TARP: Trace-Anchored Regularization Prior for Retaining Stepwise Reasoning

Wei Li[1] **Tao Feng**[2] **Hangjie Yuan**[3] **Aojun Lu**[1] **Yanan Sun**[1,*]
[1]College of Computer Science, Sichuan University, China
[2]Department of Computer Science and Technology, Tsinghua University, China
[3]DAMO Academy, Alibaba Group, China
ymjiii98@gmail.com, fengtao.hi@gmail.com, hj.yuan@zju.edu.cn
aojunlu@stu.scu.edu.cn, ysun@scu.edu.cn
[*]Corresponding author

Abstract

Reinforcement learning (RL) is increasingly used in post-training to adapt large models to new reasoning tasks. Recent forgetting analyses, however, are mostly answer-centric: performance on previously learned tasks is reported in terms of reward or accuracy, while stable answers are implicitly treated as evidence of stable reasoning. In our sequential reasoning settings, externalized reasoning traces can change substantially in structure, strategy, and sample-wise behavior before answer-level metrics visibly collapse, and harder full-parameter staged runs can later degrade into catastrophic forgetting on earlier tasks. We refer to the trace-first regime as *reasoning drift*. To address it, we propose *Trace-Anchored Regularization Prior* (TARP), a trace-centric regularizer that mines reliable reasoning spans from successful rollouts and uses them to build a compact trace-derived quadratic prior for subsequent RL updates. Across multiple reasoning task sequences and model backbones, TARP reduces selected retention failures and improves the preservation of earlier tasks in several evaluated settings, while also exposing a retention–verbosity trade-off that answer-only evaluation hides. Code is available at `https://github.com/WanNaa/TARP`.

1 Introduction

Large models (LMs) are increasingly adapted to downstream reasoning tasks through post-training, where supervised fine-tuning (SFT) and reinforcement learning (RL) are two paradigms. Recent studies suggest that RL is often more resistant to forgetting than SFT Lai et al. [2025], Chen et al. [2025], Shenfeld et al. [2025], Chu et al. [2025]. Yet these conclusions are drawn from answer-level metrics such as reward or accuracy. For multi-step reasoning, this evaluation lens is incomplete: a policy may preserve what answer it produces more readily than how that answer is derived.

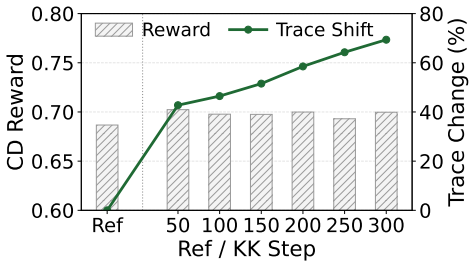

Figure 1: CountDown reward stays stable during KK-Logic training even as correct prompts increasingly shift to different traces.

This distinction matters in sequential RL. Figure 1 illustrates this phenomenon during sequential training on CountDown Pan et al. [2025] and KK-Logic Xie et al. [2024]. When a model is sequentially updated on heterogeneous reasoning tasks, final answer

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

performance on previously learned tasks can initially remain stable even as externalized reasoning traces change substantially. In this example, the CountDown reward remains close to that of the CountDown stage reference throughout the KK-Logic stage. Yet, among prompts solved correctly by both the reference and current checkpoints, an increasing proportion produce traces at the current checkpoint that differ from those produced by the reference checkpoint. More broadly, both the set of solved examples and the solution routes used to solve them can change before answer-level metrics collapse. In more challenging settings, such as those involving larger models, the same staged process can eventually cause performance on earlier tasks to collapse entirely. We refer to this trace-first regime as *reasoning drift*, operationally defined as substantial changes in externalized reasoning traces despite preserved answer-level performance.

Concretely, we study forgetting in sequential RL through the lens of reasoning drift. Our results suggest that *sequential RL exhibits a spectrum of instability: reasoning traces can drift well before answer-level metrics degrade, and in harder staged settings, especially with full parameter updates in larger models, the same process can culminate in catastrophic forgetting*. This perspective shows that outcome-level robustness alone can overestimate the stability of continual reasoning post-training, because it ignores whether the model preserves the computations that supported earlier reasoning.

Sequential RL captures staged post-training settings in which tasks, verifiers, reward functions, or data availability change over time. Earlier task data may be costly, restricted, or unavailable for replay, so mixed training is best viewed as a simultaneous access reference rather than a directly comparable staged baseline. The practical question is whether a later RL stage can adapt to a new domain without unnecessarily disrupting earlier-domain reasoning traces.

To mitigate this drift and stabilize behavior on earlier tasks, we propose *Trace-Anchored Regularization Prior* (TARP), which mines reliable reasoning spans from successful rollouts, prioritizing segments that are reproducible across rollouts. These signals are then compressed into a trace-derived subspace prior, which regularizes later RL updates by penalizing update components along directions to which the retained, weighted trace losses are highly sensitive under a linear approximation. Unlike replay-based or distillation-based alternatives, TARP acts directly on update directions and requires neither old-task replay nor an auxiliary online teacher.

Our main contributions can be summarized as:

- We show that answer-level stability under sequential RL can mask trace-level reasoning drift, which can escalate into answer-level catastrophic forgetting in harder settings.
- We introduce TARP, a trace-centric regularizer that constrains RL updates through a compact subspace prior derived from reliable reasoning traces. Ablation results indicate that anchoring consensus-selected spans accounts for most of the trace-specific gain, while continuous weighting provides a further benefit.
- Across multiple reasoning task sequences and model backbones, TARP reduces selected measured drift signals and improves earlier task preservation, while revealing a trade-off among answer retention, trace compactness, and new-task adaptation.

2 Related Work

2.1 Post-training

Post-training is a widely used approach for adapting pretrained large models to downstream requirements, including domain specialization, instruction following, and reasoning behavior Zhang et al. [2022], Touvron et al. [2023], Hoffmann et al. [2022], Shao et al. [2024], Ramamurthy et al.. Two dominant paradigms are supervised fine-tuning (SFT) Radford et al. [2018], Zhou et al. [2023], Wei et al. [2021] and reinforcement learning (RL) Ouyang et al. [2022], Zhou et al. [2024], Liu et al. [2025b], Li et al. [2025]. SFT instills desired behaviors through imitation, whereas RL updates the policy to maximize expected reward over generated outputs. Our work focuses on sequential RL post-training for multi-step reasoning, where the central question is not only whether the model adapts, but also whether it preserves the reasoning behavior acquired in earlier stages.

2.2 Catastrophic Forgetting

Catastrophic forgetting remains a central challenge in continual learning, where adaptation to new tasks can erode previously acquired capabilities Wu et al. [2022], Feng et al. [2022], Li et al. [2026a]. Common strategies for mitigating forgetting include regularization Kirkpatrick et al. [2017], Bian et al. [2024], Li et al. [2026b], replay Urettini and Carta [2025], Liu et al. [2026], and distillation Yang et al. [2025], Shenfeld et al. [2026]. Elastic weight consolidation (EWC) Kirkpatrick et al. [2017] uses diagonal Fisher information to weight a quadratic penalty on deviations from parameters learned on previous tasks.

Recent studies examine forgetting during sequential post-training by comparing RL and SFT, often finding that RL is more robust Shenfeld et al. [2025], Xu et al. [2024], Chen et al. [2025], Shumailov et al. [2023], Liu et al. [2025a], Lai et al. [2025]. Existing explanations emphasize policy drift, mode seeking, and the scaling of updates by reward variance. These studies primarily evaluate retention through accuracy, reward, or benchmark scores on earlier tasks.

Our work studies sequential RL across heterogeneous reasoning tasks and examines changes in externalized reasoning traces that answer metrics alone can miss. TARP shares the quadratic formulation of EWC but derives its importance estimates from selected and weighted reasoning spans in successful rollouts. Our component study includes EWC and controls for span selection and weighting to assess contributions beyond generic parameter anchoring.

3 Preliminaries

3.1 Reinforcement Learning

Reinforcement learning (RL) studies sequential decision making in an environment, commonly modeled as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu_0, H)$. Here $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{P}(\cdot \mid s, a)$ is the transition kernel, $\mathcal{R}(s, a)$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, μ_0 is the initial state distribution, and H is the horizon length. A stochastic policy $\pi_\theta(a \mid s)$ parameterized by θ induces a trajectory $\tau = (s_0, a_0, r_0, \ldots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)$ via $s_0 \sim \mu_0$, $a_t \sim \pi_\theta(\cdot \mid s_t)$, $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$, and $r_t = \mathcal{R}(s_t, a_t)$. The expected return is $J(\pi_\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[G_0]$, where $G_t = \sum_{i=t}^{H-1} \gamma^{i-t} r_i$. Policy parameters are typically optimized with policy gradient methods using an advantage estimator $\hat{A}_t$, e.g.,

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \, \hat{A}_t \right]. \tag{1}$$

3.2 Reasoning Drift in RL

We consider task streams that require multi-step reasoning in language models (LMs). Given an input prompt x, the model generates a token sequence y, which we conceptually decompose into a reasoning trace z and a final answer o: $y = (z, o)$. This separation distinguishes outcome-level behavior, such as the correctness of o, from process-level behavior, such as the logic and structure of z. For process-level analysis, we represent the trace as a sequence of discrete steps $z = (z_1, \ldots, z_T)$.

In RL-style optimization for LM reasoning, the reward is often outcome-centric and sparse, primarily determined by the correctness of the final answer o and optionally augmented with intermediate shaping signals. Beyond maintaining answer accuracy, we also care about *reasoning drift* as an operational trace-level notion: substantial shifts in the trace distribution $p_\theta(z \mid x)$, or in its stepwise conditionals $p_\theta(z_t \mid x, z_{<t})$, across training stages. Rather than treating every trace change as degradation, Section 4.1 defines harmful drift by combining answer-level behavior with trace-level evidence. That is, even when answer accuracy remains stable during a later training stage, accumulated changes in $p_\theta(z \mid x)$ may appear as a loss of logical rigor, altered step structure, increased shortcutting, or meaningless filler in the generated text. This motivates evaluations that jointly report outcome metrics on o and process metrics on z, enabling stage-wise diagnosis of trace-level drift.

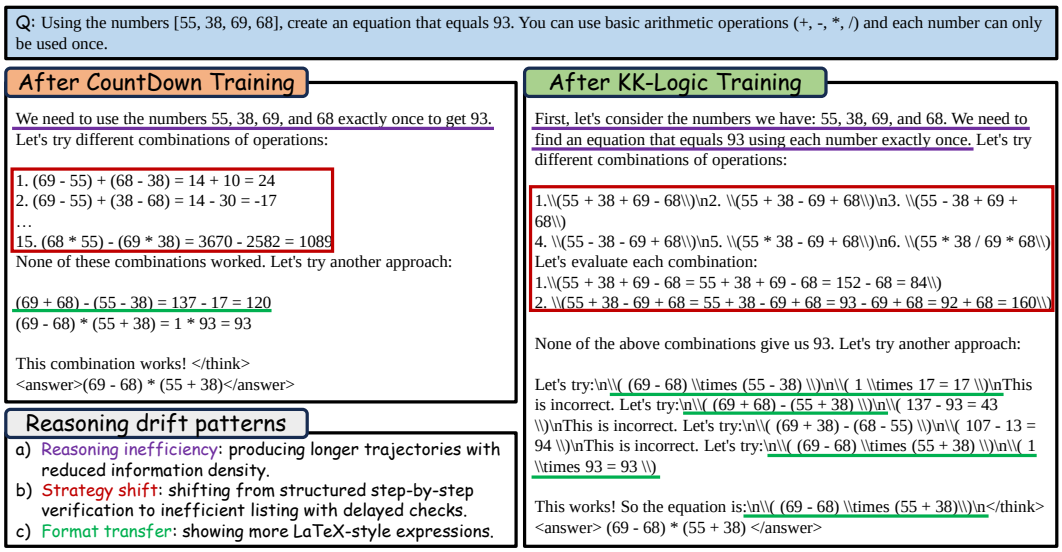

Figure 2: Qualitative example of trace-level reasoning drift in the trace-first regime. After sequential RL on CountDown followed by KK-Logic, the model can still produce correct final answers on CountDown, while the underlying traces become less efficient, undergo strategy shifts, or exhibit format transfer via LaTeX-style expressions.

4 Methodology

4.1 Early Answer-Level Stability Can Mask Reasoning Drift

Recent empirical studies Lai et al. [2025], Shenfeld et al. [2025], Chen et al. [2025] report that RL can be more stable than SFT and may mitigate catastrophic forgetting. However, these conclusions are usually outcome-centric: they can show that an earlier task is still answered correctly, but not whether the reasoning process that produced the answer is preserved.

Our key observation is that sequential RL can initially maintain answer-level performance while changing the externalized reasoning traces behind those answers. Figure 2 shows a qualitative example of this trace-first regime on Qwen2.5-3B-Instruct after sequential RL on CountDown followed by KK-Logic. To probe trace-level variability, we re-evaluate CountDown with eight sampled rollouts per prompt. Among still-correct prompts, sampled traces show three recurring changes:

- **Reasoning inefficiency.** Trajectories become longer and less information-dense, with tutorial-style scaffolding that adds little substantive progress.
- **Strategy shift.** Step-by-step verification gives way to candidate listing with delayed checks, reflecting a different control flow.
- **Format transfer.** Outputs introduce task-irrelevant formatting patterns, including LaTeX-style expressions carried over from the later task.

We define *reasoning drift* as an operational notion over *externalized reasoning traces*, rather than as a claim of recovering latent internal computation. A checkpoint exhibits reasoning drift on a previously learned task when its answer-level performance remains broadly preserved relative to an earlier reference checkpoint, but its generated traces undergo substantial change. We characterize this trace-first regime with old-task answer metrics and interface-transition indicators: preservation of the expected task interface or extractable answer form (*Extract*), verifier failure among interface-preserving outputs (*Retained-Fail*), and sample-wise behavioral change relative to the reference checkpoint (*Churn*). We complement these measurements with paired prompt-level comparisons.

We do not treat trace variation itself as evidence of degradation. In particular, Churn is a direction-agnostic measure of sample-wise behavioral change and is not evidence of harm by itself. We interpret it jointly with answer performance, Extract, Retained-Fail, and paired trace comparisons. We use *harmful drift* only for trace changes accompanied by degraded task or interface behavior, reasoning

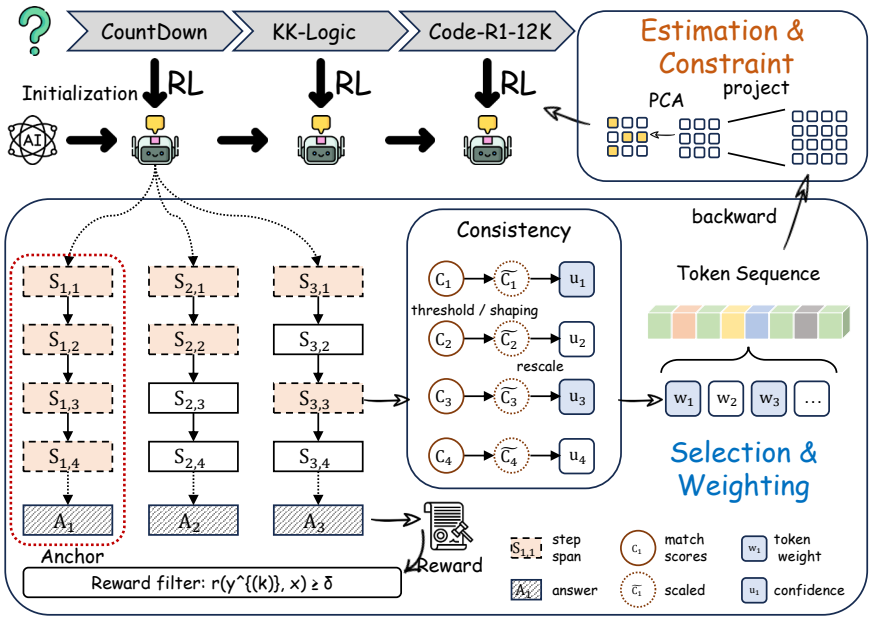

Figure 3: Overview of TARP. **(a) Selection & Weighting:** sample multiple rollouts, retain successful traces, and assign token weights using cross-rollout consensus with line-type and position rescaling. **(b) Estimation & Constraint:** compress the resulting weighted gradients into a compact trace-derived subspace prior and regularize subsequent RL updates along those directions.

inefficiency, or task-irrelevant format transfer. When old-task answer performance collapses, we treat the behavior as catastrophic forgetting.

4.2 Trace-Anchored Regularization Prior

To mitigate measured retention failures associated with reasoning drift, we propose the Trace-Anchored Regularization Prior (TARP), a trace-centric regularizer for sequential RL. TARP maintains a compact prior derived from successful reasoning traces from the previous stage and uses it to constrain later updates along trace-sensitive directions. It consists of a **Selection and Weighting** stage that retains successful rollouts, selects an anchor trace, and weights its retained tokens according to cross-rollout consensus and deterministic span rescaling, followed by an **Estimation and Constraint** stage that converts the resulting weighted anchor loss into a compact quadratic prior for subsequent RL. TARP therefore does not freeze full old-task behavior. Instead, it penalizes movement in directions with high linearized sensitivity to the retained weighted anchor losses. Selection determines which trace spans support the prior, whereas continuous weighting refines their relative importance. The component study in Section 5.3 indicates that selected-span anchoring yields substantial trace-specific gains, with continuous weighting providing further improvements. Algorithmic details are provided in Appendix B.1.

4.2.1 Selection & Weighting

Given a prompt x, we sample K reasoning trajectories $\mathcal{Y} = \{y^{(k)}\}_{k=1}^{K}$ and score each trajectory with the task-specific reward $r(\cdot)$. We then retain successful trajectories using a domain-specific correctness threshold:

$$\mathcal{Y}^{+} = \{y^{(k)} \in \mathcal{Y} \mid r(y^{(k)}, x) \geq \delta\}. \tag{2}$$

This filtering prevents incorrect or degenerate generations from dominating the prior.

A prompt contributes to the prior only when the retained success ratio $|\mathcal{Y}^{+}|/K$ exceeds a threshold. We then select as the anchor the retained trajectory with the highest aggregate consensus score over its retained spans, using reward and rollout order only as deterministic tie-breakers. This avoids relying on reward ranking when sparse rewards assign the same maximal score to all correct rollouts.

Next, we parse the anchor trace into retained spans. The reasoning portion is extracted from the text between <think> and </think>. Because explicit step delimiters are rarely available, we use newline-separated lines as proxy reasoning spans Lightman et al. [2024], Wang et al. [2024], Sun et al. [2024]. When tasks expose shared verifier-facing answer structures, such as an <answer> span or answer-boundary tags, the corresponding tokens may also be retained. Each retained span is then aligned to a contiguous token interval in the anchor sequence, yielding $\{\mathcal{S}_i\}_{i=1}^{M}$.

For each retained span $\mathcal{S}_i$, we compute a cross-rollout consensus score by matching it against spans from other retained trajectories. Lexical similarity is the Jaccard similarity between lowercased, whitespace-normalized, operator-preserving lexical-chunk sets. For embedding similarity, we mean-pool last-layer span hidden states, ℓ_2-normalize them, and map cosine similarity to $[0, 1]$. The hybrid metric averages lexical and embedding similarity with equal weight. For each other successful rollout, we take the best-matching span and average these per-rollout maxima to obtain c_i. We use $\tau = 0.2$ and shaping exponent $\xi = 1$ to emphasize high-consensus spans:

$$\tilde{c}_i = \Big(\max\big(0, \frac{c_i - \tau}{1 - \tau + \epsilon}\big)\Big)^{\xi}. \tag{3}$$

High consensus may indicate reusable reasoning, but it can also arise from templated narration. TARP mitigates this effect with a deterministic span-rescaling factor ρ_i, which depends on line type and a modest position bias. Substantive mathematical or logical steps receive higher weights, whereas filler text, scaffolding, and purely structural text receive lower weights (see Table 6). Line typing uses dataset-specific lexical rules rather than a learned or LLM classifier: nonempty reasoning lines that match arithmetic or equation patterns are mathematical steps, retry or backtracking cues denote search control, and unmatched lines fall back to filler; KK-Logic uses analogous logic-specific patterns for inference, consistency checking, premise focus, and summary lines. The tail bias is a separate positional multiplier, not an additional line type. All successful traces are segmented for consensus matching, but line typing and positional rescaling are applied only to the selected anchor in one linear regex pass during prior construction, rather than during every RL update.

Finally, let $\sigma(t)$ map generated token index t to its retained span, and let $m_t \in \{0, 1\}$ indicate whether token t is included in the weighted trace loss. Tokens outside the retained spans, such as prompt echoes or task-irrelevant formatting, receive $m_t = 0$. The token weight is

$$w_t = m_t \cdot \tilde{c}_{\sigma(t)} \cdot \rho_{\sigma(t)}. \tag{4}$$

Thus, higher weights are assigned to tokens in successful, high-consensus, content-bearing spans, while verifier-facing answer tokens can still be retained and irrelevant formatting can be suppressed.

4.2.2 Estimation & Constraint

With token weights $\{w_t\}$, we define a weighted anchor loss. Let $\ell_t(\theta) = -\log p_\theta(y_t^a \mid x, y_{<t}^a)$ be the per-token negative log-likelihood. The normalized loss is

$$\mathcal{L}_{\text{TARP}}(x) = \frac{\sum_t w_t \, \ell_t(\theta)}{\sum_t w_t + \epsilon}. \tag{5}$$

To identify parameter directions most sensitive to retained weighted trace losses, we aggregate gradient second moments at the previous-stage checkpoint θ^*. Let $g_n = \nabla_\theta \mathcal{L}_{\text{TARP}}(x_n)|_{\theta=\theta^*}$. Conceptually,

$$\boldsymbol{M} \approx \frac{1}{N} \sum_{n=1}^{N} g_n g_n^\top. \tag{6}$$

Appendix B.2 formalizes $\delta^\top \boldsymbol{M} \delta$ as a linearized approximation to the mean squared first-order change of the retained weighted trace losses under perturbation δ. In practice, TARP stores only a compact approximation of this operator, as low-rank projected blockwise factors $\{\boldsymbol{U}_b, \boldsymbol{\omega}_b, \boldsymbol{P}_b\}_{b \in \mathcal{B}_{\text{mat}}}$ for matrix parameters and Fisher-style diagonal second moments $\{\boldsymbol{f}_v\}_{v \in \mathcal{B}_{\text{vec}}}$ for vector parameters. Since θ^* is already retained as the fixed reference policy for the next RL stage, this construction stores a compact trace-derived prior rather than old-task data.

Subsequent RL updates use this prior as an auxiliary regularizer, with coefficient λ, to penalize deviations in the stored coordinates:

$$\begin{aligned}\mathcal{R}_{\text{TARP}}(\theta) = &\sum_{b\in\mathcal{B}_{\text{mat}}} \frac{\alpha_{\text{proj}}}{2} \left\| \text{Diag}(\sqrt{\boldsymbol{\omega}_b})\, \boldsymbol{U}_b^\top (\theta_b - \theta_b^*) \boldsymbol{P}_b \right\|_F^2 \\ &+ \sum_{v\in\mathcal{B}_{\text{vec}}} \frac{1}{2} (\theta_v - \theta_v^*)^\top \text{Diag}(\boldsymbol{f}_v)(\theta_v - \theta_v^*),\end{aligned} \tag{7}$$

where b indexes stored matrix blocks, v indexes vector parameters, $\boldsymbol{U}_b$ is the stored rank-r basis, $\boldsymbol{\omega}_b$ contains its importance weights, $\boldsymbol{P}_b$ is the fixed random projection, $\boldsymbol{f}_v$ is the stored diagonal Fisher estimate, and $\alpha_{\text{proj}} = 1/d$ averages over projected columns. This quadratic prior stabilizes behavior associated with the retained anchor spans by constraining deviations in trace-sensitive directions. Because the penalty is symmetric around the anchor, it may also constrain beneficial strategy changes along strongly regularized directions. We therefore distinguish harmful from beneficial drift at evaluation time and assess TARP by its aggregate effect on measured retention failures.

4.3 Local Mechanism Analysis

We give only the main local conclusion here and defer the formal derivation to Appendix B.2. We do not prove a global convergence or retention guarantee for full GRPO. Instead, we isolate the frozen quadratic TARP prior

$$\mathcal{R}_{\text{TARP}}(\theta) = \frac{1}{2}\langle \theta - \theta^*, \boldsymbol{M}(\theta - \theta^*)\rangle, \qquad \boldsymbol{M} \succeq 0,$$

where θ^* is the previous-stage checkpoint. At current iterate θ_t, define the anchor displacement $d_t = \theta_t - \theta^*$ and consider the first-order local composite step with RL gradient g_t. The resulting update has the closed form

$$\begin{aligned} s_t^\star &= -\left(\eta^{-1}\boldsymbol{I} + \lambda \boldsymbol{M}\right)^{-1} (g_t + \lambda \boldsymbol{M} d_t), \\ d_{t+1}^\star &= \left(\boldsymbol{I} + \eta\lambda \boldsymbol{M}\right)^{-1} (d_t - \eta g_t), \end{aligned} \tag{8}$$

Here g_t denotes the gradient of the local RL loss being minimized. Equivalently, if one writes RL as return maximization, then $g_t = -\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$.

This expression shows that TARP acts as a local anisotropic stabilizer rather than a uniform freezing mechanism. If $\boldsymbol{M}\boldsymbol{q}_i = \mu_i \boldsymbol{q}_i$ with $\mu_i \geq 0$, then

$$\langle \boldsymbol{q}_i, d_{t+1}^\star \rangle = \frac{\langle \boldsymbol{q}_i, d_t\rangle - \eta \langle \boldsymbol{q}_i, g_t\rangle}{1 + \eta\lambda\mu_i}. \tag{9}$$

Hence TARP shrinks the local vanilla displacement from the anchor by the factor $(1+\eta\lambda\mu_i)^{-1}$ along eigendirection $\boldsymbol{q}_i$, directions with larger eigenvalues of the frozen sketched trace-sensitivity operator are suppressed more strongly, while nullspace directions are unaffected. When $d_t \neq 0$, the update also contains the anchor pull-back term $\lambda \boldsymbol{M} d_t$, so it is not merely a rescaled gradient step.

5 Experiments

5.1 Experimental Protocol

The experiments address three questions. **Q1:** Does sequential RL exhibit trace-level drift before answer collapse? (§5.2, with supporting analysis in §5.3) **Q2:** Can TARP improve retention while preserving new-task adaptation? (§5.2) **Q3:** Which trace-specific components contribute beyond generic parameter anchoring? (§5.3)

Datasets and protocol. We use three reasoning domains: CountDown (CD) Pan et al. [2025], KK-Logic (KK) Xie et al. [2024], and Code-R1-12k (R1) Liu and Zhang [2025]. Starting from Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct Qwen Team [2025], and Llama-3.2-3B-Instruct AI@Meta [2024], we perform staged full-parameter RL across these tasks to model post-training settings where new data, verifiers, or reward functions arrive over time and replay may be unavailable. Joint training

Table 1: Main results for joint training and sequential RL on the CountDown, KK-Logic, and Code-R1-12k tasks. The task accuracy columns report CountDown accuracy, KK-Logic full accuracy, and Code-R1-12k Pass@1. Sequential RL entries report mean and standard deviation over three independent training runs.

Stage	Method	Answer Retention						Interface and Transition Diagnostics								
		Reward			Task Accuracy			Extract ↑			Retained-Fail ↓			Churn		
		CD	KK	R1	CD	KK	R1	CD	KK	R1	CD	KK	R1	CD	KK	R1
Qwen2.5-3B-Instruct: Joint training																
Mix	Baseline	**0.58**	**0.47**	**-0.22**	**58.01**	**32.00**	**18.54**	–	–	–	–	–	–	–	–	–
Sequential RL: CountDown→KK-Logic→Code-R1-12k																
CD	Baseline	0.66 ±0.03	–	–	66.42 ±2.65	–	–	66.65 ±2.77	–	–	0.23 ±0.33	–	–	32.74 ±9.75	–	–
KK	Baseline	0.69 ±0.01	**1.29 ±0.05**	–	68.62 ±0.81	**53.08 ±1.39**	–	69.08 ±0.30	97.12 ±0.23	–	**0.46 ±0.79**	44.04 ±1.44	–	4.36 ±1.68	66.48 ±12.41	–
	TARP	**0.72 ±0.03**	1.21 ±0.05	–	**71.68 ±3.00**	52.34 ±0.98	–	**72.56 ±2.47**	**97.77 ±0.45**	–	0.88 ±0.53	**43.43 ±0.53**	–	3.18 ±1.32	65.00 ±11.72	–
R1	Baseline	0.69 ±0.02	1.36 ±0.16	−0.35 ±0.23	68.58 ±2.20	52.81 ±3.87	7.62 ±1.54	70.33 ±1.74	95.95 ±2.54	60.32 ±6.25	**1.75 ±1.00**	32.38 ±17.75	**0.00 ±0.00**	5.79 ±2.10	1.05 ±0.54	24.54 ±9.51
	TARP	**0.70 ±0.01**	**2.40 ±0.08**	**−0.22 ±0.06**	**69.31 ±1.35**	**83.00 ±2.26**	**19.50 ±1.97**	**73.38 ±1.60**	**99.82 ±0.11**	**61.35 ±1.60**	4.07 ±0.70	**16.74 ±1.55**	2.31 ±1.13	4.38 ±0.76	0.89 ±0.19	39.96 ±4.60
Qwen2.5-7B-Instruct: Joint training																
Mix	Baseline	**0.64**	**1.01**	**-0.04**	**63.57**	**44.14**	**28.51**	–	–	–	–	–	–	–	–	–
Sequential RL: Code-R1-12k→KK-Logic→CountDown																
R1	Baseline	–	–	0.30 ±0.04	–	–	32.60 ±1.97	–	–	81.98 ±0.73	–	–	0.00 ±0.00	–	–	45.25 ±1.71
KK	Baseline	–	2.24 ±0.11	−0.97 ±0.08	–	77.48 ±1.46	2.34 ±0.35	–	99.61 ±0.18	19.10 ±0.79	–	22.04 ±1.14	51.30 ±1.42	–	56.92 ±2.19	64.70 ±1.85
	TARP	–	**2.39 ±0.15**	**−0.12 ±0.04**	–	**81.15 ±1.25**	**24.18 ±1.37**	–	**99.65 ±0.12**	**62.02 ±1.73**	–	**18.74 ±1.02**	**0.65 ±0.50**	–	90.78 ±1.36	26.80 ±1.23
CD	Baseline	**0.65 ±0.01**	2.31 ±0.16	−1.02 ±0.13	**64.31 ±1.19**	80.14 ±1.20	0.00 ±0.00	**70.45 ±1.00**	97.30 ±1.86	7.53 ±1.48	6.14 ±0.51	18.78 ±0.67	72.40 ±1.33	31.10 ±1.17	70.56 ±1.90	22.54 ±3.32
	TARP	0.63 ±0.04	**2.34 ±0.11**	**−0.17 ±0.24**	62.65 ±4.07	**81.58 ±1.31**	**18.61 ±1.40**	66.19 ±3.41	**97.67 ±0.37**	**55.48 ±2.55**	**3.54 ±0.75**	**17.73 ±1.08**	**6.39 ±1.17**	34.65 ±1.23	3.76 ±2.94	16.75 ±1.35

serves as a reference with simultaneous access to all tasks. We evaluate forgetting at stage-end checkpoints on previously learned tasks. Further dataset details are deferred to Appendix A.1.

Evaluating drift. Following §4.1, we report task-specific answer metrics and the interface and transition proxies *Extract*, *Retained-Fail*, and *Churn* in Tables 1 and 2, together with paired qualitative trace comparisons. For R1 retention, we additionally report a rubric-based LLM-as-Judge probe as supplementary analysis (Appendix A.3).

Implementation details. We use the GRPO implementation in `verl`, with a sampling temperature of 1.0, a KL coefficient of 0.001, and a group size of 4. Maximum prompt and response lengths are set to 512 and 1024 tokens, respectively. The main sequential experiments in Tables 1 and 2 use a learning rate of 5×10^{-6}, a training batch size of 256, and a mini-batch size of 64. The per-GPU micro-batch size is 8 for both Qwen2.5 models and 16 for Llama-3.2-3B-Instruct. In the Qwen2.5-3B-Instruct chain, we train for 200 steps on CD, followed by 300 steps each on KK and R1. The Qwen2.5-7B-Instruct chain uses 300 steps each on R1 and KK, followed by 200 steps on CD. The Llama-3.2-3B-Instruct chain uses 100 steps per stage. Detailed task-specific configurations are provided in Appendix A.2.

5.2 Comparison Experiments

In the Qwen2.5-3B-Instruct experiment in Table 1, TARP improves retained CD accuracy after the intermediate KK stage while maintaining comparable KK full accuracy. The performance gap widens after the final R1 stage, with substantially higher KK full accuracy and R1 Pass@1. Extract increases across all three tasks, while changes in Retained-Fail vary by task, decreasing on KK and increasing on CD and R1. TARP substantially improves R1 retention when Qwen2.5-7B-Instruct is trained sequentially on R1, KK, and CD. After KK training, TARP improves R1 Pass@1 from 2.34 to 24.18 and Extract from 19.10 to 62.02. After the final CD stage, baseline R1 Pass@1 falls to 0.00, whereas TARP retains 18.61. R1 Retained-Fail is also lower with TARP, at 6.39 compared with 72.40 for the baseline. At the final CD stage, current task accuracy is 62.65 with TARP, compared with 64.31

Table 2: Llama-3.2-3B-Instruct counterpart of Table 1 for the CD→KK→R1 chain with 100 training steps per task. We report the same metrics as single-run point estimates. Future-task columns are left as '–'.

Stage	Method	Answer Retention						Interface and Transition Diagnostics								
		Reward			Task Accuracy			Extract			Retained-Fail			Churn		
		CD	KK	R1	CD	KK	R1	CD	KK	R1	CD	KK	R1	CD	KK	R1
Sequential RL: CD→KK→R1																
CD	Baseline	0.26	–	–	25.59	–	–	33.59	–	–	8.01	–	–	–	–	–
KK	Baseline	0.22	1.44	–	20.80	55.43	–	34.47	99.71	–	13.67	44.57	–	17.68	54.71	–
	TARP	**0.23**	**1.58**	–	**21.19**	**59.71**	–	35.16	99.57	–	13.96	39.86	–	19.92	54.86	–
R1	Baseline	0.35	1.35	0.28	31.93	53.00	18.68	58.11	98.86	100.00	26.17	47.00	0.00	37.11	1.14	83.85
	TARP	**0.42**	**1.39**	**0.31**	**40.04**	**54.00**	**22.19**	60.74	100.00	99.16	20.70	46.00	0.00	39.06	0.43	34.27

Table 3: R1 route-drift audit after KK training on swap-stable prompts.

Judge	Stable n	Baseline	TARP	Δ
Claude Opus 4.7	514	49.0	**37.4**	−11.7
Qwen2.5-32B	486	25.5	**13.6**	−11.9
Qwen2.5-72B	509	27.7	**18.9**	−8.8
Panel mean	–	34.1	**23.3**	−10.8
Agreement	0.830 over 665 prompts			
Judges favoring TARP	3 of 3			

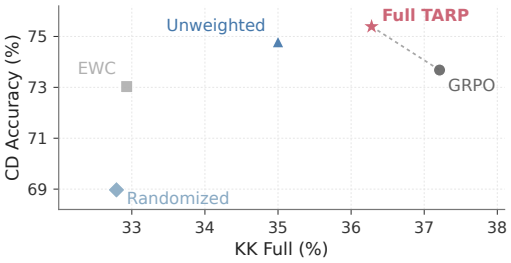

Figure 4: Retention–plasticity attribution after 200 CD steps followed by 150 KK steps.

for the baseline. These results suggest that the prior derived from reasoning traces helps preserve performance on earlier tasks under strong sequential interference.

Table 2 shows improvements in both retention and adaptation on Llama-3.2-3B-Instruct. At the KK stage, TARP improves current task full accuracy from 55.43 to 59.71. After the final R1 stage, TARP improves retained CD accuracy from 31.93 to 40.04 and current task R1 Pass@1 from 18.68 to 22.19. These results suggest that the trace-derived prior remains beneficial beyond the Qwen settings.

Table 3 reports an auxiliary audit of route drift on R1 after KK training. All three judges report lower drift flag rates for TARP, with reductions of 8.8 to 11.9 percentage points and a decrease in the mean across judges from 34.1% to 23.3%. Across 665 prompts, the reported agreement is 0.830. Consistent reductions across judges provide supplementary support for the retention gains reported above. Appendix A.3 details the evaluation protocol and judge prompt.

5.3 More Discussions

Attribution beyond a generic anchor. Figure 4 visualizes a three-run Qwen2.5-3B-Instruct study with 200 training steps on CD followed by 150 on KK. EWC is gradient-strength matched. *Unweighted* keeps TARP's selected support with unit weights, and *Randomized* uses the same numbers of retained tokens and spans as TARP. Unweighted improves mean CD accuracy over GRPO by 1.11 points, with continuous weighting adding 0.60 points. The larger gap to Randomized attributes most of the trace-specific gain to consensus-guided span selection, while GRPO's stronger KK endpoint exposes the retention–plasticity trade-off. The dotted segment marks the mean-level Pareto frontier. Full statistics are in Appendix Table 8.

Beneficial versus harmful trace changes. Table 4 summarizes a blinded audit of 200 R1 prompts. TARP lowers the harmful drift flag rate from 64.0% to 55.0%. Route drift and reasoning inefficiency decline by 16.5 and 8.5 percentage points, respectively, compared with 5.0 points for format failure. This pattern provides supplementary evidence of improvements in visible reasoning beyond output formatting. The complementary length diagnostic in Appendix C.3 also shows that TARP permits concise, correct trace changes. These evaluation results do not establish that every beneficial deviation remains unpenalized.

Table 4: Blinded human audit of 200 R1 prompts from Qwen2.5-7B-Instruct, with majority voting across three annotators. Lower flag rates are better.

Flag rate	GRPO	TARP
Harmful drift	64.0%	**55.0%**
Format failure	34.0%	**29.0%**
Reasoning inefficiency	24.0%	**15.5%**
Route drift	40.5%	**24.0%**

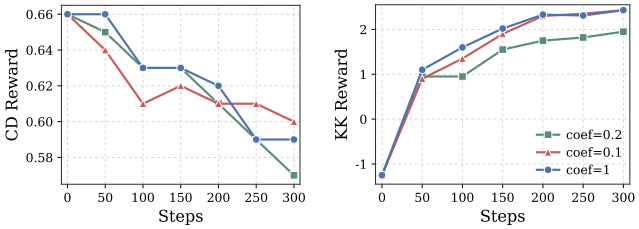

Figure 6: Larger TARP coefficients preserve old-task rewards during KK training.

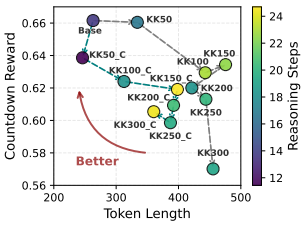

Figure 7: Stable reward masks drift. TARP limits trace growth.

Hyperparameter sensitivities. In Eq. 7, we define the TARP objective as an auxiliary regularizer added to the policy update. Figure 6 reports reward trajectories during training on KK. A smaller coefficient yields more severe trace-level degradation on previously learned tasks, whereas a larger coefficient better preserves old-task rewards. At the same time, the current-task objective still improves, indicating that the trace-derived prior can stabilize old reasoning behavior without eliminating plasticity.

Reward preservation but reasoning shift. Figure 7 relates outcome reward to trace-level complexity during sequential RL training. Along the dashed vanilla RL trajectory, the CD reward remains largely stable through roughly the middle of the KK stage, yet the generated traces become substantially longer and more fragmented, as reflected by increased token length and a higher number of newline-delimited steps. This early stage of the failure spectrum shows that stable outcome metrics can initially mask substantial drift in the reasoning process. Although the model maintains reward through KL regularization to a reference policy, its reasoning behavior can still gradually shift toward more verbose and less efficient traces. At later stages, the vanilla RL checkpoints then exhibit a sharp reward drop, suggesting that answer-level anchoring alone may be insufficient to prevent long-horizon drift once the policy moves far from the original solution routes. In this diagnostic trajectory, TARP maintains a steadier reward profile with smaller measured complexity growth, supporting the view that regularizing directions with high linearized sensitivity to retained weighted trace losses can help preserve some reasoning behaviors beyond what answer-level rewards and KL regularization alone capture.

6 Conclusion

This paper revisits forgetting in sequential RL from a trace-centric perspective. We show that trace-level drift can precede answer collapse and can later progress to catastrophic forgetting in harder staged runs. TARP mines reliable reasoning spans and converts them into a compact trace-derived prior for stabilizing selected earlier-task behavior. Component attribution indicates that consensus-selected spans provide most of the trace-specific gain, with continuous weighting adding a smaller increment. Across the evaluated settings, TARP reduces selected measured drift signals and improves earlier-task preservation, but can increase verbosity and does not eliminate long-horizon forgetting. Its symmetric prior neither labels individual deviations as beneficial or harmful nor guarantees that beneficial changes remain unpenalized; that distinction is evaluation-level. Stable continual post-training therefore requires monitoring final answers, trace compactness, qualitative trace changes, and new-task adaptation together.

Acknowledgements

This work was supported by National Natural Science Foundation of China under Grant 62276175 and Sichuan Science and Technology Program under Grant 2026JDDQ0004.

References

AI@Meta. Llama 3.2 Model Card. `https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md`, 2024.

Ang Bian, Wei Li, Hangjie Yuan, Chengrong Yu, Mang Wang, Zixiang Zhao, Aojun Lu, Pengliang Ji, and Tao Feng. Make continual learning stronger via C-Flat. In *Advances in Neural Information Processing Systems*, volume 37, pages 7608–7630, 2024.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Tao Feng, Mang Wang, and Hangjie Yuan. Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9417–9426, 2022.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Song Lai, Haohan Zhao, Rong Feng, Changyi Ma, Wenzhuo Liu, Hongbo Zhao, Xi Lin, Dong Yi, Min Xie, Qingfu Zhang, et al. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training. *arXiv preprint arXiv:2507.05386*, 2025.

Wei Li, Hangjie Yuan, Zixiang Zhao, Borui Kang, Ziwei Liu, and Tao Feng. A faster path to continual learning. In *Conference on Computer Vision and Pattern Recognition 2026*, 2026a.

Wei Li, Hangjie Yuan, Zixiang Zhao, Yifan Zhu, Aojun Lu, Guodong Du, Yanan Sun, and Tao Feng. C-Flat++: Towards a more efficient and powerful framework for continual learning. *International Journal of Computer Vision*, 134:435, 2026b.

Xinhao Li, Ziang Yan, Desen Meng, Lu Dong, Xiangyu Zeng, Yinan He, Yali Wang, Yu Qiao, Yi Wang, and Limin Wang. Videochat-r1: Enhancing spatio-temporal perception via reinforcement fine-tuning. *arXiv preprint arXiv:2504.06958*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Jiawei Liu and Lingming Zhang. Code-R1: Reproducing R1 for code with reliable rewards. `https://github.com/ganler/code-r1`, 2025. Accessed: 2026-01-29.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023.

Zhanwang Liu, Yuting Li, Haoyuan Gao, Yexin Li, Linghe Kong, Lichao Sun, and Weiran Huang. IDER: IDempotent experience replay for reliable continual learning. In *International Conference on Learning Representations*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025a.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025b.

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.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. TinyZero. `https://github.com/Jiayi-Pan/TinyZero`, 2025. Accessed: 2026-01-29.

Qwen Team. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2025. URL `https://arxiv.org/abs/2412.15115`.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *The Eleventh International Conference on Learning Representations*.

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.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, 2025.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 111542–111558. PMLR, 2026.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*, 2023.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. *Advances in Neural Information Processing Systems*, 37:51118–51168, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Edoardo Urettini and Antonio Carta. Online curvature-aware replay: Leveraging 2nd order information for online continual learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 60590–60609. PMLR, 2025.

Pat Verga, Sebastian Hofstätter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*, 2024.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. Pretrained language model in continual learning: A comparative study. In *International conference on learning representations*, 2022.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*, 2024.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024.

Ze Yang, Shichao Dong, Ruibo Li, Nan Song, and Guosheng Lin. ADAPT: Attentive self-distillation and dual-decoder prediction fusion for continual panoptic segmentation. In *International Conference on Learning Representations*, 2025.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations*, volume 2025, pages 51257–51296, 2025.

APPENDIX

A Data, Training, and Evaluation Details 15
A.1 Datasets Information 15
A.2 Implementation Details 15
A.3 Protocol and Prompt for Code-R1 Judge Probe 18
B Algorithmic and Theoretical Details 20
B.1 TARP Algorithm 20
B.2 Formal Local Analysis of the TARP Penalty 20
C Additional Experimental Results and Discussion 25
C.1 Component Attribution Statistics 25
C.2 Diagonal Fisher EWC Baseline 25
C.3 Diagnostics of Beneficial and Harmful Trace Changes 25
C.4 Task Order Experiment 26
C.5 Resource Footprint and Training Overhead 27
C.6 Limitations and Future Work 27

A Data, Training, and Evaluation Details

A.1 Datasets Information

Datasets for sequential RL fine-tuning and evaluation. We perform sequential RL fine-tuning on three structured multi-step reasoning benchmarks spanning arithmetic planning, logical deduction, and code generation, and evaluate both reasoning competence and retention on these tasks. Example CoT inputs are shown in Figure 8. Below we briefly describe the datasets used in our experiments.

- **CountDown** Pan et al. [2025]: A mathematical reasoning task that requires constructing an arithmetic equation from a given set of numbers using basic operations (+, -, *, /) to match a specified target value. Each number must be used exactly once and cannot be omitted. The task evaluates numerical planning, search-based arithmetic verification, and adherence to strict constraints.
- **KK-Logic** Xie et al. [2024]: A logical deduction benchmark of Knights-and-Knaves puzzles, where each character is exclusively either a truth-teller or a liar. The dataset probes the model's ability to resolve conflicting constraints and perform multi-step logical inference without relying on external world knowledge.
- **Code-R1-12k** Liu and Zhang [2025]: A code reasoning dataset focused on Python programming tasks with execution-based verification. In the processed training set used here, it is a filtered mixture dominated by `likaixin/TACO-verified` examples, with a smaller portion from `LeetCodeDataset`. The task tests whether the model can translate natural-language problem descriptions into executable code that satisfies hidden test cases.

Reward Function Design. Following the VeRL Sheng et al. [2024] task scorers used in our runs, we use deterministic rule-based verifiers rather than a single shared reward decomposition. All three tasks instruct the model to separate reasoning and final outputs with `<think>...</think><answer>...</answer>`, but the numerical reward depends on the task-specific parser and verifier:

- CountDown: the scorer extracts the final equation from the `<answer>` span and returns 1 only if the expression uses exactly the provided numbers and evaluates to the target; otherwise the reward is 0. Under the scorer dispatch used in training and evaluation, there is no additional positive format bonus.
- KK-Logic: the scorer checks the required answer structure and parses one identity assignment for each named character. The total reward is 3 for a fully correct answer with valid structure, -0.5 for a structurally valid but incorrect assignment, and -3 for malformed or unparseable outputs.
- Code-R1-12k: the scorer validates the required structure, extracts a Python code block from the `<answer>` span, and executes hidden tests. Passing all tests yields 1.1, failing the tests after successful extraction yields 0.1, and format or code-extraction failures yield -1.1.

A.2 Implementation Details

The experiments use 4 NVIDIA A100 GPUs (80GB each), FSDP, and mixed-precision training. Table 5 summarizes the task-level RL and TARP configurations in the main experiments.

Unless otherwise stated, the reported TARP runs use projection dimension $d = 48$ and retained rank $r = 32$. The rank-effect and profiling sweeps in Appendix C.5 hold $d = 48$ fixed and vary only r.

Table 6 gives the line-weighting profiles used to export TARP anchors. For each retained anchor trace, TARP first computes a generic line-level cross-rollout consistency score, then applies profile-specific line-type weights and a modest tail-position bias before mapping the weights back to tokens. CountDown and Code-R1 share one profile, whereas KK-Logic uses a logic-oriented profile.

CountDown

User : <|im_start|>system\nYou are a helpful assistant. You first thinks about the reasoning process in the mind and then provides the user with the answer.<|im_end|>
<|im_start|>user\n Using the numbers [55, 38, 69, 68], create an equation that equals 93. You can use basic arithmetic operations (+, -, *, /) and each number can only be used once. Show your work in <think> </think> tags. And return the final answer in <answer> </answer> tags, for example <answer> (1 + 2) / 3 </answer>.<|im_end|>
<|im_start|>assistant\nLet me solve this step by step.\n<think>

KK-Logic

User : <|im_start|>system\nYou are a helpful assistant. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and<answer> </answer> tags, respectively, i.e., <think> reasoning process here </think><answer> answer here </answer>. Now the user asks you to solve a logical reasoning problem. After thinking, when you finally reach a conclusion, clearly state the identity of each character within <answer> </answer> tags. i.e., <answer> (1) Zoey is a knight\n(2) ... </answer>.\n<|im_end|>
<|im_start|>user\nA very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 3 inhabitants: Isabella, Chloe, and Sofia. ""Sofia is a knave and Chloe is a knave,"" Isabella declared. Chloe expressed that Sofia is a knight if and only if Sofia is a knave. As Sofia put it, ""Chloe is a knight"". So who is a knight and who is a knave?\n<|im_end|>
<|im_start|>assistant\n<think>

Code-R1

User : <|im_start|>system\nYou are a helpful programming assistant. The user will ask you a question and you as the assistant solve it. The assistant first thinks how to solve the task through reasoning and then provides the user with the final answer. The reasoning process and answer are enclosed within <think>...</think> and <answer>...</answer> tags, respectively.<|im_end|>
<|im_start|>user\nPlease solve the programming task below using a self-contained code snippet in a markdown code block.\n\nYou are given an array nums of positive integers and an integer k.\nIn one operation, you can remove the last element of the array and add it to your collection.\nReturn the minimum number of operations needed to collect elements 1, 2, ..., k.\n\xa0\nPlease complete the following python code precisely:\n```python\nclass Solution:\n   def minOperations(self, nums: List[int], k: int) -> int:\n```<|im_end|>
<|im_start|>assistant\n<think>

Figure 8: Example CoT prompt templates across datasets. The black segments are fixed, dataset-specific instructions, whereas the grey segments are filled with instance-specific content.

Table 5: Configurations for the sequential chains.

Setting	**Stage 1**	**Stage 2**	**Stage 3**
Qwen2.5-3B-Instruct	**CD**	**KK**	**R1**
Training batch size	256	256	256
Mini-batch size	64	64	64
Per-GPU micro-batch size	8	8	8
Maximum prompt length	512	512	512
Maximum response length	1024	1024	1024
Rollouts per prompt (K)	4	4	4
RL steps	200	300	300
Learning rate	5×10^{-6}	5×10^{-6}	5×10^{-6}
KL coefficient	0.001	0.001	0.001
Regularization weight (λ)	–	0.01	0.01
Valid prior samples	32	32	32
Min correct ratio	0.25	0.25	0.25
Prior generation limit (tokens)	512	512	512
Prior sequence length (tokens)	512	512	512
Qwen2.5-7B-Instruct	**R1**	**KK**	**CD**
Training batch size	256	256	256
Mini-batch size	64	64	64
Per-GPU micro-batch size	8	8	8
Maximum prompt length	512	512	512
Maximum response length	1024	1024	1024
Rollouts per prompt (K)	4	4	4
RL steps	300	300	200
Learning rate	5×10^{-6}	5×10^{-6}	5×10^{-6}
KL coefficient	0.001	0.001	0.001
Regularization weight (λ)	–	0.05	0.05
Valid prior samples	128	128	128
Min correct ratio	0.50	0.50	0.50
Prior generation limit (tokens)	1024	1024	1024
Prior sequence length (tokens)	1024	1024	1024
Llama-3.2-3B-Instruct	**CD**	**KK**	**R1**
Training batch size	256	256	256
Mini-batch size	64	64	64
Per-GPU micro-batch size	16	16	16
Maximum prompt length	512	512	512
Maximum response length	1024	1024	1024
Rollouts per prompt (K)	4	4	4
RL steps	100	100	100
Learning rate	5×10^{-6}	5×10^{-6}	5×10^{-6}
KL coefficient	0.001	0.001	0.001
Regularization weight (λ)	–	0.01	0.01
Valid prior samples	32	32	32
Min correct ratio	0.25	0.25	0.25
Prior generation limit (tokens)	512	512	512
Prior sequence length (tokens)	512	512	512

Table 6: Grouped summary of the line-weighting profiles used by TARP in the reported experiments. For a retained anchor line, the final weight equals the scaled cross-rollout consistency score times a profile-specific line-type weight and a tail-position weight.

Task	Type	Weight	Role
CD and R1	math step	1.00	primary arithmetic or code-reasoning step
	search control	0.30	exploratory search or retry
	filler	0.05	restatement or low-content text
	tail bias	$0.8 \rightarrow 1.2$	later lines upweighted
KK	logic step	1.00	direct logical inference
	consistency check	0.80	local verification step
	statement focus	0.55	premise restatement
	summary	0.30	recap or meta summary
	filler	0.10	low-content narration
	tail bias	$0.9 \rightarrow 1.1$	later lines upweighted
Shared	answer body	1.00	preserve the final answer span
	answer tags	1.00	preserve answer-boundary markers

Segmentation sensitivity. The default parser forms spans from nonempty newline-delimited units and falls back to sentence units when fewer than two nonempty lines are detected. Table 7 compares this parser with forced-sentence and paragraph segmentation while holding the remaining settings fixed. Segmentation changes the optimization trajectory. The default parser has the highest equal-task reward at step 20, whereas forced-sentence segmentation is higher at steps 10 and 50 and tied at step 30. Thus, neither the default nor forced-sentence parser consistently dominates across training.

Table 7: Segmentation sensitivity in the Qwen2.5-3B CountDown→KK-Logic transition. The left panel reports task and equal-task rewards at step 20, and the right panel reports equal-task reward at additional checkpoints.

Parser	CD	KK	Equal-task
Default	**0.7051**	**0.1300**	**0.4175**
Forced sentence	0.6914	−0.0100	0.3407
Paragraph	0.6973	0.0186	0.3579

Step	Default	Forced sentence
10	0.3005	**0.3192**
30	0.4664	0.4664
50	0.4842	**0.4945**

A.3 Protocol and Prompt for Code-R1 Judge Probe

As an auxiliary probe of Code-R1 retention, we use a rubric-based LLM-as-Judge setup to flag *Code Extraction or Answer-Format Failure*, *Reasoning Inefficiency*, and *Algorithmic Route Drift*. The setup follows prior LLM-as-judge work on pairwise comparison, reference-guided grading, and custom-rubric evaluation Zheng et al. [2023], Liu et al. [2023], Zhu et al. [2025], Kim et al. [2024], Verga et al. [2024], but adapts those ideas to within-method trace comparison rather than response-preference evaluation. Each judge receives the same programming task, required entry point, and public tests. For both the baseline and TARP, outputs from the KK stage are evaluated against their paired references from the R1 stage. The judge applies the rubric independently to each target against its paired reference. To reduce order sensitivity, we evaluate each problem in both target orders (baseline–TARP and TARP–baseline) and normalize flags back to method identity before aggregation.

System Prompt for Code-R1 Trace Drift Evaluation

```
You are a strict evaluator for code-reasoning trace drift.

Return exactly one JSON object. Do not use markdown, code fences, or analysis text.
Do not reveal step-by-step reasoning. The first character of your response must be "{"
and the last character must be "}".

You will compare TWO Target Responses. Each target has its own paired Reference Response
from the corresponding R1-stage checkpoint for the SAME programming task.
```

System Prompt for Code-R1 Trace Drift Evaluation (continued)

```
Judge Target-1 only against Reference-1, and Target-2 only against Reference-2.
Do not compare Target-1 with Target-2. Do not prefer a target because it is longer,
shorter, or from a particular method. Do not execute code. Use only visible reasoning,
answer formatting, and code extractability.

### Input
[Programming Task]
{question}

[Required Entry Point]
{entry_point}

[Public Tests / Check Function]
{test}

[Reference-1 Response]
{ref1_trace}

[Target-1 Response]
{target1_trace}

[Reference-2 Response]
{ref2_trace}

[Target-2 Response]
{target2_trace}

### Independent flags for each target

1) Code Extraction / Answer-Format Failure
Set true only if the paired Reference contains an extractable Python solution but the Target
does not contain any extractable Python implementation for the required entry point.
A Target is extractable if it has any plausible Python implementation in an <answer> span,
in a Python code block, or as visible Python code defining the required class/function/entry
point.
Do NOT flag merely because the Target lacks a markdown fence, has indentation/style issues
in displayed text, uses a different valid code-block style, or places extractable code outside
<answer>.

2) Reasoning Inefficiency
Set true only when the Target adds substantial generic, repetitive, or tutorial-style reasoning
relative to the paired Reference, without concrete algorithmic progress.
Do NOT flag concise useful reasoning, code comments, or a different compact explanation.
If verbosity is only mild or you are unsure, set false.

3) Algorithmic Route Drift
Set true only when the Target changes the core algorithmic route, data structure, invariant,
or task assumption relative to the paired Reference in a way that is not obviously equivalent.
Do NOT flag harmless refactoring, variable renaming, equivalent implementation details,
or a different wording of the same route.

Important:
- If uncertain, set the flag to false.
- Quote short evidence from the Target only, at most 18 words.
- Keep each explanation under 18 words.
- severity_score: 0 for no issue, 1-2 mild, 3-4 clear, 5 severe.
- Return JSON only.

### Output JSON ONLY
{
  "tgt1_judge": {
    "code_extract_failure_detected": bool,
    "reasoning_inefficiency_detected": bool,
    "algorithmic_route_drift_detected": bool,
    "severity_score": int,
    "primary_issue": "Code Extraction / Answer-Format Failure | Reasoning Inefficiency |
    Algorithmic Route Drift | None",
    "evidence_quote": "short quote from Target-1",
    "explanation": "short justification"
  },
  "tgt2_judge": {
    "code_extract_failure_detected": bool,
    "reasoning_inefficiency_detected": bool,
    "algorithmic_route_drift_detected": bool,
    "severity_score": int,
    "primary_issue": "Code Extraction / Answer-Format Failure | Reasoning Inefficiency |
    Algorithmic Route Drift | None",
    "evidence_quote": "short quote from Target-2",
```

System Prompt for Code-R1 Trace Drift Evaluation (continued)

```
    "explanation": "short justification"
  }
}
```

B Algorithmic and Theoretical Details

B.1 TARP Algorithm

Algorithm 1 summarizes the overall TARP procedure, including consensus-guided trace selection and token weighting, followed by block-wise trace sensitivity estimation and the resulting regularization used in subsequent sequential RL updates. After all retained anchors have been processed, we divide each accumulated matrix covariance and vector Fisher statistic by the number of retained anchor losses contributing to the prior. Thus the stored statistics represent empirical second moments rather than unnormalized sums. When gradients are accumulated from batches that represent more than one retained anchor loss, the effective denominator should be the corresponding number of represented retained anchor losses.

B.2 Formal Local Analysis of the TARP Penalty

This section gives a formal local mechanism analysis of the TARP penalty used in our implementation. We do not prove a global convergence or retention guarantee for full GRPO, whose training dynamics additionally include stochastic rollouts, clipping, KL regularization, and practical gradient-ratio caps. Instead, we isolate the frozen quadratic TARP prior loaded from the previous stage and analyze its local effect exactly.

Roadmap. This subsection proceeds in four steps:

- We first introduce a trace-divergence interpretation of the weighted anchor loss in §B.2.1 and its linearized second-moment form in §B.2.2.
- We then discuss the low-rank sketch view used in practice in §B.2.3 and how it motivates the local trace-sensitivity operator approximated by TARP.
- Next, §B.2.4 formalizes the resulting quadratic operator and shows that it is self-adjoint and positive semidefinite.
- Finally, §B.2.5 analyzes the induced local anchor-regularized update.

B.2.1 Trace Divergence View

One way to interpret the weighted anchor loss is as a local surrogate for stability on the retained anchor prefixes around the previous-stage policy. For a prompt x, let $a = (a_1, \ldots, a_T)$ denote a retained anchor sequence produced by the selection-and-weighting procedure at checkpoint θ^*, let $\mathcal{T}(a)$ denote its retained token positions, and define per-sequence normalized weights

$$\bar{w}_t = \frac{w_t}{\sum_{s \in \mathcal{T}(a)} w_s + \epsilon}, \qquad t \in \mathcal{T}(a).$$

A corresponding weighted conditional divergence is

$$D_{\text{anchor}}(\theta, \theta^*) = \mathbb{E}_{x,a}\left[\sum_{t \in \mathcal{T}(a)} \bar{w}_t \, \mathrm{KL}(\pi_{\theta^*}(\cdot \mid x, a_{<t}) \parallel \pi_\theta(\cdot \mid x, a_{<t}))\right].$$

This retained set may include reasoning tokens and, when configured by the masking rule in Section 4, answer or interface tokens as well. TARP does not estimate the full divergence D_{anchor} exactly, and the analysis below does not require such an identity. Instead, it uses gradients of the implemented weighted anchor loss on retained anchor sequences to identify local parameter directions along which these retained conditionals are most sensitive. The resulting quadratic prior should therefore be read as a local approximation to stability on the retained anchor prefixes, not as a guarantee that all semantic reasoning behavior is preserved.

Algorithm 1: TARP: Trace-Anchored Regularization Prior Algorithm

Input: Current policy π_θ, prompt dataset $\mathcal{D}$, reference checkpoint θ^*
Hyperparams: rollouts K, correctness threshold δ, minimum retained-success rule, anchor strategy, consensus threshold τ, shaping exponent ξ, subspace rank r, projection dim d, ϵ
Output: Stored prior $\{\boldsymbol{U}_b, \boldsymbol{\omega}_b, \boldsymbol{P}_b\}_{b\in\mathcal{B}_{\mathrm{mat}}}$, $\{\boldsymbol{f}_v\}_{v\in\mathcal{B}_{\mathrm{vec}}}$, and constraint $\mathcal{R}_{\mathrm{TARP}}$

Initialize gradient covariance estimators $\boldsymbol{\Sigma}_b$ for each matrix block b and diagonal accumulators $\boldsymbol{f}_v$ for each vector parameter v;
Initialize retained-anchor counter $N_{\mathrm{prior}} \leftarrow 0$;
Initialize a Rademacher projection matrix $\boldsymbol{P}_m \in \mathbb{R}^{\mathrm{in}(m)\times d}$ for each matrix parameter m and rescale its entries by $1/\sqrt{\mathrm{in}(m)}$;
Set $\alpha_{\mathrm{proj}} \leftarrow 1/d$ to average over the projected columns;
foreach *prompt* $x \in \mathcal{D}$ **do**
  `// Stage 1: Selection & Weighting`
  Sample K trajectories $\mathcal{Y} = \{y^{(1)}, \ldots, y^{(K)}\}$ by stochastic decoding;
  Filter trajectories $\mathcal{Y}^+ \leftarrow \{y \in \mathcal{Y} \mid r(y, x) \geq \delta\}$;
  if $\mathcal{Y}^+$ *satisfies the configured minimum retained-success rule* **then**
    Select anchor $y^a \in \mathcal{Y}^+$ by the configured strategy, prioritizing the highest consensus score and using reward and rollout order only as tie-breakers;
    Parse the retained portion of y^a into spans $\{\mathcal{S}_i\}_{i=1}^M$, including reasoning spans and any retained answer spans or task-interface spans, and map them to token intervals;
    foreach *span* $\mathcal{S}_i$ **do**
      Compute consensus score c_i against $\mathcal{Y}^+ \setminus \{y^a\}$;
      $\tilde{c}_i \leftarrow \left(\max\left(0, \frac{c_i - \tau}{1 - \tau + \epsilon}\right)\right)^{\xi}$;
      Compute deterministic span rescaling ρ_i from line type and position bias;
    foreach *token position* t *in* y^a **do**
      Compute weight $w_t \leftarrow m_t \cdot \tilde{c}_{\sigma(t)} \cdot \rho_{\sigma(t)}$;
    `// Stage 2: Estimation`
    $\mathcal{L}_{\mathrm{TARP}}(x) \leftarrow \frac{\sum_t w_t\, \ell_t(\theta)}{\sum_t w_t + \epsilon}$;
    Compute gradient $g \leftarrow \nabla_\theta \mathcal{L}_{\mathrm{TARP}}(x)$;
    foreach *matrix parameter* m *and its gradient* g_m **do**
      Reshape g_m into $G_m \in \mathbb{R}^{\mathrm{out}(m)\times \mathrm{in}(m)}$;
      Project: $\tilde{G}_m \leftarrow G_m \boldsymbol{P}_m \in \mathbb{R}^{\mathrm{out}(m)\times d}$;
      Partition $\tilde{G}_m$ along the output dimension into blocks $\{\tilde{G}_{m,b}\}$;
      foreach *block* b **do**
        $\boldsymbol{\Sigma}_b \leftarrow \boldsymbol{\Sigma}_b + \alpha_{\mathrm{proj}} \cdot \tilde{G}_{m,b} \tilde{G}_{m,b}^\top$;
    foreach *vector parameter* v *and its gradient* g_v **do**
      $\boldsymbol{f}_v \leftarrow \boldsymbol{f}_v + g_v \odot g_v$;
    $N_{\mathrm{prior}} \leftarrow N_{\mathrm{prior}} + 1$;
`// Normalize empirical second moments before finalization`
foreach *parameter block* b **do**
  $\boldsymbol{\Sigma}_b \leftarrow \boldsymbol{\Sigma}_b / \max(N_{\mathrm{prior}}, 1)$;
  Perform PCA on $\boldsymbol{\Sigma}_b$ to obtain top-r basis $\boldsymbol{U}_b$ and eigenvalues $\boldsymbol{\omega}_b$;
foreach *vector parameter* v **do**
  $\boldsymbol{f}_v \leftarrow \boldsymbol{f}_v / \max(N_{\mathrm{prior}}, 1)$;
`// Constraint`
$\mathcal{R}_{\mathrm{TARP}}(\theta) \leftarrow$
$\sum_{b\in\mathcal{B}_{\mathrm{mat}}} \frac{\alpha_{\mathrm{proj}}}{2} \left\| \mathrm{Diag}(\sqrt{\boldsymbol{\omega}_b})\, \boldsymbol{U}_b^\top (\theta_b - \theta_b^*) \boldsymbol{P}_b \right\|_F^2 + \sum_{v\in\mathcal{B}_{\mathrm{vec}}} \frac{1}{2} (\theta_v - \theta_v^*)^\top \mathrm{Diag}(\boldsymbol{f}_v)(\theta_v - \theta_v^*)$;
Return $\{\boldsymbol{U}_b, \boldsymbol{\omega}_b, \boldsymbol{P}_b\}_{b\in\mathcal{B}_{\mathrm{mat}}}, \{\boldsymbol{f}_v\}_{v\in\mathcal{B}_{\mathrm{vec}}}$;

B.2.2 Linearized Trace Sensitivity

The next proposition states the exact first-order quantity used by TARP. It does not claim that the implementation recovers the full Hessian of D_{anchor}.

Proposition B.1 (Linearized trace sensitivity)**.** *Let $L_n(\theta)$ denote the weighted anchor loss for the n-th retained anchor sequence and let*

$$g_n = \nabla_\theta L_n(\theta^*).$$

Define the empirical second-moment operator

$$\widehat{\boldsymbol{M}} = \frac{1}{N}\sum_{n=1}^{N} g_n g_n^\top. \tag{10}$$

Then, for any perturbation δ,

$$\delta^\top \widehat{\boldsymbol{M}}\delta = \frac{1}{N}\sum_{n=1}^{N}\langle g_n, \delta\rangle^2. \tag{11}$$

Thus $\delta^\top \widehat{\boldsymbol{M}}\delta$ measures the mean squared first-order change of the retained weighted anchor losses under perturbation δ. The block-wise PCA prior used by TARP is a compact approximation to this empirical linearized sensitivity structure.

Proof. For each n,

$$\delta^\top g_n g_n^\top \delta = (g_n^\top \delta)^2 = \langle g_n, \delta\rangle^2.$$

Summing over n and dividing by N proves Eq. 11. □

This proposition concerns only the first-order sensitivity of the retained weighted anchor loss. It should not be interpreted as a semantic or causal identification of latent reasoning mechanisms. Throughout the spectral discussion below, μ_i denotes an eigenvalue of the frozen quadratic operator $\boldsymbol{M}$. We use it only as a local linearized retained-loss sensitivity measure, not as a guarantee that the direction q_i encodes a reasoning mechanism.

B.2.3 Low-Rank Sketch Interpretation

The block-wise PCA step is a compact approximation to the empirical second-moment operator in Eq. 10. Within each projected block, the retained rank-r basis captures the dominant directions of weighted anchor-loss gradients observed during prior export. Larger r preserves more of the measured trace-sensitive subspace and reduces truncation error, but it also increases storage and runtime cost. The random projection and row-block decomposition further introduce approximation error. Thus the rank and projection choices control a practical coverage-cost trade-off; they do not imply perfect coverage of all reasoning-relevant directions.

B.2.4 The TARP Quadratic Operator

Let $\mathcal{B}_{\text{mat}}$ and $\mathcal{B}_{\text{vec}}$ denote the sets of matrix and vector parameter blocks stored by TARP. For each matrix block $b \in \mathcal{B}_{\text{mat}}$, let $\Delta_b = \theta_b - \theta_b^*$ denote the displacement from the anchor checkpoint, let $\boldsymbol{U}_b \in \mathbb{R}^{m_b \times r_b}$ and $\boldsymbol{\omega}_b \in \mathbb{R}_+^{r_b}$ denote the stored basis and importances, and let $\boldsymbol{P}_b \in \mathbb{R}^{n_b \times d}$ denote the fixed random projection associated with that block. When several row blocks originate from the same matrix parameter, they share the same underlying projection matrix in the implementation; we absorb this association into the block index b. The implementation contributes the block penalty

$$\mathcal{R}_b(\theta) = \frac{\alpha_{\text{proj}}}{2}\left\|\text{Diag}(\sqrt{\boldsymbol{\omega}_b})\,\boldsymbol{U}_b^\top \Delta_b \boldsymbol{P}_b\right\|_F^2,$$

which is equivalently

$$\mathcal{R}_b(\theta) = \frac{\alpha_{\text{proj}}}{2}\,\text{tr}\Big(\boldsymbol{P}_b^\top \Delta_b^\top \boldsymbol{U}_b\,\text{Diag}(\boldsymbol{\omega}_b)\boldsymbol{U}_b^\top \Delta_b \boldsymbol{P}_b\Big).$$

For each vector block $v \in \mathcal{B}_{\text{vec}}$, let $\Delta_v = \theta_v - \theta_v^*$ and let $\boldsymbol{f}_v \geq 0$ denote the stored diagonal Fisher estimate. The implementation contributes

$$\mathcal{R}_v(\theta) = \frac{1}{2}\Delta_v^\top\,\text{Diag}(\boldsymbol{f}_v)\Delta_v.$$

We analyze TARP on the finite-dimensional direct-sum space

$$\mathcal{H} = \Big(\bigoplus_{b \in \mathcal{B}_{\mathrm{mat}}} \mathbb{R}^{m_b \times n_b} \Big) \oplus \Big(\bigoplus_{v \in \mathcal{B}_{\mathrm{vec}}} \mathbb{R}^{d_v} \Big),$$

equipped with the block inner product

$$\langle \Delta, \Delta' \rangle_{\mathcal{H}} = \sum_{b \in \mathcal{B}_{\mathrm{mat}}} \mathrm{tr}(\Delta_b^\top \Delta_b') + \sum_{v \in \mathcal{B}_{\mathrm{vec}}} \Delta_v^\top \Delta_v'.$$

For each matrix block define

$$\boldsymbol{A}_b = \boldsymbol{U}_b \,\mathrm{Diag}(\boldsymbol{\omega}_b) \boldsymbol{U}_b^\top, \qquad \boldsymbol{B}_b = \boldsymbol{P}_b \boldsymbol{P}_b^\top,$$

and for each vector block define $\boldsymbol{D}_v = \mathrm{Diag}(\boldsymbol{f}_v)$. Since $\boldsymbol{\omega}_b \geq 0$ and $\boldsymbol{f}_v \geq 0$ by construction, $\boldsymbol{A}_b$, $\boldsymbol{B}_b$, and $\boldsymbol{D}_v$ are all symmetric positive semidefinite.

In the implementation, each $\boldsymbol{P}_b$ is sampled as a fixed Rademacher matrix and rescaled entrywise by the inverse square root of the input dimension. We write $\alpha_{\mathrm{proj}} = 1/d$ for the same projected-column averaging convention used in Algorithm 1. Its appearance in the covariance export and in the runtime penalty reflects averaging over the d projected columns in both the stored projected-gradient second moments and the projected displacement norms used during regularization.

Now define the linear operator $\boldsymbol{M} : \mathcal{H} \to \mathcal{H}$ blockwise by

$$(\boldsymbol{M}\Delta)_b = \alpha_{\mathrm{proj}} \, \boldsymbol{A}_b \Delta_b \boldsymbol{B}_b, \qquad (\boldsymbol{M}\Delta)_v = \boldsymbol{D}_v \Delta_v. \tag{12}$$

Then the full TARP penalty can be written compactly as

$$\mathcal{R}_{\mathrm{TARP}}(\theta^* + \Delta) = \frac{1}{2} \langle \Delta, \boldsymbol{M}\Delta \rangle_{\mathcal{H}}. \tag{13}$$

Theorem B.2. *The operator $\boldsymbol{M}$ defined in Eq. 12 is self-adjoint and positive semidefinite on $\mathcal{H}$. Consequently, $\mathcal{R}_{TARP}(\theta^* + \Delta) \geq 0$ for all $\Delta \in \mathcal{H}$.*

Proof. Fix any $\Delta, \Delta' \in \mathcal{H}$. For a matrix block b,

$$\begin{aligned} \langle \Delta_b, (\boldsymbol{M}\Delta')_b \rangle_F &= \alpha_{\mathrm{proj}} \, \mathrm{tr}\big(\Delta_b^\top \boldsymbol{A}_b \Delta_b' \boldsymbol{B}_b\big) \\ &= \alpha_{\mathrm{proj}} \, \mathrm{tr}\big((\boldsymbol{A}_b \Delta_b \boldsymbol{B}_b)^\top \Delta_b'\big) \end{aligned}$$

because $\boldsymbol{A}_b^\top = \boldsymbol{A}_b$ and $\boldsymbol{B}_b^\top = \boldsymbol{B}_b$. Hence

$$\langle \Delta_b, (\boldsymbol{M}\Delta')_b \rangle_F = \langle (\boldsymbol{M}\Delta)_b, \Delta_b' \rangle_F.$$

For a vector block v,

$$\langle \Delta_v, (\boldsymbol{M}\Delta')_v \rangle_2 = \Delta_v^\top \boldsymbol{D}_v \Delta_v' = (\boldsymbol{D}_v \Delta_v)^\top \Delta_v' = \langle (\boldsymbol{M}\Delta)_v, \Delta_v' \rangle_2$$

because $\boldsymbol{D}_v^\top = \boldsymbol{D}_v$. Summing over all blocks shows

$$\langle \Delta, \boldsymbol{M}\Delta' \rangle_{\mathcal{H}} = \langle \boldsymbol{M}\Delta, \Delta' \rangle_{\mathcal{H}},$$

so $\boldsymbol{M}$ is self-adjoint.

To prove positive semidefiniteness, consider first a matrix block b. Using $\boldsymbol{A}_b = \boldsymbol{U}_b \,\mathrm{Diag}(\boldsymbol{\omega}_b) \boldsymbol{U}_b^\top$ and $\boldsymbol{B}_b = \boldsymbol{P}_b \boldsymbol{P}_b^\top$,

$$\begin{aligned} \langle \Delta_b, (\boldsymbol{M}\Delta)_b \rangle_F &= \alpha_{\mathrm{proj}} \, \mathrm{tr}\Big(\boldsymbol{P}_b^\top \Delta_b^\top \boldsymbol{U}_b \,\mathrm{Diag}(\boldsymbol{\omega}_b) \boldsymbol{U}_b^\top \Delta_b \boldsymbol{P}_b \Big) \\ &= \alpha_{\mathrm{proj}} \, \Big\| \mathrm{Diag}(\sqrt{\boldsymbol{\omega}_b}) \, \boldsymbol{U}_b^\top \Delta_b \boldsymbol{P}_b \Big\|_F^2 \geq 0. \end{aligned} \tag{14}$$

For a vector block v,

$$\langle \Delta_v, (\boldsymbol{M}\Delta)_v \rangle_2 = \Delta_v^\top \boldsymbol{D}_v \Delta_v = \| \,\mathrm{Diag}(\sqrt{\boldsymbol{f}_v}) \, \Delta_v \|_2^2 \geq 0. \tag{15}$$

Summing Eqs. 14 and 15 over all blocks yields

$$\langle \Delta, \boldsymbol{M}\Delta \rangle_{\mathcal{H}} \geq 0.$$

Therefore $\boldsymbol{M} \succeq 0$, and Eq. 13 implies $\mathcal{R}_{\mathrm{TARP}}(\theta^* + \Delta) \geq 0$. □

B.2.5 Local Anchor-Regularized Composite Step

To understand the effect on optimization, let

$$d_t = \theta_t - \theta^*,$$

and let

$$s = \theta_{t+1} - \theta_t.$$

Consider the local composite model

$$\min_{s \in \mathcal{H}} \langle g_t, s \rangle_{\mathcal{H}} + \frac{1}{2\eta} \|s\|_{\mathcal{H}}^2 + \frac{\lambda}{2} \langle d_t + s, \boldsymbol{M}(d_t + s) \rangle_{\mathcal{H}}, \tag{16}$$

where $g_t \in \mathcal{H}$ is the local RL gradient, $\eta > 0$ is the nominal step size, and $\lambda \geq 0$ is the TARP coefficient.

Theorem B.3. *The objective in Eq. 16 is strongly convex and admits the unique minimizer*

$$s_t^\star = -\left(\eta^{-1}\boldsymbol{I} + \lambda \boldsymbol{M}\right)^{-1} (g_t + \lambda \boldsymbol{M} d_t). \tag{17}$$

Equivalently,

$$d_{t+1}^\star = d_t + s_t^\star = \left(\boldsymbol{I} + \eta\lambda \boldsymbol{M}\right)^{-1} (d_t - \eta g_t). \tag{18}$$

Proof. Define

$$\boldsymbol{H}_{\eta,\lambda} = \eta^{-1}\boldsymbol{I} + \lambda \boldsymbol{M}.$$

By Theorem B.2, $\boldsymbol{M}$ is self-adjoint and positive semidefinite. Therefore, for any nonzero $s \in \mathcal{H}$,

$$\langle s, \boldsymbol{H}_{\eta,\lambda} s \rangle_{\mathcal{H}} = \eta^{-1}\|s\|_{\mathcal{H}}^2 + \lambda \langle s, \boldsymbol{M} s \rangle_{\mathcal{H}} \geq \eta^{-1}\|s\|_{\mathcal{H}}^2 > 0. \tag{19}$$

Hence $\boldsymbol{H}_{\eta,\lambda}$ is positive definite, so Eq. 16 is strongly convex and has a unique minimizer.

Taking the Fréchet derivative of Eq. 16 with respect to s gives the first-order optimality condition

$$g_t + \eta^{-1} s_t^\star + \lambda \boldsymbol{M}(d_t + s_t^\star) = 0,$$

or equivalently

$$\left(\eta^{-1}\boldsymbol{I} + \lambda \boldsymbol{M}\right) s_t^\star = -(g_t + \lambda \boldsymbol{M} d_t).$$

Since $\boldsymbol{H}_{\eta,\lambda}$ is invertible by Eq. 19, we obtain the closed form

$$s_t^\star = -\boldsymbol{H}_{\eta,\lambda}^{-1}(g_t + \lambda \boldsymbol{M} d_t) = -\left(\eta^{-1}\boldsymbol{I} + \lambda \boldsymbol{M}\right)^{-1} (g_t + \lambda \boldsymbol{M} d_t),$$

which proves Eq. 17.

Adding d_t to both sides gives

$$\begin{aligned} d_{t+1}^\star &= d_t - \left(\eta^{-1}\boldsymbol{I} + \lambda \boldsymbol{M}\right)^{-1} (g_t + \lambda \boldsymbol{M} d_t) \\ &= \eta \left(\boldsymbol{I} + \eta\lambda \boldsymbol{M}\right)^{-1} \left(\eta^{-1} d_t - g_t\right) \\ &= \left(\boldsymbol{I} + \eta\lambda \boldsymbol{M}\right)^{-1} (d_t - \eta g_t), \end{aligned}$$

which proves Eq. 18. □

Corollary B.4. *Let $\boldsymbol{M} q_i = \mu_i q_i$ with $\mu_i \geq 0$, and let*

$$d_{t+1}^{\text{vanilla}} = d_t - \eta g_t$$

denote the anchor displacement produced by the same local Euclidean step without TARP. Then

$$d_{t+1}^\star = \left(\boldsymbol{I} + \eta\lambda \boldsymbol{M}\right)^{-1} d_{t+1}^{\text{vanilla}}. \tag{20}$$

Along eigendirection q_i,

$$\langle q_i, d_{t+1}^\star \rangle_{\mathcal{H}} = \frac{\langle q_i, d_t \rangle_{\mathcal{H}} - \eta \langle q_i, g_t \rangle_{\mathcal{H}}}{1 + \eta\lambda\mu_i}, \tag{21}$$

and

$$\langle q_i, s_t^\star \rangle_{\mathcal{H}} = -\frac{\eta\left(\langle q_i, g_t \rangle_{\mathcal{H}} + \lambda\mu_i \langle q_i, d_t \rangle_{\mathcal{H}}\right)}{1 + \eta\lambda\mu_i}. \tag{22}$$

Thus, TARP shrinks the local vanilla displacement from the anchor by the factor $(1+\eta\lambda\mu_i)^{-1} \in (0, 1]$. Directions with larger eigenvalues μ_i of the frozen sketched trace-sensitivity operator are suppressed more strongly, while directions in the nullspace of $\boldsymbol{M}$ are unaffected by this quadratic prior. When $d_t = 0$, Eq. 22 reduces to the simpler gradient-shrinkage formula from the anchored-at-initialization special case. For $d_t \neq 0$, however, the update also contains the anchor pull-back term $\lambda \boldsymbol{M} d_t$, so it is not merely a rescaled gradient step.

Proof. Because $\boldsymbol{M}$ is self-adjoint on a finite-dimensional inner-product space, the spectral theorem gives an orthonormal eigenbasis $\{q_i\}_{i=1}^{D}$ with eigenvalues $\mu_i \geq 0$. Applying Eq. 18 to each eigenvector yields

$$\langle q_i, d_{t+1}^{\star} \rangle_{\mathcal{H}} = \left\langle q_i, (\boldsymbol{I} + \eta\lambda \boldsymbol{M})^{-1} (d_t - \eta g_t) \right\rangle_{\mathcal{H}} = \frac{\langle q_i, d_t \rangle_{\mathcal{H}} - \eta \langle q_i, g_t \rangle_{\mathcal{H}}}{1 + \eta\lambda\mu_i},$$

which proves Eq. 21. Since $s_t^{\star} = d_{t+1}^{\star} - d_t$, Eq. 22 follows immediately. Finally, Eq. 20 shows that the same factor $(1+\eta\lambda\mu_i)^{-1}$ rescales the local vanilla anchor displacement along each eigendirection. □

C Additional Experimental Results and Discussion

C.1 Component Attribution Statistics

Table 8 provides the full statistics underlying Figure 4. EWC is matched to TARP in regularization gradient strength. Randomized uses the same numbers of retained tokens and spans as TARP. Unweighted uses TARP's selected support with unit weights.

Table 8: Component attribution on Qwen2.5-3B-Instruct after 200 CD steps followed by 150 KK steps.

Method	CD Reward ↑	KK Reward ↑	CD Acc ↑	KK Full ↑	CD Ret-Fail ↓	CD Churn ↓
GRPO	0.74 ± 0.05	$\mathbf{0.78 \pm 0.08}$	73.68 ± 4.45	$\mathbf{37.21 \pm 3.51}$	0.44 ± 0.08	3.91 ± 0.27
EWC	0.74 ± 0.06	0.68 ± 0.02	73.03 ± 5.21	32.93 ± 1.48	0.83 ± 0.08	4.45 ± 0.23
Randomized	0.74 ± 0.05	0.69 ± 0.07	68.97 ± 1.06	32.79 ± 3.69	0.69 ± 0.14	4.89 ± 0.26
Unweighted	0.76 ± 0.06	0.74 ± 0.07	74.79 ± 4.52	35.00 ± 2.18	0.34 ± 0.09	$\mathbf{3.20 \pm 0.13}$
Full TARP	$\mathbf{0.77 \pm 0.05}$	0.75 ± 0.09	$\mathbf{75.39 \pm 3.84}$	36.28 ± 4.39	$\mathbf{0.25 \pm 0.07}$	3.72 ± 0.30

C.2 Diagonal Fisher EWC Baseline

In addition to the shortened attribution study, we also include a pure diagonal Fisher EWC baseline at the longer endpoint of the Qwen2.5-3B-Instruct CountDown→KK-Logic setting. The comparison starts from the model at the end of the CountDown stage and continues through the end of the KK-Logic stage with $\lambda = 0.01$. Table 9 compares the endpoint from one training run against the sequential baseline and TARP checkpoints trained with the same seed, using the same CountDown retention and KK plasticity metrics as Table 1. Generic Fisher regularization slightly improves retention on CountDown and reduces Retained-Fail to zero, but incurs a noticeably larger cost to KK plasticity than either the baseline or TARP.

Table 9: Results per method on Qwen2.5-3B-Instruct after CountDown→KK-Logic training.

	Forgetting on CountDown			Plasticity on KK	
Method	CD Acc ↑	CD Churn ↓	CD Ret-Fail ↓	KK Full ↑	KK Reward ↑
Baseline	69.92	4.49	0.49	55.43	1.42
TARP	70.48	4.18	0.48	55.00	1.47
Diagonal Fisher EWC	70.12	4.98	0.00	44.86	1.07

C.3 Diagnostics of Beneficial and Harmful Trace Changes

The symmetric TARP penalty does not determine whether an individual deviation from the anchor is semantically beneficial or harmful. We therefore use two complementary diagnostics conducted after training to test whether the observed retention pattern extends beyond preservation of output formatting or exact old strategies.

First, we compare the reference checkpoint after 200 CD training steps with the GRPO and TARP checkpoints after 300 additional KK training steps, using three stochastic decoding repeats. We retain only prompts answered correctly by all three checkpoints, normalize their traces, and count a

changed trace as shorter or longer when its length differs from the reference by at least 20%. Table 10 reproduces the reported variation across decoding repeats. TARP permits at least as many changes to shorter correct traces as GRPO and fewer changes to longer correct traces. This shows that the prior does not mechanically freeze all concise, correct trace changes, but it does not establish that shorter traces are semantically superior or that every beneficial deviation remains unpenalized.

Table 10: Changes in output length among correctly answered prompts from the reference.

Change relative to reference	GRPO	TARP
At least 20% shorter	8.80 ± 0.95	$\mathbf{11.32 \pm 0.56}$
At least 20% longer	13.73 ± 1.87	$\mathbf{12.14 \pm 1.81}$

Second, we conduct a blinded human audit on 200 prompts sampled uniformly without replacement from the Qwen2.5-7B Code-R1 test set of 712 prompts. For each prompt, three annotators independently compare the shared reference response after 300 R1 training steps with the matched GRPO and TARP responses after 300 additional KK training steps. Candidate identities are randomized independently per sample. Annotators see the task, required entry point, public tests, reference response, and two anonymized candidates, but not the method mapping or the other annotators' labels. They assign four binary flags using the rubric below and do not execute candidate code or hidden tests.

Flag	Operational criterion
Harmful drift	visible degradation relative to the reference in reasoning reliability, solution route, or required answer interface
Format failure	loss of an extractable implementation or required answer structure that is present in the reference
Reasoning inefficiency	substantial repetitive or generic text, or text resembling a tutorial, without corresponding algorithmic progress
Route drift	a change in the core algorithm, data structure, invariant, or task assumption that is not equivalent to the reference

Table 4 in the main paper reports the majority vote for each sample across the three annotators.

C.4 Task Order

We examine a different task order using Qwen2.5-3B-Instruct trained on Code-R1-12k for 100 steps followed by KK-Logic for 100 steps in Table 11. After KK-Logic training, TARP retains substantially more of the preceding Code-R1 behavior than GRPO. R1 reward is 0.43 versus -0.96, and Pass@1 is 33.80 versus 3.24. GRPO adapts more strongly to the current KK task, reaching reward 1.21 and full accuracy 48.86, compared with 1.06 and 44.57 for TARP. Thus the retention advantage also appears when Code-R1 is placed first, but with a measurable cost to adaptation on the current task.

The same run also provides a view of retention and trace changes over time. Figure 9 tracks R1 retention while the model trains on KK-Logic. At step 20, the GRPO R1 reward remains at 0.50, equal to the value at the start of the stage, while mean output length has already increased from 22.64 to 25.25 lines. By step 40, mean output length reaches 87.94 and reward falls to -0.55. TARP keeps mean output length near 22 lines throughout and retains substantially higher R1 reward. In this run, trace growth precedes the sharp decline in R1 reward.

Table 11: Task order experiment on Qwen2.5-3B-Instruct, trained on Code-R1-12k for 100 steps followed by KK-Logic for 100 steps.

Stage	Method	R1 Reward	KK Reward	R1 Pass@1	KK Full	R1 Extract	KK Extract	R1 Ret-Fail	KK Ret-Fail
R1	-	0.50	–	40.74	–	99.07	–	0.00	–
KK	GRPO	−0.96	**1.21**	3.24	**48.86**	97.69	100.00	2.31	**51.14**
KK	TARP	**0.43**	1.06	**33.80**	44.57	**99.07**	100.00	**0.93**	55.43

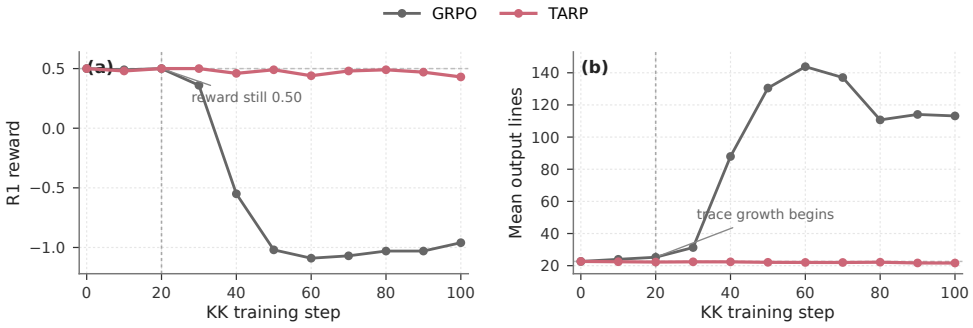

Figure 9: R1 retention over time. The vertical dotted line marks step 20, where GRPO reward is still unchanged although its traces have begun to lengthen. TARP keeps both reward and mean output length comparatively stable.

C.5 Resource Footprint and Training Overhead

Tables 12 and 13 evaluate resource usage and endpoint rewards at different retained ranks on Qwen2.5-3B, with projection dimension fixed at $d = 48$. The default retained rank is $r = 32$.

Table 12: Dedicated profile of Qwen2.5-3B at different retained ranks with $d = 48$. Memory is peak allocated GPU memory and time is seconds per training step.

Stage	Method	Rank r	Memory (GB)	Time per step (s)
KK-Logic	GRPO	–	38.10	137.40
	TARP	8	54.54	141.62
	TARP	16	55.43	145.67
	TARP	32	56.50	150.60
Code-R1-12k	GRPO	–	37.60	174.00
	TARP	8	55.09	216.58
	TARP	16	55.23	223.99
	TARP	32	56.50	229.60

At $r = 32$, peak memory exceeds GRPO by 48.3% on KK-Logic and 50.3% on Code-R1-12k. Reducing r to 8 saves 3.5% and 2.5% memory, and reduces step time by 6.0% and 5.7%, respectively. Most memory overhead comes from cached anchor parameters and temporary full parameter views for the FSDP penalty, so reducing rank provides limited savings. Larger savings would require sharding or eliminating the anchor cache and avoiding temporary materialization of unsharded parameters.

Among the tested ranks, $r = 32$ achieves the highest reward on both tasks (Table 13). The intermediate rank is not consistently better than $r = 8$, so these results support the default choice without establishing monotonic improvement with rank.

C.6 Limitations and Future Work

Our evaluation covers staged GRPO on three structured tasks using Qwen2.5 and Llama. Broader generalization remains untested, and retention gains can coexist with higher Retained-Fail, Churn, or verbosity. Human and model judgments remain auxiliary. Annotators do not execute hidden tests,

Table 13: Effect of retained rank on Qwen2.5-3B after 200 CD steps followed by 300 KK steps, with $d = 48$. Values are endpoint rewards.

Rank r	CountDown	KK-Logic
8	0.6387	1.1793
16	0.6611	1.0171
32	**0.6719**	**1.2814**

and model judges lack calibration against human labels. These flags should complement answer metrics and deterministic verifiers.

Future work will explore more robust segmentation methods for reasoning traces with diverse structures and formatting. Evaluations on harder benchmarks will test whether the observed retention benefits extend to more challenging tasks. Experiments across broader model families will assess the generality of these findings.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Section 1 state the operational definition of reasoning drift, the scope of TARP, and the principal empirical claims and trade-offs, which are reflected in the experiments and conclusion.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Appendix C.6 discusses generalization and evaluation limitations, including mixed effects on retention metrics. Appendix B.2 states the scope of the local theoretical analysis.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Appendix B.2 states the assumptions and provides proofs for linearized loss sensitivity, positive semidefiniteness of the frozen quadratic prior, and the idealized local composite update.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Sections 4 and 5, together with Appendix A, describe the method, task sequences, hyperparameters, and evaluation protocols needed to reproduce the experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Code, processed data, environment specifications, and reproduction commands are available at `https://github.com/WanNaa/TARP`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5 and Appendix A provide model and dataset details, training configurations, reward functions, and evaluation settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Table 1 reports mean $\pm$ standard deviation over three training seeds, and Appendix Table 8 reports the same statistics over three independent runs. Joint references and Llama results are point estimates. Comparisons are descriptive.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide the GPU configuration, peak memory usage, and time per training step in Appendices A.2 and C.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The study uses established reasoning benchmarks and reports aggregate model-level results. The auxiliary human audit evaluates anonymized model outputs, collects no personal or sensitive participant attributes, and is reported only in aggregate. We identify no deviation from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

Answer: [N/A]

Justification: This work studies a general training regularizer and diagnostic methodology for sequential reasoning rather than a deployed application, user-facing system, or high-risk released asset. Any societal effects are indirect and shared with general improvements in language-model post-training.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release a high-risk pretrained model, image generator, scraped dataset, or other asset that requires controlled-release safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We cite the original sources of the models, datasets, and software used in this work and follow their respective licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The public TARP repository is the new software asset introduced with the paper. Its accompanying documentation states the release scope and exclusions, repository layout, pinned environment, data provenance and checksums, required base models, reproduction commands, output locations, limitations, Git-based versioning, and Apache-2.0 license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [Yes]

    Justification: Appendix C.3 reports the sampling procedure, materials shown to annotators, per-sample randomization and blinding, the complete four-flag operational rubric, the no-execution instruction, and majority-vote aggregation. No screenshot-based participant interface is used in the reported protocol.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

Answer: [N/A]

Justification: The annotators provide task-level labels of model outputs; the study does not treat annotators as research subjects, analyze their personal characteristics or behavior, or collect personal or sensitive data. Only aggregate model-level flag rates are reported.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: Sections 4 and 5 describe the language models used as RL policies, and Appendix A.3 documents the auxiliary LLM-as-Judge protocol and full evaluation prompt.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.