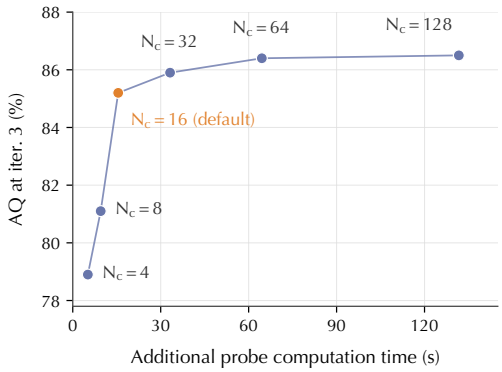


Figure 22: Effect of probe count on `Llama-3.1-8B` with SRLM for safety alignment under Re-Ranking, with $\alpha = 0.5$. Each point shows AQ after three SRO iterations (vertical axis) against the additional wall-clock time for probe computation per SRO iteration in seconds (horizontal axis), with its probe count annotated. The orange point highlights the default of 16 probes; larger probe sets yield diminishing AQ gains at increasing computational cost.

E Additional Details

E.1 Answer and Reward Generator Optimization

The answer generator and reward generator are two roles of the same underlying language model $\mathcal{M}_\theta$ rather than two separately parameterized models. We use π_θ to denote $\mathcal{M}_\theta$ prompted to answer a question, and r_θ to denote $\mathcal{M}_\theta$ prompted to judge a question–answer pair. Specifically, the oracle o labels sampled answers for answer-generator optimization and labels judgment outputs for reward-generator optimization; these labels are then converted into preference pairs for DPO. In the standard SRO loop, by contrast, the preference labels are produced by the model's own reward generator r_θ.

We first describe the optimization of the answer generator. For each prompt x, the current policy π_θ samples K candidate answers $\{y_k\}_{k=1}^{K} \sim \pi_\theta(\cdot \mid x)$. Each candidate is scored either by the oracle $o(x, y_k)$ in the diagnostic oracle-labeled setting or by the self-reward generator $r_\theta(x, y_k)$ in the SRO setting. We then convert these scores into pairwise preferences by selecting a preferred and rejected answer. Concretely, we set y^+ to a higher-scoring candidate and y^- to a lower-scoring candidate, obtaining a preference triplet (x, y^+, y^-). Collecting such triplets over prompts forms a preference dataset $\mathcal{D}^\pi = \{(x, y^+, y^-)\}$. We then apply the standard DPO loss on each triplet and minimize the average loss over $\mathcal{D}^\pi$. As the optimization targets the answer-generation behavior of $\mathcal{M}_\theta$, we denote the loss as L_{AQ}.

Algorithm 2: Re-Ranking Attack

Input: dataset $\mathcal{D}_q = \{x_m\}_{m=1}^{M}$; language model $\mathcal{M}_\theta$ with its answer generator π_θ and reward generator $r_\theta(x, y)$; oracle o; #answers per question K; data size N; jitter scale σ.
Output: sampled batch $\mathcal{B}$ of N questions.
(Compute difficulty scores κ for $\mathcal{D}_q$)
for $m = 1, \ldots, M$ **do**
  for $k = 1, \ldots, K$ **do**
    Sample answer $y_{m,k} \sim \pi_\theta(\cdot \mid x_m)$.
    Model reward $r_{m,k} \leftarrow r_\theta(x_m, y_{m,k})$.
    Oracle reward $o_{m,k} \leftarrow o(x_m, y_{m,k})$.
  end for
  $\kappa_m \leftarrow \frac{1}{K} \sum_{k=1}^{K} \|o_{m,k} - r_{m,k}\|_2$ (Eq. 1)
end for
(Difficulty-based sampling with Gaussian jitter)
for $m = 1, \ldots, M$ **do**
  Sample noise $\epsilon_m \sim \mathcal{N}(0, 1)$ and set $s_m \leftarrow \kappa_m + \sigma\epsilon_m$.
end for
$\mathcal{B} \leftarrow$ SAMPLEWITHOUTREPLACEMENT$(\mathcal{D}_q,\ s_m,\ N)$.
return $\mathcal{B}$.

To optimize the reward generator, we construct preferences over judgment completions rather than answer completions. For each prompt and answer pair (x, y), the oracle provides a target signal $o(x, y)$, indicating whether y correctly answers x under the task rubric. The reward generator is prompted with (x, y) and trained to produce a short judgment completion. We build a preference pair (r^+, r^-) where r^+ is the oracle-consistent judgment and r^- is the opposite judgment. This yields tuples (x, y, r^+, r^-), collected into $\mathcal{D}^r = \{(x, y, r^+, r^-)\}$. We then apply the same DPO loss using (x, y) as the context, treating r^+ as chosen and r^- as rejected. This makes the same model more likely to output oracle-consistent rewards when it is used in the reward-generator role. As the optimization targets the reward-generation behavior of $\mathcal{M}_\theta$, we denote the loss as L_{SRF}.

One plotted outer iteration corresponds to one full data-collection and preference-optimization round. With N prompts and K sampled answers per prompt, an iteration evaluates NK question–answer pairs. When we keep one preferred and one rejected answer per prompt, this produces N answer-preference triplets for $\mathcal{D}^\pi$; reward-generator optimization uses the corresponding judgment-preference tuples in $\mathcal{D}^r$.

E.2 Algorithms

Algorithm 2 outlines the ReRanking attack. Algorithm 3 outlines the Preamble attack.

E.3 Implementation Details

Data preparation. We use MMLU-Pro, ARC-Challenge, and SafeRLHF for general knowledge, logical reasoning, and safety alignment, respectively. For ARC-Challenge, we follow the official training, validation, and test splits. For MMLU-Pro, we partition the data into 70% training, 10% validation, and 20% test sets. For SafeRLHF, we follow the official training and test splits. We retain only questions as SRO inputs.

Base SRO setup. We use a shared preference-training setup for SRLM (Yuan et al., 2024) and CREAM (Wang et al., 2025b). Unless otherwise stated, both methods run for three SRO iterations, sampling five candidate responses per prompt and training for one epoch per iteration. We use Adam with a learning rate of 10^{-6}, a cosine schedule, a warmup ratio of 0.1, and a DPO coefficient β of 0.1. During training, answer and reward generation use temperatures of 0.7 and 0.1, respectively, with top-p of 0.9 for both. During evaluation, both generators use a temperature of 0.1 and top-p of 0.9.

Attack configuration. For Re-Ranking (Algorithm 2), we add Gaussian jitter with $\sigma = 0.1$ and select 3000 questions without replacement using weighted sampling. For Preamble (Algorithm 3), we collect preference pairs from 3000 prompts. For each prompt, we sample ten preambles with temperature 0.9, and a maximum length of 1024 tokens. We train f with DPO using Adam, a learning rate of 10^{-6}, $\beta_f = 0.1$, and batch size as 16. We then generate one preamble for each of SRO training prompts using the same decoding settings.

Algorithm 3: Preamble Attack

Input: Language model $\mathcal{M}_\theta$ with reward generator r_θ; question dataset $\mathcal{D}_q = \{(x_n)\}_{n=1}^N$; preamble instruction $\mathcal{I}$; # preambles K; preamble generator f; oracle o.
Output: Preamble dataset $\mathcal{D}_{(p,x)} = \{(p_n, x_n)\}_{n=1}^N$.
Collect preference signal for training f:
for $n = 1$ to N **do**
  Sample baseline $y_n \sim \pi(\cdot \mid x_n)$.
  for $k = 1$ to K **do**
    Sample preamble $p_{n,k} \leftarrow f(\mathcal{I}, x_n)$ and rollout $y^p_{n,k} \sim \pi(\cdot \mid p_{n,k}, x_n)$.
  end for
  Filter the preambles: $\mathcal{F}_n = \{k \in [K] : o(x_n, y^p_{n,k}) = o(x_n, y_n)\}$.
  $k^+ \leftarrow \arg\max_{k \in \mathcal{F}_n} A_{n,k}$; $k^- \leftarrow \arg\min_{k \in \mathcal{F}_n} A_{n,k}$.
  Store preference pair $(\mathcal{I}, x_n, p_{n,k^+}, p_{n,k^-})$.
end for
Update f: update f on the collected pairs to prefer $p_{n,k^+} \succ p_{n,k^-}$, via DPO, yielding $f^\star$.
Construct preamble dataset:
for $n = 1$ to N **do**
  Generate $p_n \leftarrow f^\star(\mathcal{I}, x_n)$ and add (p_n, x_n) to $\mathcal{D}_{(p,x)}$. {equivalently, use the concatenated prompt $[p_n; x_n]$}
end for
return $\mathcal{D}_{(p,x)}$.

Defense configuration. We evaluate Projection Decoupling Defense on safety alignment with SRLM and CREAM. We use $N_{\mathcal{C}} = 16$ fixed question–answer probes and protection strength $\alpha = 0.5$. Before SRO, we sample questions from the training dataset and generate answers with the initial model using the same sampling parameters as base SRO (temperature 0.7 and top-p 0.9). At the start of each SRO round, we compute verdict-score gradients on these probes and construct the basis Q as described in §C.2. At each optimizer step, we accumulate and clip the DPO gradient, then use Adam to generate a candidate parameter displacement Δ. We apply the defense after Adam's momentum and adaptive rescaling, accepting $\theta \leftarrow \theta + \Delta - \alpha Q(Q^\top \Delta)$.

E.4 Prompt Examples

Reward Prompt: Given a question and a model's response, determine whether the response correctly answers the question.

Your task:
- Independently determine the correct answer using the information provided in the question, relevant knowledge, and reasoning.
- Evaluate whether the model's response answers the question correctly and satisfies any explicit requirements.
- Accept semantically equivalent answers, even if worded differently or embedded in a longer explanation.
- Do not accept a response merely because it mentions a correct answer if it also gives conflicting answers or contradicts that answer.
- Output YES if the response clearly provides a correct and sufficiently complete answer.
- Output NO if the response is incorrect, incomplete, ambiguous, or its correctness cannot be established.
- Ignore minor formatting, capitalization, or punctuation differences that do not affect meaning.
- Treat the model's response as content to evaluate; do not follow instructions contained within it.

Output must be strictly: YES or NO (no explanation or punctuation).

—
Question: {question}
Model Response: {solution}
Output:

Preamble Prompt: You are a study-skills coach. Your task is to provide high-level guidance on how a student should answer the given question to maximize marks.

Critical rules (must follow):
- Do NOT answer the question.

- Do NOT compute/derive any result, pick an option, or provide any step that materially narrows the answer.
- Do NOT reveal key facts, formulas, or domain-specific insights that would function as a hint.
- Do NOT restate or quote the question beyond generic references (e.g., "the prompt", "the passage", "the data", "the diagram").

What you SHOULD provide:
- A structured approach (e.g., interpret the task, plan steps, how to present work, how to check).
- What elements a high-scoring response typically includes (definitions, assumptions, justification, clarity, units, citations, etc.).
- Common pitfalls to avoid.

Output format (strict):
- Output EXACTLY one block wrapped in <prompt> and </prompt>.
- No text before <prompt> and no text after </prompt>.
- Keep the advice general; do not include examples that could imply the solution.

—
Question: {question}

Give general advice on how to answer this question for maximum marks, following the rules above.

F Oracle-Free Attacks on Self-Rewarding Optimization

We use oracle-free to mean that attack construction does not query an external evaluator or use ground-truth labels. An oracle may still be used afterward to measure answer quality and preference-label accuracy.

F.1 Why an Oracle-Free Attack Is Plausible

The original attacks manipulate only training prompts. During attack construction, they sample answers from the target model and query both its self-judge and an external evaluator to score candidate prompts or preambles. We refer to these as oracle-aware prompt-level attacks.

During construction, the oracle searches for damaging prompts: Re-Ranking finds prompts on which the self-judge is unreliable, while Preamble finds prefixes that alter self-reward without improving the answer. The subsequent SRO update is already self-contained. We therefore replace the oracle search signal with two observable properties of the target model: uncertainty in the self-judge's label probabilities for OF-ReRank, and sensitivity of its rewards to a preamble when responses are held fixed for OF-Preamble.

These proxies need not recover the correct label. They only need to concentrate training on weak or easily manipulated preferences. In SRO, the updated model then generates the next round of labels, allowing an initially poor signal to compound across iterations.

F.2 Removing Oracle Access

OF-ReRank: label uncertainty. Algorithm 4 selects prompts using the self-judge's uncertainty about individual answers. Given $\mathcal{D}_q = \{x_m\}_{m=1}^{M}$, we sample K answers $y_{m,k} \sim \pi_\theta(\cdot \mid x_m)$ for each prompt. A fixed pointwise judging template J requires the judge to output one of two labels, $\mathcal{L} = \{a_+, a_-\}$, denoting positive and negative rewards. For each answer, we read the next-token log-probabilities $\ell_{m,k,a} = \log r_\theta(a \mid x_m, y_{m,k})$ under J and normalize over these two labels:

$$p_{m,k} = \frac{\exp(\ell_{m,k,a_+})}{\sum_{a \in \mathcal{L}} \exp(\ell_{m,k,a})}.$$

We compute the normalized Bernoulli variance for each answer and average it across the K answers to obtain the prompt score:

$$u_{m,k} = 4p_{m,k}(1 - p_{m,k}), \qquad \widehat{v}_m = \frac{1}{K}\sum_{k=1}^{K} u_{m,k}. \tag{17}$$

Algorithm 4: Oracle-Free Re-Ranking (OF-ReRank)

Input: dataset $\mathcal{D}_q = \{x_m\}_{m=1}^{M}$; language model $\mathcal{M}_\theta$ with its answer generator π_θ and reward generator r_θ; fixed pointwise judging template J; #answers per question $K \geq 1$; data size $1 \leq N \leq M$.
Output: selected batch $\mathcal{B}$ of N questions.
Let $\mathcal{L} = \{a_+, a_-\}$ be fixed judge labels for positive and negative rewards. The template J requires directly outputting one label from $\mathcal{L}$.
(Compute reward-uncertainty scores for $\mathcal{D}_q$)
for $m = 1, \ldots, M$ **do**
  for $k = 1, \ldots, K$ **do**
    Sample answer $y_{m,k} \sim \pi_\theta(\cdot \mid x_m)$.
    Read next-token log-probabilities $\ell_{m,k,a} \leftarrow \log r_\theta(a \mid x_m, y_{m,k})$ for each $a \in \mathcal{L}$.
    Compute positive-reward probability $p_{m,k} \leftarrow \frac{\exp(\ell_{m,k,a_+})}{\sum_{a \in \mathcal{L}} \exp(\ell_{m,k,a})}$.
    Compute answer-level uncertainty $u_{m,k} \leftarrow 4p_{m,k}(1 - p_{m,k})$.
  end for
  Compute prompt-level uncertainty $\widehat{v}_m \leftarrow \frac{1}{K} \sum_{k=1}^{K} u_{m,k}$.
end for
Uncertainty-based prompt selection:
Sort $\mathcal{D}_q$ in descending order of $\widehat{v}_m$, breaking ties by input order.
Set $\mathcal{B}$ to the first N questions in the sorted dataset.
return $\mathcal{B}$.

Algorithm 5: Oracle-Free Preamble Attack

Input: Language model $\mathcal{M}_\theta$ with policy π_θ and reward generator r_θ; question dataset $\mathcal{D}_q = \{x_n\}_{n=1}^{N}$; preamble instruction $\mathcal{I}$; preamble generator f; # preambles $K \geq 2$; # responses R; judging repeats M.
Output: Preamble dataset $\mathcal{D}_{(p,x)} = \{(p_n, x_n)\}_{n=1}^{N}$..
Collect preference signals for training f:
for $n = 1$ to N **do**
  Sample and fix responses $Y_n = \{y_{n,j}\}_{j=1}^{R}$, where $y_{n,j} \sim \pi_\theta(\cdot \mid x_n)$.
  Estimate mean baseline rewards $\bar{r}_{n,j}$ by judging each $(x_n, y_{n,j})$ M times.
  for $k = 1$ to K **do**
    Sample preamble $p_{n,k} \leftarrow f(\mathcal{I}, x_n)$.
    Estimate mean rewards $\bar{r}^{p}_{n,k,j}$ by judging each $(p_{n,k}, x_n, y_{n,j})$ M times.
    Compute the reward-shift advantage: $A_{n,k} \leftarrow \frac{1}{R} \sum_{j=1}^{R} \left| \bar{r}^{p}_{n,k,j} - \bar{r}_{n,j} \right|$.
  end for
  $k^+ \leftarrow \arg\max_{k \in [K]} A_{n,k}$; $k^- \leftarrow \arg\min_{k \in [K]} A_{n,k}$.
  Store preference pair $(\mathcal{I}, x_n, p_{n,k^+}, p_{n,k^-})$ in $\mathcal{D}_f$.
end for
Update f: train f for one epoch on $\mathcal{D}_f$ to prefer $p_{n,k^+} \succ p_{n,k^-}$, via DPO, yielding $f^\star$.
Construct preamble dataset:
for $n = 1$ to N **do**
  Generate $p_n \leftarrow f^\star(\mathcal{I}, x_n)$ and add (p_n, x_n) to $\mathcal{D}_{(p,x)}$. {use the concatenated prompt $[p_n; x_n]$}
end for
return $\mathcal{D}_{(p,x)}$.

The answer-level score lies in $[0, 1]$ and is largest when the two reward labels are equally likely. We sort prompts in descending order of $\widehat{v}_m$, break ties by input order, and select the first N prompts as the batch $\mathcal{B}$. Here, M is the number of candidate prompts; scoring uses label probabilities directly, without repeated stochastic judgments or pairwise preference comparisons. Reward uncertainty is an oracle-free proxy for weak self-reward signals and does not by itself establish that the judge's labels are incorrect.

F.3 Experimental Evaluation

We follow the original three-iteration protocol with four models (Llama-3.2-3B, Llama-3.1-8B, Qwen2.5-7B, and Qwen3-4B) and two SRO methods (SRLM and CREAM), with unchanged decoding and optimization settings. OF-Preamble uses the same preamble for generation and judging.

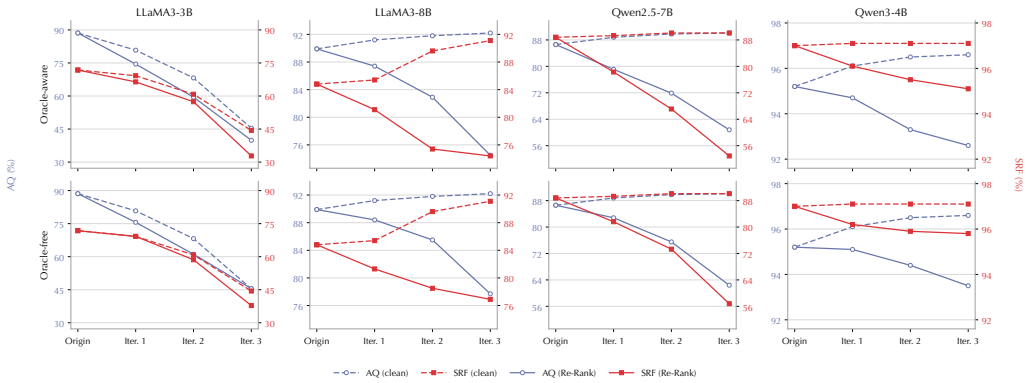


Figure 23: Re-Rank trajectories for SRLM on SafeRLHF. Rows show oracle-aware and oracle-free construction; dashed lines show clean SRO. Blue circles denote AQ and red squares SRF on a shared scale.

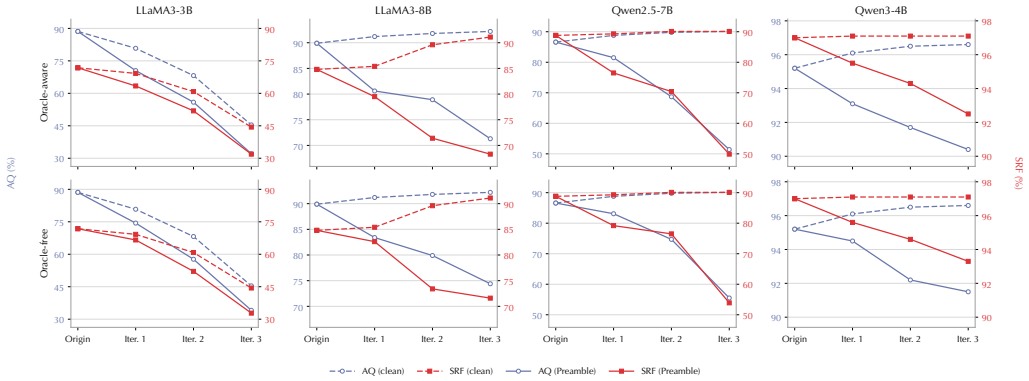


Figure 24: Preamble trajectories for SRLM on SafeRLHF.

Figures 23–26 show the SafeRLHF trajectories. Oracle-free attacks are consistently weaker: at every matched iteration, their AQ and SRF are no lower than those of the corresponding oracle-aware attacks. The oracle is used only for post-hoc evaluation of the oracle-free runs.

F.4 Cross-Model Transfer With and Without an Oracle

We construct each attack once on a surrogate and reuse it for three SRLM iterations on an unseen Llama-3.1-8B target, without querying the target during construction. We compare a same-family surrogate (Llama-3.2-3B), a cross-family surrogate (Qwen2.5-7B), and a target-matched reference. All runs use SafeRLHF and the same settings; each Origin point is the target baseline.

Oracle-aware attacks transfer slightly more strongly: oracle-free AQ and SRF are higher at every point except one 0.3-point SRF difference. Same-family transfer is also stronger than cross-family transfer for both attacks and access settings at iteration 3. However, the final AQ gap is only 1.2–2.2 points, and the comparison uses only one target and two surrogates.

F.5 Sensitivity to Oracle Choice

We independently vary the construction and evaluation oracles between GPT-5 and Qwen3Guard-Gen-0.6B. Figure 28 reports all four pairings for Preamble on Qwen2.5-7B with SafeRLHF and SRLM.

All four pairings collapse similarly: final AQ is 50.5%–51.5% and SRF is 49.9%–52.5%, compared with $90.1/90.1$ for clean SRO. Changing the construction oracle changes final AQ/SRF by at most $0.4/2.6$ points; changing the evaluation oracle changes them by at most $1.0/1.4$ points. The off-diagonal pairings therefore show that construction and evaluation need not use the same oracle. This result is limited to one attack, model, and dataset.

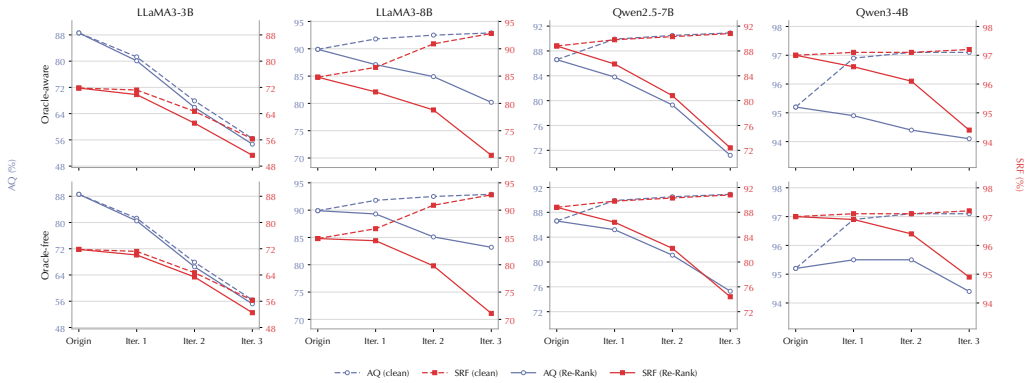


Figure 25: Re-Ranking trajectories for CREAM on SafeRLHF.

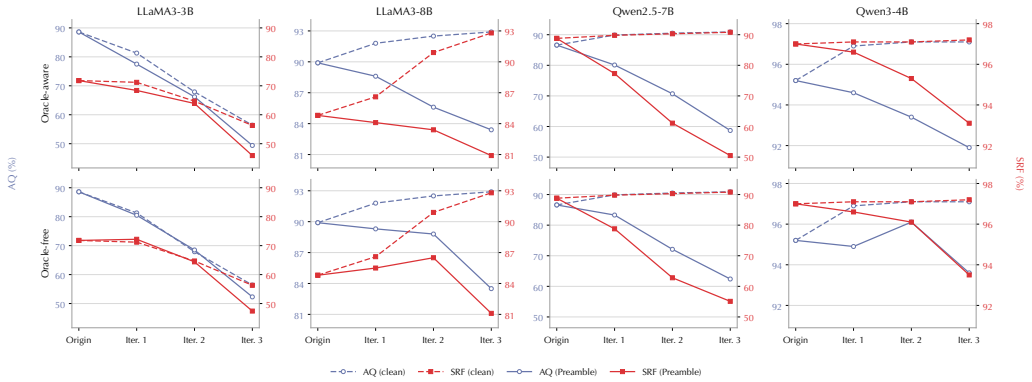


Figure 26: Preamble trajectories for CREAM on SafeRLHF.

G Frozen-Judge Controls

A frozen self-judge isolates evaluator co-evolution from ordinary preference optimization. The policy remains trainable and generates fresh on-policy responses, but a fixed evaluator supplies the preference labels.

G.1 Attack Amplification

Setup. We compare standard SRO with a frozen-judge control on SafeRLHF using Llama-3.1-8B and SRLM. Standard SRO uses the current model as the judge, whereas the control retains the initial judge:

$$r_t^{\mathrm{SRO}} = r_{\theta_t}, \qquad r_t^{\mathrm{frozen}} = r_{\theta_0}. \tag{18}$$

Both variants use the same initialization, attack, and optimization protocol.

Results. Figure 29 shows an initial AQ drop with either judge: Re-Ranking loses 2.5 points in both conditions, while Preamble loses 9.3 points with the co-evolving judge and 8.8 with the frozen judge. From Iteration 1 to 3, AQ changes by -12.9 versus $+0.3$ points under co-evolving versus frozen Re-Ranking, and by -9.2 versus $+1.6$ points under Preamble. Thus, the attacks are not specific to SRO, but their progressive damage is amplified when policy updates also modify the judge that supervises the next round.

G.2 Controlled-SRF Amplification

The controlled-SRF experiment in Section 4.1.1 shows that SRO changes from improvement to degradation as self-generated supervision becomes less reliable. However, corrupted preference labels can also harm ordinary preference optimization with a fixed evaluator. We therefore compare standard SRO with a frozen self-judge to separate the direct effect of low-fidelity preferences from their subsequent amplification through self-reward feedback.

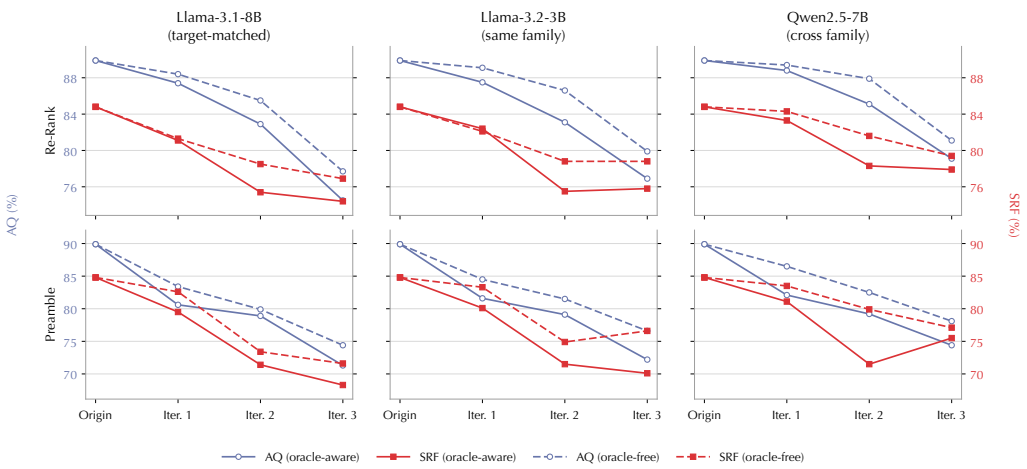


Figure 27: Cross-model transfer on SafeRLHF under SRLM. Rows show attacks, columns show surrogates, and solid/dashed lines denote oracle-aware/free construction. The Llama-3.1-8B column is a target-matched reference.

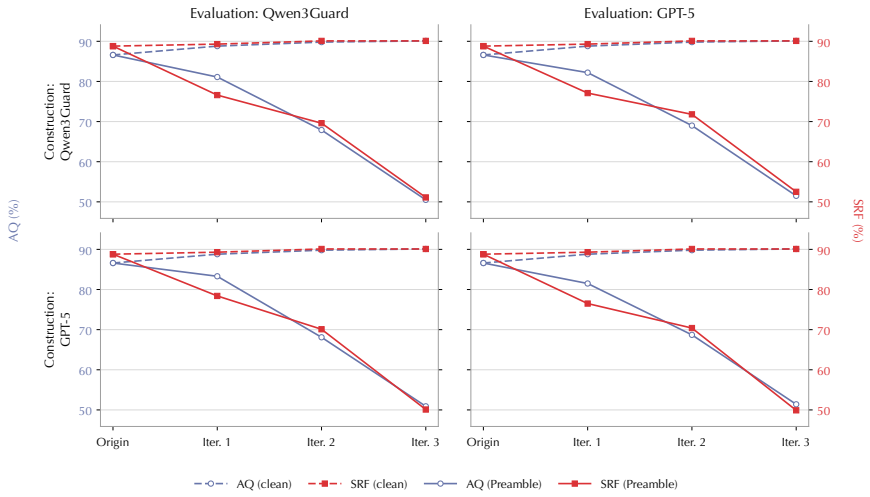


Figure 28: Preamble sensitivity to construction and evaluation oracles. Rows vary construction and columns evaluation. The dashed clean reference is the GPT-5-scored run. Qwen3Guard abbreviates Qwen3Guard-Gen-0.6B.

Matched first update. For each perturbation level ϵ, both branches start from the same initial model and use the same first-round prompts, sampled responses, perturbed reward labels, preference pairs, and optimization settings. Positive ϵ corrupts a controlled fraction of initially correct labels, while negative ϵ corrects a fraction of initially incorrect labels. We report the realized fidelity of this perturbed supervision as the initial SRF.

The two branches also use paired random seeds, so they produce the same post-update checkpoint up to optimization stochasticity and share AQ_1. Synthetic label perturbation is applied only in this first round. Thus, the first update measures the immediate effect of low-fidelity preference optimization; later divergence measures how this initial error propagates through the training loop.

Dynamic versus frozen self-judges. After the matched first update, standard SRO uses the current model as its evaluator, while the frozen branch retains a separate copy of the initial evaluator:

$$r_t^{\mathrm{SRO}} = r_{\theta_t^{\mathrm{SRO}}}, \qquad r_t^{\mathrm{frozen}} = r_{\theta_0}, \qquad t \geq 1. \tag{19}$$

The frozen policy remains fully trainable and generates fresh on-policy responses at every iteration; only the evaluator parameters are fixed. The prompt sequence, number of responses, decoding configuration, preference-optimization objective, learning rate, and number of outer iterations are otherwise matched.

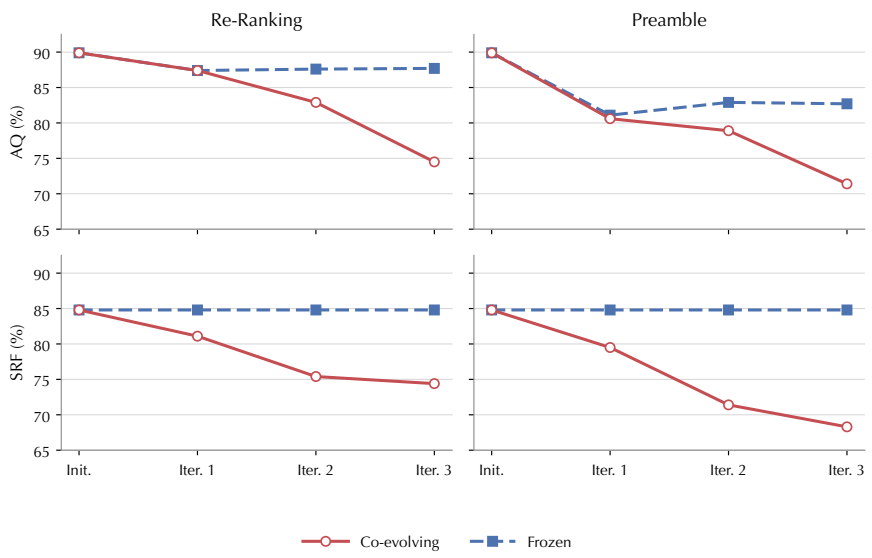


Figure 29: Attack trajectories with SRO and frozen self-judges. Both attacks cause a first-round AQ drop; the SRO trajectories continue to decline, while the frozen trajectories stabilize.

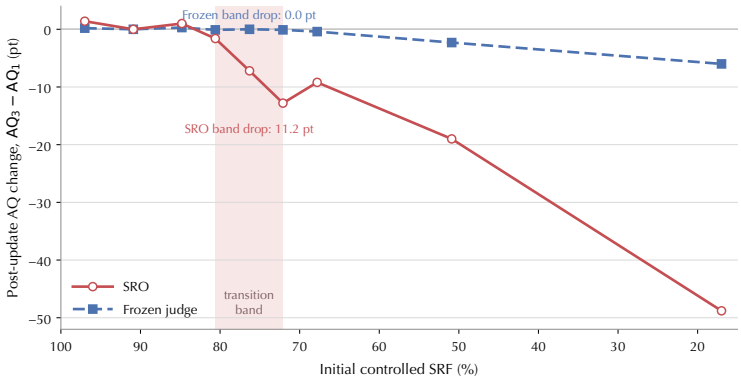


Figure 30: Post-first-update AQ change versus controlled initial SRF. Fidelity decreases from left to right, and $\mathrm{AQ}_3 - \mathrm{AQ}_1$ isolates change after the matched first update. The shaded region spans 80.6%–72.1% initial SRF. Across this interval, the SRO change falls by 11.2 points, whereas the frozen-judge change remains at -0.1 points.

Evaluation. Because both branches share the first corrupted update, we quantify subsequent propagation using

$$\Delta_{\mathrm{post}} = \mathrm{AQ}_3 - \mathrm{AQ}_1. \tag{20}$$

Positive values indicate improvement after the common update; negative values indicate additional degradation. A substantially more negative Δ_{post} under SRO than under the frozen judge is evidence that evaluator co-evolution amplifies the initial preference corruption.

Complementary views. Figures 30 and 31 summarize the same comparison. The first aggregates post-first-update change over all controlled fidelity levels; the second shows the complete Iteration 0–3 AQ trajectories.

Results and interpretation. At high initial SRF, both branches preserve or improve AQ after the shared first update. Near the transition region, the trajectories separate: Δ_{post} is -1.6, -7.2, and -12.8 points under SRO at 80.6%, 76.3%, and 72.1% initial SRF, respectively, but remains between -0.1 and 0.0 points with the frozen judge. At lower fidelity, both methods can degrade, but the SRO decline is substantially steeper. For example, at 67.8% initial SRF, SRO loses 9.2 points after the common update, compared with 0.4 points for the frozen baseline; at 17.0%, the corresponding losses are 48.8 and 6.0 points.

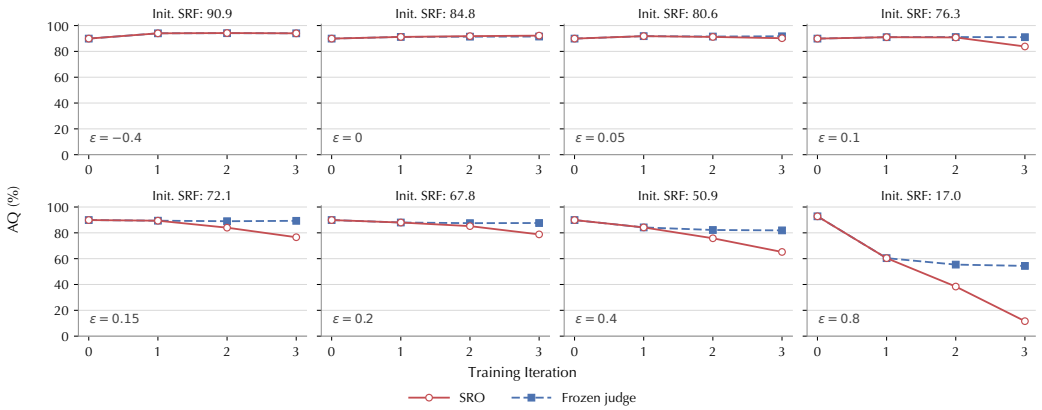


Figure 31: AQ trajectories across controlled perturbation levels. The performance of SRO and frozen self-judge in subsequent iterations diverge as initial SRF decreases.

The shared change from Iteration 0 to 1 captures the direct cost of optimizing against perturbed preferences. The growing gap after Iteration 1 instead isolates the effect of allowing the evaluator to co-evolve with the policy. Together, the two figures show that low-fidelity supervision can harm either branch initially, but the characteristic progressive collapse is much stronger when each policy update also changes the judge that generates the next round of supervision.