ENTROPY TARGETING IN SEQUENTIAL REINFORCEMENT FINE-TUNING OF LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential reinforcement fine-tuning asks a language model to keep improving as its tasks change, yet the auxiliary signals added to stabilize such training are rarely checked for whether they reach the policy update at all. We study this question in AdaCoT-CL, a group-relative sequential learner that combines three consistency signals (entropy targeting on the reasoning span, alignment between reasoning and answer representations, and answer agreement) with replay, and we reanalyze 63 recorded runs on a five-task reasoning stream, 59 of them with Qwen3-8B. We first show that the agreement signal supplies no direct policy gradient: its group-shared reward bonus cancels whenever rewards are centered within each prompt group, and its auxiliary loss is detached. This result partitions the seven ablation variants into four effective objectives and motivates a grouped comparison. In that comparison, configurations with entropy targeting improve on average and those without it decline. On fixed monitoring panels, the 12 runs with entropy targeting gain 13.2 accuracy points from initialization to the end of the sequence, while the 9 runs without it lose 8.7 points (difference 21.8 points, exact permutation $p = 8.8 \times 10^{-5}$; a contrast averaged within seed and alignment blocks gives 24.1 points, $p = 1.7 \times 10^{-4}$), and the primary comparison replicates this direction against five baselines. The largest divergence occurs between the end of the LogiQA phase and step 32 of the final task, a block that combines training on the new task with replay of earlier prompts: mean accuracy on the earlier panels rises by 27 points on average with entropy targeting, and the smallest such rise, 16.7 points, exceeds the largest without it, 4.2 points. The evidence has clear boundaries: the panels are drawn from training prompts, initial accuracy is low under a 512-token completion budget, the benefit shrinks under other task orders, and the same objective collapses Llama-3.1-8B-Instruct. We therefore present entropy targeting as the consistency signal that distinguishes improving from declining configurations of this learner, and we specify the held-out protocol needed to establish it as a continual-learning method.

1 INTRODUCTION

Language models are increasingly improved by reinforcement learning against automatic verifiers, and in practice this training rarely happens only once. New task families arrive over time, and each round of fine-tuning should add a capability without eroding those already learned. This is the continual reinforcement learning problem (Khetarpal et al., 2022; Abel et al., 2023), now posed for a policy whose actions are tokens and whose rewards come from answer checkers (Shao et al., 2024; Guo et al., 2025). Recent studies find that reinforcement learning forgets less than supervised fine-tuning in their settings (Shenfeld et al., 2025; Chen et al., 2026; Lai et al., 2026). They motivate a more specific question: which of the auxiliary signals added to a sequential reinforcement learner actually act on its update, and which of those sustain improvement under a limited completion budget?

A common response is to add auxiliary signals that constrain how the policy changes. Reasoning models offer several cheap candidates: the entropy of the tokens in a reasoning trace, the relation between the representations of the reasoning and the answer, and the agreement among sampled answers. A signal regularizes the policy directly only if it has an effective gradient path, however, and whether it does depends on implementation details such as group normalization of rewards and

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

which tensors are detached; it can also act indirectly, through replay selection or coefficient updates, so the two roles must be examined separately. When this is not checked, an ablation can credit an effect to a component that cannot act.

We study this problem in AdaCoT-CL, a group-relative sequential learner that combines all three consistency signals with replay, a probe-triggered replay intervention, and a scalar coefficient controller (Figure 1). We ask three questions in order: which signals reach the policy update, which of those changes outcomes across a task sequence, and how and when its effect appears or fails. To answer them, we combine a gradient-path analysis of the implemented objective with a reanalysis of 63 recorded runs on a five-task reasoning stream, 59 of them with Qwen3-8B (Yang et al., 2025).

The answers are clear. The agreement signal supplies no direct policy gradient (Proposition 1), so the seven ablation variants collapse into four effective objectives (Section 3.2). Grouped this way, configurations with entropy targeting improve on average and those without it decline, also under permutation tests that preserve seed and alignment, while all 15 runs of the five primary baselines decline. Much of the separation arises between the end of the LogiQA phase and step 32 of the final task, a block that combines training on the new task with replay (Figure 2). Entropy targeting acts as a self-annealing, two-sided entropy regularizer (Section 3.2), and the timing suggests that it keeps the policy able to gain on earlier tasks later in the sequence, a hypothesis the protocol of Appendix N can test.

These findings have boundaries that we treat as part of the result (Section 4.5): every panel consists of training prompts, the completion budget shapes what accuracy can measure, the size of the benefit depends on task order, and the same objective fails on Llama-3.1-8B-Instruct. Our claim is therefore specific: the findings characterize the tested pipeline on its monitored training-prefix panels, and identifying the mechanism and establishing held-out retention require separate validation. We make three contributions:

- **A gradient-path analysis** of consistency regularization for group-relative learners, which identifies the effective objectives against which ablations must be read (Section 3).
- **A retrospective reanalysis** of 63 runs, with exact permutation tests that also preserve seed and alignment, and a phase decomposition that locates when the separation arises (Section 4).
- **An analysis of sensitivity** to task order and model family under the tested completion budget, with a held-out protocol for turning the finding into a continual-learning claim (Section 4.5, Appendix N).

2 Problem and comparable measurements

We consider T task-delimited prompt distributions $\mathcal{D}_1, \ldots, \mathcal{D}_T$ with correctness rewards. A language policy $\pi_\theta(y \mid x)$ learns the current task, optionally using a bounded memory of earlier prompts. The objective is to acquire useful behavior while preserving earlier capabilities. This finite sequence is a restricted continual-RL setting (Abel et al., 2023); our experiments do not evaluate open-ended environmental adaptation.

Let θ_0 denote the policy before any task training and θ_j the policy after task j. Write $\widehat{A}_j(\theta; D)$ for four-sample plurality-answer accuracy on panel D. With the same held-out panel at both checkpoints, backward transfer is conventionally measured as

$$\mathrm{BWT}(D) = \frac{1}{T-1} \sum_{j<T} [\widehat{A}_j(\theta_T; D_j) - \widehat{A}_j(\theta_j; D_j)]. \tag{1}$$

This task-matrix comparison separates performance after acquisition from its subsequent change (Lopez-Paz & Ranzato, 2017).

What the archived comparison measures. Our acquisition evaluations use 64-prompt panels V_j, while initial and later monitoring evaluations use their first 16 prompts, M_j. Both belong to the training source pool. The original reported quantity is therefore the *logged contrast*

$$\Delta_{\log} = \frac{1}{T-1} \sum_{j<T} [\widehat{A}_j(\theta^{\log}_{T,j}; M_j) - \widehat{A}_j(\theta_j; V_j)]. \tag{2}$$

The notation $\theta^{\log}_{T,j}$ permits replay updates between task evaluations. No such update occurs during the final monitoring pass of the primary runs; it does occur in some sensitivity runs. For any fixed scoring function, each summand decomposes as

$$\underbrace{A_j(\theta^{\log}_{T,j}; M_j) - A_j(\theta_j; M_j)}_{\text{matched-prefix retention}} + \underbrace{A_j(\theta_j; M_j) - A_j(\theta_j; V_j)}_{\text{acquisition-panel discrepancy}}. \tag{3}$$

Because acquisition scores on M_j were not separately retained, the second term is unknown. Consequently, $\Delta_{\log}$ alone does not identify even the sign of matched-prefix retention. Appendix M explains this ambiguity and distinguishes panel composition from stochastic decoding variation.

A matched-panel quantity that is recoverable. The initial and final monitoring entries use the same stored list M_j in the implementation. They permit the separate within-run comparison

$$\Delta_{\text{init}} = \frac{1}{T-1} \sum_{j<T} [\widehat{A}_j(\theta^{\log}_{T,j}; M_j) - \widehat{A}_j(\theta_0; M_j)]. \tag{4}$$

Unlike $\Delta_{\log}$, this statistic does not mix 64- and 16-prompt panels. It measures net change from before the task sequence, including acquisition and subsequent learning; it is neither BWT nor held-out generalization. We introduce it as an additional descriptive analysis of existing logs, not a new evaluation or a replacement for Equation 1. For the primary order, both contrasts average GSM8K, ARC, CommonsenseQA, and LogiQA, excluding the final task, SVAMP. Appendix E reports every seed and a further comparison between two later monitoring checkpoints.

How variable is a single evaluation? Every evaluation samples four completions per prompt at temperature 0.7, so repeated evaluations of one policy differ. For the untrained policy, evaluated on the identical panels $M_1, \ldots, M_4$ at the start of all 59 Qwen3-8B runs, the four-task mean is 0.316 with standard deviation 0.033 (range 0.250 to 0.391), and we did not detect a difference in mean initial scores across the seven primary conditions (one-way ANOVA, $p = 0.40$); two such evaluations differ by about 0.046 in standard deviation. This describes variation before training only, so we use it as a descriptive reference; inference rests on between-run permutation tests (Section 4, Appendix F).

3 Consistency regularization and its optimization paths

3.1 A group-relative sequential learner

For each prompt x_i, the learner samples $G = 4$ completions and obtains binary verifier rewards r_{ig}. It standardizes rewards within the group to form clipped advantages $\widehat{A}_{ig}$. Let $\ell_\theta(y \mid x)$ be the mean non-padding generated-token log probability, and let ℓ_{old} be the score retained during sampling. The implemented policy term is

$$d_{ig} = \text{clip}(\ell_\theta - \ell_{\text{old}}, -10, 10), \qquad \rho_{ig} = \exp(d_{ig}), \tag{5}$$

$$\mathcal{L}_{\text{pg}} = -\frac{1}{NG} \sum_{i,g} \min\{\rho_{ig}\widehat{A}_{ig}, \text{clip}(\rho_{ig}, 1-\epsilon, 1+\epsilon)\widehat{A}_{ig}\}. \tag{6}$$

The objective is inspired by PPO and GRPO (Schulman et al., 2017; Shao et al., 2024), with two implementation distinctions. It uses a ratio formed from *sequence-mean* scores rather than tokenwise clipped terms, and penalizes $\mathbb{E}[d^2]$ rather than a distributional KL to a frozen reference model. Sampling scores also undergo generation-time transformations that update scores do not. Thus, we use "group-relative baseline" for the implemented learner; an exact PPO importance-ratio interpretation is unwarranted.

The configured objective adds consistency terms and an optional importance penalty:

$$\mathcal{L} = \mathcal{L}_{\text{pg}} + \beta\mathbb{E}[d^2] + m_B\alpha_B\mathcal{L}_B + m_A\alpha_A\mathcal{L}_A + m_C\alpha_C\mathcal{L}_C + \mathcal{L}^{\text{eff}}_{\text{imp}}. \tag{7}$$

Here $m_\cdot \in \{0, 1\}$ defines the consistency mask, and replay changes the training buffer. The intended importance term penalizes movement from a stored anchor with a squared-gradient proxy distinct from EWC (Kirkpatrick et al., 2017); in the recorded executions every importance estimate before a later task failed, so $\mathcal{L}^{\text{eff}}_{\text{imp}} = 0$ in all recorded updates (Section 4.1 and Appendix B.5).

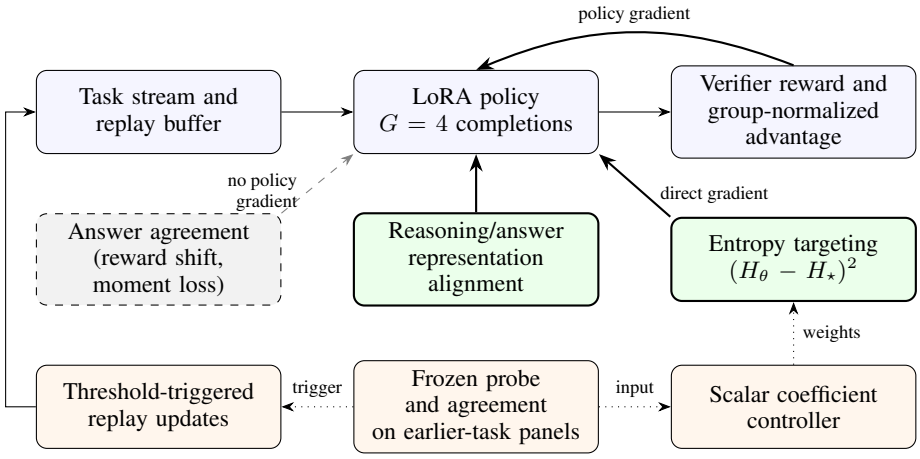


Figure 1: What reaches the policy update in AdaCoT-CL. Thick arrows carry policy gradients. Answer agreement enters as a reward shift and a detached moment loss, neither of which supplies a policy gradient (Proposition 1). Dotted arrows show measurements and scalar settings; in the primary runs the controller receives zero input and follows a common coefficient schedule (Section 3.3). Algorithm 1 gives the order of operations.

3.2 What the consistency components measure

The first component, called BCT in the supplied implementation, targets the mean token entropy H_θ over the response's reasoning span:

$$\mathcal{L}_B = \mathbb{E}[(H_\theta - H_\star)^2], \qquad H_\star = \mathrm{softplus}(2) \simeq 2.127 \text{ nats}. \tag{8}$$

Because this term penalizes deviation in both directions, its gradient shows what it does: $\nabla_\theta \, \alpha_B (H_\theta - H_\star)^2 = 2\alpha_B (H_\theta - H_\star) \nabla_\theta H_\theta$, which at the current parameters equals the gradient of an entropy bonus $-\lambda H_\theta$ with $\lambda = 2\alpha_B (H_\star - H_\theta)$. Entropy targeting is therefore a self-annealing, two-sided entropy regularizer: below the target it raises entropy with a strength that shrinks as the target is approached, and above the target it lowers entropy (Proposition 2 in Appendix B.4). This places it between two familiar tools. A fixed entropy bonus rewards higher entropy without an interior target (WILLIAMS & PENG, 1991; Mnih et al., 2016), whereas soft actor-critic adapts a temperature until a target entropy is reached (Haarnoja et al., 2019). In entropy targeting the multiplier follows the common scalar schedule, while each response's signed deviation sets the direction and strength of its correction, which is strongest when entropy is furthest from the target, which matters for reasoning models whose entropy tends to collapse under reinforcement learning (Cui et al., 2025; Yu et al., 2025). The term never compares the policy with an earlier one, so any retention benefit must arise indirectly. The target is an implementation constant, not a tuned value, and the logs record only the squared deviation (Section 4.4).

The second component, ACT, aligns mean final-layer representations of the reasoning and answer spans:

$$\mathcal{L}_A = \mathbb{E}[1 - \cos(\bar{h}_{\text{think}}, \bar{h}_{\text{answer}})]. \tag{9}$$

Its scope is representational similarity, not semantic entailment or reasoning faithfulness; Appendix B specifies the delimiter detection and fallback spans that decide which tokens are regularized.

The third component uses plurality agreement $c_i = \max_a G^{-1} \sum_g \mathbf{1}\{\text{answer}(y_{ig}) = a\}$. The code calls this quantity SCE, although it is not an entropy. It adds the same agreement-dependent reward offset to every completion in a group and compares detached agreement moments with a previous-task prior.

Proposition 1 (Agreement paths in the implemented update)**.** *An additive reward offset shared by all completions of a prompt leaves their centered, standardized advantages unchanged in exact arithmetic. Moreover, the agreement-moment auxiliary has zero direct gradient with respect to policy parameters when its current statistics and prior are detached.*

Algorithm 1 Tested sequential framework: order of operations

Input: task order, condition, seed, LoRA policy, training-prefix panels.
Initialize enabled modules; record pre-sequence scores on each M_j.
For task $k = 1, \ldots, T$ **do**
  Construct the task's fixed current/replay buffer.
  For ordinary step $s = 1, \ldots, 64$ **do**
    Sample grouped completions; update the policy; collect probe features.
    If $s \in \{32, 64\}$ **then**
      Request a probe fit.
      If $k > 1$ **then**: monitor each earlier task in order, allowing enabled targeted replay after its check; then update enabled controller scalars. **EndIf**
    EndIf
  EndFor
  Finalize retention state and memory; refit/freeze the probe; record acquisition scores and the agreement prior.
EndFor; save matrices and reconstruct Δ_{init} and $\Delta_{\log}$ (Algorithm 6).

The proof is given in Appendix B.4. The proposition decides how the ablations must be read. At the same policy state, batch, and auxiliary state, switching the agreement term on or off changes no policy update, and the controller cannot reintroduce a difference because its objective sums all three coefficients regardless of the mask (Corollary 1 in Appendix G). The seven consistency masks therefore implement only four effective objectives: no active auxiliary (agreement only), alignment, entropy, and entropy plus alignment, giving 12 runs with entropy targeting and 9 without it. Gradient equivalence does not make separately executed runs independent: runs with the same seed label start from the same generator state, although their data order can diverge once branches that consume random draws, such as interventions, differ. Section 4.3 therefore also tests the grouping with permutations that preserve seed and alignment.

3.3 Monitoring, coefficient updates, and targeted replay

To monitor the learner, a logistic probe predicts completion correctness from the final generated-token representation. A copy frozen after each task is compared with its acquisition baseline, and if the drop exceeds 0.17, the full variant attempts two extra updates on that task's replay examples; the no-intervention variant omits this branch.

Separately, the controller updates consistency weights and replay fraction by matching scalar targets. Its implemented input averages positive differences between an acquisition probe baseline and *current agreement*, mixing two signal types. Reconstruction gives zero for every one of its eight updates in all six primary full/no-intervention runs. Their coefficient trajectories are identical, ending at $\alpha_B = \alpha_A = \alpha_C = 0.6734$ and replay fraction 0.4900. The experiments therefore test a common coefficient schedule, not feedback adaptation (Appendix B.6).

4 Experiments: which signal matters, and where it fails

4.1 Protocol, reasoning budget, and uncertainty

The primary model is Qwen3-8B (Yang et al., 2025) with rank-16 LoRA (Hu et al., 2021), trained in order on GSM8K, ARC-Challenge, CommonsenseQA, English-filtered LogiQA2.0, and SVAMP (Cobbe et al., 2021; Clark et al., 2018; Talmor et al., 2019; Liu et al., 2023; Patel et al., 2021). Each task receives 64 updates of eight prompts with four completions each. Generation uses the model's thinking mode with a 512-token completion budget, and evaluation scores the plurality answer of four samples on the training-prefix panels of Section 2. Every worker log records successful loading of all five tasks (Appendix A).

The 512-token budget defines the regime in which our results hold. Under it, the untrained Qwen3-8B policy reaches only 0.255 on GSM8K and 0.079 on LogiQA, and in 59 initial evaluations LogiQA never reaches the four-option chance level of 0.25. On the same panels and prompts, Llama-3.1-8B-Instruct (Grattafiori et al., 2024), which has no native thinking mode, averages 0.875

Table 1: Primary Qwen3-8B results (three seeds; mean ± sample SD). Initial and final use the same 16 training-prefix prompts per earlier task, and Δ_{init} is their difference; $\Delta_{\log}$ uses 64-prompt acquisition scores. All 15 baseline runs have negative Δ_{init}. Neither quantity is held-out BWT.

Condition	Initial	Final	Δ_{init}	$\Delta_{\log}$
Group-relative	0.3385	0.1875	-0.1510 ± 0.0502	-0.0859 ± 0.0681
Nominal importance	0.3281	0.2083	-0.1198 ± 0.0239	-0.0807 ± 0.0060
Replay	0.2969	0.1719	-0.1250 ± 0.0271	-0.1315 ± 0.0558
Projection control	0.2969	0.2083	-0.0885 ± 0.0180	-0.0651 ± 0.0410
Nominal imp. + replay	0.3125	0.1562	-0.1562 ± 0.0413	-0.1133 ± 0.0408
Full, no intervention	0.2969	0.3854	$+0.0885 \pm 0.1721$	$+0.0573 \pm 0.1064$
Full	0.3177	0.4479	$+0.1302 \pm 0.0916$	$+0.1107 \pm 0.0844$

to 0.900 across the five panels (Appendix F). Truncated reasoning and answer-format failures are plausible contributors to the low Qwen3 accuracy, but completion lengths and per-example outputs were not saved, so their contributions cannot be measured from this record. Gains in this regime can therefore include learning to answer within the budget, a useful competence for budgeted inference (Aggarwal & Welleck, 2025) but not the same as retaining reasoning skill.

Two execution facts matter: the importance penalty never became active (Section 3; all 16,637 logged values are zero), so the EWC-style conditions are nominal rather than tests of EWC, and sampling errors removed one and three updates from two runs, while the other 61 record all 320.

Each training run is one observation, and we report every seed. Three-seed comparisons are descriptive (Agarwal et al., 2021; Colas et al., 2018); grouped comparisons use exact permutation tests, over all run labels and within blocks that preserve seed and alignment, with within-group bootstrap intervals. All grouping and timing analyses are retrospective: permutation probabilities assume exchangeability under the stated relabeling, which enumeration does not establish, and the bootstrap ignores possible dependence among runs sharing a seed label.

4.2 Primary comparison: the full configuration improves where every baseline declines

Table 1 compares the seven primary conditions on the matched-panel change Δ_{init}. The full configuration raises mean accuracy on the four earlier-task panels from 0.318 to 0.448, a change of $+0.130 \pm 0.092$, and all three seeds are positive ($+0.063$, $+0.234$, $+0.094$). Every baseline moves the other way: the five baseline conditions have mean changes between -0.156 and -0.089, and all 15 of their runs are negative.

The baselines' decline is informative in its own right: training on later tasks leaves the earlier panels below the untrained model, and replay alone does not prevent it (-0.125). The full configuration's recorded mean acquisition accuracy on the earlier-task panels is also higher than the group-relative baseline's (0.337 versus 0.273). The historical $\Delta_{\log}$ keeps the same ordering but mixes panel sizes (Appendix E, Figure 4).

Because the primary comparison changes several components at once, the consistency ablation is needed to say which one is responsible.

4.3 With entropy targeting, configurations improve on average; without it, they decline

The seven ablation variants share everything except their consistency masks, so Table 2 and Figure 2a group their 21 runs by the four effective objectives of Section 3.2.

Entropy targeting is the factor that divides the outcomes. The two objectives that include it improve on average ($+0.138$ and $+0.125$), and the two that exclude it decline (-0.042 and -0.177); 11 of the 12 runs with entropy targeting have positive Δ_{init}, and 8 of the 9 without it negative. The pooled difference is $+0.218$ (within-group bootstrap interval $[0.144, 0.295]$), it is positive within every seed ($+0.146$, $+0.245$, $+0.264$), and a contrast that weights the six seed-by-alignment blocks equally is $+0.241$. In exact permutation tests, 26 of the 293,930 relabelings of all runs reach the pooled

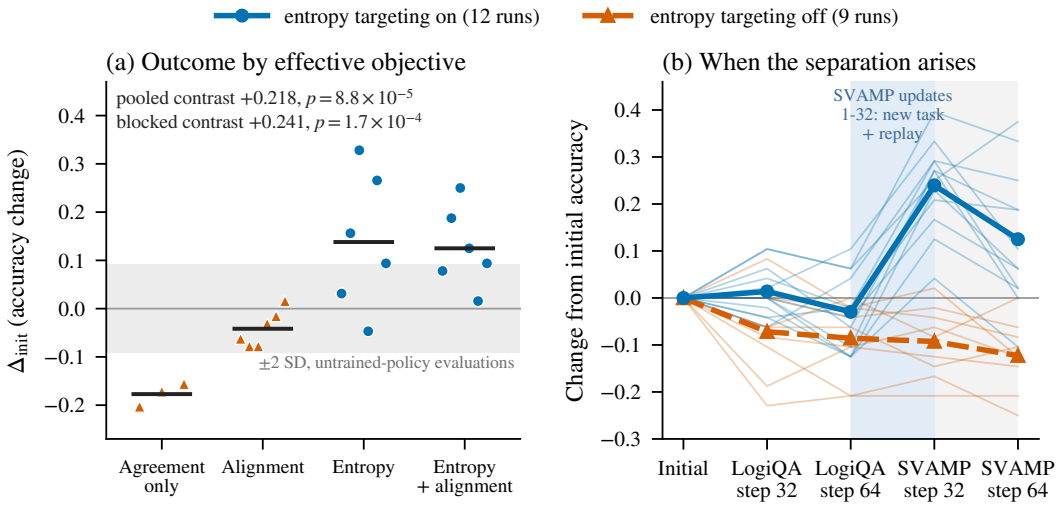


Figure 2: The evidence in one view, drawn directly from the saved reward matrices. (a) Matched-panel change Δ_{init} for each of the 21 consistency-ablation runs, grouped by effective objective; bars are class means, and the shaded band (±2 SD of the difference between two evaluations of the untrained policy) is a descriptive reference, not a confidence interval. (b) Change from initial accuracy on the GSM8K, ARC, and CommonsenseQA panels at the checkpoints all three share; thin lines are runs and thick lines group means. Over the first 32 updates of the final task (shaded blue), which combine SVAMP training with replay, the smallest rise with entropy targeting (0.167) exceeds the largest without it (0.042).

Table 2: Consistency ablation grouped by effective objective (Section 3.2), one run-level Δ_{init} per run. Bottom rows: the pooled contrast (entropy on minus off) with a within-group bootstrap interval and Hedges' g, and the contrast averaged over six equally weighted seed-by-alignment blocks, each with its exact permutation p.

Effective objective	Mask variants	Runs	Mean Δ_{init}	SD	Range
No active auxiliary	agreement only	3	-0.177	0.024	$[-0.203, -0.156]$
Alignment	alignment only; no entropy	6	-0.042	0.038	$[-0.078, +0.016]$
Entropy	entropy only; no alignment	6	$+0.138$	0.142	$[-0.047, +0.328]$
Entropy + alignment	no agreement; all three	6	$+0.125$	0.083	$[+0.016, +0.250]$
Pooled contrast, 12 vs 9 runs: $+0.218$, 95% interval $[0.144, 0.295]$, $g = 2.15$, $p = 8.8 \times 10^{-5}$					
Seed-by-alignment contrast, six blocks weighted equally: $+0.241$, $p = 1.7 \times 10^{-4}$					

difference ($p = 8.8 \times 10^{-5}$), and no other of the 5,832 relabelings within blocks reaches the blocked contrast ($p \approx 1.7 \times 10^{-4}$); blocking preserves the seed labels and the unequal share of alignment runs.

The association with entropy targeting is positive both without alignment ($+0.315$) and with it ($+0.167$), so it holds across alignment settings, although the size of the interaction remains uncertain (Appendix G).

Four checks guard against artifacts. The separation holds on each of the four tasks (per-task exact $p \leq 0.037$); it is not explained by weaker learning of each new task, since acquisition accuracy is slightly higher with entropy targeting (0.326 versus 0.303); $\Delta_{\log}$ separates the groups in the same way (exact $p = 6.5 \times 10^{-5}$); and removing runs with a replay intervention in the final monitoring pass leaves a difference of $+0.183$, though only two runs remain without entropy targeting (Appendix G).

4.4 When and how the separation arises

Most of the observed separation develops between the end of the LogiQA phase and step 32 of the final task. Figure 2b and Table 15 (Appendix I) split the change on the three earliest panels (GSM8K, ARC, CommonsenseQA) into three phases. Up to the end of the LogiQA phase, runs with and without entropy targeting differ by only 0.056 (exact $p = 0.14$). Between that checkpoint and step 32 of the final task, a block of 16 updates on SVAMP followed by 16 updates on replayed earlier prompts, accuracy on these panels rises by 0.269 in the entropy-targeted runs and changes by -0.007 without entropy targeting. The smallest rise with entropy targeting (0.167) exceeds the largest without it (0.042; exact $p = 3.4 \times 10^{-6}$), and the separation persists when runs with an intervention in either endpoint pass are excluded ($+0.275$ in 9 runs against -0.029 in 5). About 43% of the rise reverses over the last 32 updates. The rise appears on every panel, at $+0.31$ on GSM8K, $+0.31$ on ARC, and $+0.19$ on CommonsenseQA, and the primary full and no-intervention series show it while the 15 baseline runs do not (Appendix I).

The two checkpoints locate the rise but cannot show whether it is sudden, nor separate transfer from SVAMP from relearning through replay. Earlier phases with equally large replay blocks show no comparable rise (-0.02 and -0.03), and across the five orders, four of six arithmetic phases exceed a rise of 0.15 against none of nine multiple-choice phases; but all four large rises occur in the fifth task, so arithmetic content is not separated from accumulated training, final-task position, or replay composition (Appendix I). Finishing the reasoning within the budget could raise every panel at once, but the logs cannot confirm it (Section 4.1). The pattern suggests that entropy targeting does less to protect earlier skills step by step than to leave the policy able to gain on them later, a plasticity hypothesis (Lyle et al., 2023; Dohare et al., 2024) that this record cannot isolate from replay, sequence position, formatting, or ratio dynamics.

The training logs are consistent with Section 3.2 but cannot confirm its direction. In the primary full runs the root-mean-square gap to the entropy target shrinks from about 1.5 nats over the first eight updates to about 0.4 over the last eight, and the score-gap second moment $\mathbb{E}[d^2]$ is 6.6 times larger with entropy targeting (Appendix H). The logs record neither the sign of the entropy gap nor signed score gaps or clipping frequencies, so they cannot show whether entropy rose or how often the clipped branch of the policy term was active. In a single-seed sweep that varies all three consistency coefficients jointly, Δ_{init} is -0.172 at $\alpha = 0.01$, reaches $+0.250$ at $\alpha = 0.2$, and falls to $+0.109$ at $\alpha = 0.5$ (Appendix K.3).

4.5 Where the effect weakens or fails

Task order matters. Under the four alternative orders, run only with the full configuration, mean Δ_{init} over 12 runs is $+0.022$ with five runs positive, against $+0.130$ and three of three in the primary order (Appendix K.4). Without baselines under those orders, and with content and position confounded in the timing analysis, we can say only that the size of the benefit depends on order.

Model family matters more. In single-seed runs, the full configuration moves Δ_{init} from -0.016 to $+0.219$ on Qwen3-14B but from -0.031 to -0.766 on Llama-3.1-8B-Instruct. Llama starts at 0.75 to 1.00 on every task, yet after training its own-task accuracy on the four later tasks is between 0.08 and 0.44, while its logged entropy deviation falls from 1.24 to 0.11. When reasoning delimiters are absent or undetected, the fallback span can place answer and padding positions under the entropy term; the records cannot show whether this happened for Llama or caused its collapse, but the possibility argues for restricting entropy targeting to a detected reasoning span with a per-model target.

Finally, the replay intervention is not identified: it fired twice in the three primary full runs, and the largest paired difference from the no-intervention variant occurs in a seed without any trigger (Appendix L.1).

5 Related work

Continual learning and continual RL. Classical defenses against forgetting protect important parameters (Kirkpatrick et al., 2017), replay past data (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019), or consolidate after acquisition (Schwarz et al., 2018). In reinforcement learning, CLEAR

combines replay with behavioral cloning (Rolnick et al., 2019), synaptic models slow consolidation (Kaplanis et al., 2018), Continual World measures forgetting and transfer together (Wołczyk et al., 2021; Wolczyk et al., 2022), and continual RL has been applied to controlled text generation (Shulev & Sima'an, 2024). Continually trained networks also lose plasticity (Nikishin et al., 2022; Lyle et al., 2023; Abbas et al., 2023; Dohare et al., 2024), so forgetting must be read together with learning (Chaudhry et al., 2018; Díaz-Rodríguez et al., 2018); our timing analysis motivates, but cannot establish, a link between entropy targeting and this plasticity side.

Forgetting in language-model post-training. Continual fine-tuning degrades earlier abilities of language models (Luo et al., 2025; Kotha et al., 2024; Shi et al., 2024), and low-rank or orthogonal updates trade learning for retention (Biderman et al., 2024; Wang et al., 2023a). Recent studies report less forgetting with reinforcement learning than with supervised fine-tuning in their settings, explained by proximity to the base policy (Shenfeld et al., 2025), the mode-seeking behavior induced by on-policy data (Chen et al., 2026), or implicitly regularized, selective updates (Lai et al., 2026). Our implemented baseline declines in this regime, but because its scoring and policy objective differ from those methods (Section 3), this is not a direct test of their proposed protection mechanisms.

Entropy in policy optimization. Entropy bonuses encourage exploration (WILLIAMS & PENG, 1991; Mnih et al., 2016) and smooth the optimization landscape (Ahmed et al., 2019), and soft actor-critic tunes a temperature toward a target entropy (Haarnoja et al., 2019). In reasoning language models, entropy collapses early in reinforcement learning and performance saturates with it (Cui et al., 2025); decoupled clipping (Yu et al., 2025) and updates restricted to high-entropy forking tokens (Wang et al., 2025) counteract this, yet entropy minimization alone can also help (Agarwal et al., 2025). The useful direction of entropy change thus depends on the regime, and our results extend the question to sequential training.

Group-relative objectives, budgets, and internal signals. GRPO standardizes rewards within a group (Shao et al., 2024), and its normalization terms introduce length and difficulty biases (Liu et al., 2025; Ahmadian et al., 2024); Proposition 1 follows the same logic. Budgeted reasoning is itself a training objective (Aggarwal & Welleck, 2025), so our budget is part of the regime. Self-consistency, semantic entropy, and hidden-state probes show that correctness is partly recoverable from outputs and representations (Wei et al., 2022; Wang et al., 2023b; Kuhn et al., 2023; Kadavath et al., 2022; Azaria & Mitchell, 2023), but probe accuracy alone does not show what a representation encodes (Hewitt & Liang, 2019), so our monitors remain descriptive.

6 Discussion and conclusion

We asked which auxiliary consistency signals act on the update of a sequential reinforcement learner and which accompany improved learning. Analytically, the agreement signal supplies no direct policy gradient: its group-shared bonus cancels whenever rewards are centered within each prompt group, and its moment loss is detached. Empirically, configurations with entropy targeting improve on average on the monitored panels and those without it decline: grouped by effective objective, runs with it gain 13.2 points on average and runs without it lose 8.7, and the difference holds within every seed and under blocked permutations. Much of the separation arises in the first training-and-replay block of the final task; this localization motivates controlled tests of how entropy targeting, replay, and sequence position interact. The methodological lesson is to audit auxiliary signals for gradient paths before ablating them, since a detached or normalized-away signal yields arms that implement the same objective.

Limitations. Beyond the failure cases of Section 4.5, three limits bound every conclusion: we measure change on trained prompts rather than held-out retention; without saved outputs, the content of the rise, plausibly answering within the budget, cannot be observed; and entropy targeting always co-occurred with replay and may interact with the implemented sequence-level ratio. Appendix N specifies the experiments that address each limit.

REPRODUCIBILITY STATEMENT

The appendix specifies the implemented objective, training and reconstruction algorithms, hyperparameters, task loaders, all reward-contrast seeds, the matched-panel reanalysis, and the worker-log audit. Numerical results were reconstructed from a 63-run archive rather than newly generated. The archive contains source and aggregate logs but no model weights or per-example predictions; it therefore supports the tabulated reconstruction, not retrospective held-out reevaluation. Appendix A documents successful data loads, hardware and attention fallback, failed importance estimates, and the missing example manifests and locked environment. The effective-objective reanalysis (Algorithm 7), the evaluation variability (Appendix F), the optimization diagnostics (Appendix H), and the timing analysis (Appendix I) use only saved reward matrices and training logs, and can be recomputed from the archive without new generations. The supplementary material contains the unaltered archive (its SHA-256 is in Appendix A), a manifest giving the condition, seed, model, and task order of all 63 runs, and two scripts that read the archive without modifying it: `make_figures.py` draws Figure 2, and `reanalysis.py` recomputes every grouped, blocked, timing, and variability statistic reported here. Its README lists checksums and the expected outputs, and separates these reconstructions from the prospective experiments of Appendix N.

AI USE STATEMENT

We designed AdaCoT-CL, implemented its training code, and ran all 63 training runs; the resulting logs, included unaltered in the supplementary material, are the only experimental data in this paper, and we did not use generative AI tools to generate, modify, or clean any of them. We used generative AI tools to help formulate and prove the mathematical statements in Section 3 and Appendix B.4, to design, implement, and interpret the retrospective reanalyses of those logs (the effective-objective grouping, the phase decomposition, and the permutation tests), and to propose the hypotheses discussed in Section 4.4. Additionally, we used them to audit our implementation against the recorded logs, to draft and edit parts of the manuscript, to create Figures 1 and 2, and to identify relevant literature and check bibliographic metadata. We have reviewed all AI-assisted work: every reported statistic is recomputed from the unaltered archive by the supplementary scripts, we checked the proofs, and we verified every reference against its source. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup (eds.), *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pp. 620–636. PMLR, 22–25 Aug 2023. URL https://proceedings.mlr.press/v232/abbas23a.html.

David Abel, Andre Barreto, Benjamin Van Roy, Doina Precup, Hado van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 50377–50407. Curran Associates, Inc., 2023. doi: 10.52202/075280-2192. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/9d8cf1247786d6dfeefeeb53b8b5f6d7-Paper-Conference.pdf.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. 2025. URL https://arxiv.org/abs/2505.15134.

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. 2025. URL https://arxiv.org/abs/2503.04697.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 151–160. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ahmed19a.html.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=aloEru2qCG. Featured Certification.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. 2019. URL https://arxiv.org/abs/1902.10486.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting. 2026. URL https://arxiv.org/abs/2510.18874.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. 2021. URL https://arxiv.org/abs/2110.14168.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. 2018. URL https://arxiv.org/abs/1806.08295.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. 2025. URL https://arxiv.org/abs/2505.22617.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. 2018. URL https://arxiv.org/abs/1810.13166.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni,

Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models. 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao,

Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. 2019. URL https://arxiv.org/abs/1812.05905.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2733–2743, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 2021. URL https://arxiv.org/abs/2106.09685.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2497–2506. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/kaplanis18a.html.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. 2022. URL https://arxiv.org/abs/2012.13490.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. 2024. URL https://arxiv.org/abs/2309.10105.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. 2023. URL https://arxiv.org/abs/2302.09664.

Song Lai, Haohan Zhao, Rong Feng, Changyi Ma, Wenzhuo Liu, Hongbo Zhao, Xi Lin, Dong Yi, Qingfu Zhang, Hongbin Liu, Gaofeng Meng, and Fei Zhu. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training. 2026. URL https://arxiv.org/abs/2507.05386.

Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. Logiqa 2.0—an improved dataset for logical reasoning in natural language understanding. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 31:2947–2962, July 2023. ISSN 2329-9290. doi: 10.1109/TASLP.2023.3293046. URL https://doi.org/10.1109/TASLP.2023.3293046.

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. 2025. URL https://arxiv.org/abs/2503.20783.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. 2025. URL https://arxiv.org/abs/2308.08747.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23190–23211. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/lyle23b.html.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/mniha16.html.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/nikishin22a.html.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. 2019. URL https://arxiv.org/abs/1811.11682.

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

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International*

Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pp. 4528–4537. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/schwarz18a.html.

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

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. 2025. URL https://arxiv.org/abs/2509.04259.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. 2024. URL https://arxiv.org/abs/2404.16789.

Velizar Shulev and Khalil Sima'an. Continual reinforcement learning for controlled text generation. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 3881–3889, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.343/.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. 2025. URL https://arxiv.org/abs/2506.01939.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. 2023b. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

RONALD J. WILLIAMS and JING PENG. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991. doi: 10.1080/09540099108946587. URL https://doi.org/10.1080/09540099108946587.

Maciej Wołczyk, MichałZając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. In M. Ranzato,

A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 28496–28510. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/ef8446f35513a8d6aa2308357a268a7e-Paper.pdf.

Maciej Wolczyk, MichałZając, Razvan Pascanu, Ł ukasz Kuciński, and Piotr Mił oś. Disentangling transfer in continual reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 6304–6317. Curran Associates, Inc., 2022. doi: 10.52202/068431-0456. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/2938ad0434a6506b125d8adaff084a4a-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Appendix roadmap. Every number in these appendices is recomputed from the saved archive; no model was trained or evaluated anew. Two labels recur in the supplied plots and are defined once here: conditions labelled EWC or "importance" are nominal, because their penalty never became active (Appendix A), and axes labelled BWT_reward, BWT_probe, or BWT_sce show the historical contrasts $\Delta_{\log}$, Δ_P, and Δ_C, not backward transfer in the sense of Equation 1. Appendix A documents the record and its execution. Appendix B gives the complete objective, the proofs of Propositions 1 and 2, and the controller. Appendix C states every algorithm, including the reanalysis procedure (Algorithm 7), and Appendix D lists data, optimization, and evaluation settings. Appendices E to I contain the analyses behind Sections 2 and 4: matched-panel changes, evaluation variability and the budget regime, the effective-objective reanalysis with Corollary 1, optimization diagnostics, and the timing of the separation. Appendix J explains the execution mismatch between nominally identical launches, Appendix K reports every recorded run, Appendices L and M cover intervention diagnostics and measurement validity, and Appendix N specifies the prospective protocol.

A Experimental record and provenance

This appendix distinguishes the configured procedure from its recorded execution. The runtime audit resolves the loader and hardware questions left open by the source alone and identifies an important discrepancy: the configured importance regularizer contributed zero to every recorded ordinary update in the relevant runs.

A.1 Record inventory and source correspondence

The archive contains 184 files and 63 unique run tags (Table 3). Separate JSON downloads contain 55 unique tags; the archive supplies the eight remaining alternate-order records. Duplicate downloads of two component records were deduplicated by their internal tags and do not constitute additional replications. All archived reward, probe, and agreement retention summaries match reconstruction from their task/checkpoint matrices.

Table 3: Experimental coverage. Every archived run supplies all four earlier-task contrast terms and final-task step-64 monitoring entries. Seeds identify training runs, not independent task samples.

Family	Configurations	Runs	Seeds
Primary comparison	7	21	0, 1, 2
Consistency masks	7	21	0, 1, 2
Fixed coefficients	5	5	0
Cross-model comparison	4	4	0
Alternate task orders	4	12	0, 1, 2
Total	27	63	

The 13 separately supplied Python files are byte-identical to their archived counterparts. Differences between loose and archived versions of a run's JSON are confined to postprocessed retention and negative-retention summaries; underlying matrices and logs agree. This establishes consistency of the supplied artifacts, but does not independently establish the exact source revision executed during training. Dependencies specify minimum versions rather than a fully locked environment.

A.2 Dataset loads, workers, and hardware

The 69 archived log files comprise 63 unique worker logs, five launcher logs, and one byte-identical worker-log duplicate. Worker tags match all 63 result records, and every worker records a saved-result marker. Each worker logs ten successful dataset loads: the five tasks at both `train n=64` and `train n=256`, totaling 630 successful requests. The 16-prompt monitoring panels derive from the 64-prompt panels and have no separate loader call. No worker logs the loader's synthetic-fallback message. Thus, the recorded runs show successful configured dataset loading, rather than evidence of synthetic substitution. The source nevertheless evaluates nested prefixes of the training

pool; successful loading does not make these panels held out. Without saved example identifiers or content hashes, parsing and exact dataset contents cannot be independently checked or locked retrospectively.

Every worker reports an NVIDIA H200 and one visible GPU. Every model-load log also records an `ImportError` followed by fallback from the requested FlashAttention2 implementation to SDPA. H200 and SDPA therefore describe the recorded execution. Launcher concurrency and shared-device memory usage prevent equating summed worker wall times with exclusive GPU-hours.

A.3 Importance-estimation failures and completed updates

Fifty-two runs enabled the importance regularizer. All 208 estimates after tasks 1–4, which could have affected subsequent ordinary training, failed with CUDA out-of-memory messages. Including final-task calls, 259 of 260 estimates failed (Table 4). The sole call without a failure message follows the final SVAMP task in the Llama full-method run; its importance state was not saved and cannot affect already completed ordinary training.

Table 4: Recorded importance-estimation failures. The middle column covers estimates preceding later training tasks; the last column includes final-task estimates. Enabled denotes configuration, not an effective penalty.

Family	Enabled runs	Pre-later-task failures	All failures
Primary comparison	12	48/48	60/60
Consistency masks	21	84/84	105/105
Fixed coefficients	5	20/20	25/25
Cross-model comparison	2	8/8	9/10
Alternate task orders	12	48/48	60/60
Total	52	208/208	259/260

The estimator initializes a zero accumulator, catches failures, and replaces its stored importance state with the accumulator. Crucially, all 16,637 recorded ordinary `ewc_loss` entries in these 52 runs are exactly zero. This directly establishes the absence of a logged importance-penalty contribution; it does not require inferring every internal accumulator value from an exception message. Accordingly, nominal importance-enabled rows are not valid tests of functioning EWC, nor do they establish gains on top of functioning EWC plus replay. They remain separate conditions because failed calls can still consume randomness, computation, and memory.

Four additional sampling OOMs produced 319 ordinary updates for baseline seed 0 and 317 for the no-intervention variant, seed 1; the other 61 runs record 320 each. The total is 20,156, versus 20,160 configured updates. The saved `n_skipped` field resets at each task and reports only the final task, so its zero value cannot validate earlier steps. No model-update error messages or nonfinite recorded training scalars were found. Silent exception paths in evaluation, probe fitting, and intervention updates limit stronger claims of execution completeness.

A.4 Reconstruction limits and artifact identity

The archive contains aggregate summaries, diagnostics, and logs, but no model or adapter checkpoints, optimizer states, per-example predictions, or example identifiers. The saved initial and final monitoring scores support the matched-panel initial-to-final reconstruction in Appendix E. They do not recover the missing acquisition-checkpoint scores on the 16-prompt panels. Reconstructing acquisition-relative matched-panel retention or evaluating unseen-test performance therefore requires additional artifacts or new runs. For exact identification, the archive's SHA-256 is:

`07507f9e28043e731d956ba36958540c776c908ced2ca9c1ebac301b5b448cfc`

This fingerprint is an integrity identifier, not a public artifact-access statement or evidence of independent reproduction.

B COMPLETE OBJECTIVE AND MECHANISM ANALYSIS

B.1 SAMPLING, REWARDS, AND SEQUENCE SCORES

A prompt x_i produces four sampled completions y_{ig}. Numeric answers receive reward one when the parsed value is within 10^{-3} of the target. Multiple-choice answers are compared with the target after extracting an A–E answer letter. Let k_i denote the optional prompt-level agreement reward shift. The standardized advantage is

$$\widehat{A}_{ig} = \mathrm{clip}\left(\frac{r_{ig} + k_i - \bar{r}'_i}{s'_i + 10^{-8}}, -a_{\max}, a_{\max}\right), \quad \bar{r}'_i = G^{-1}\sum_g (r_{ig} + k_i). \tag{10}$$

The clipping bound is $a_{\max} = \mathrm{softplus}(1.6) + 1$, and the standard deviation s'_i uses the framework's sample-SD default. When all rewards agree, centered advantages are zero. This case can be frequent for groups of four binary outcomes, but the supplied aggregate logs do not support a quantitative frequency estimate.

For generated-token indicator q_t, the update score is

$$\ell_\theta(y \mid x) = \frac{\sum_t q_t \log p_\theta(y_t \mid x, y_{<t})}{\max(\sum_t q_t, 1)}. \tag{11}$$

Thus $\exp(\ell_\theta - \ell_{\mathrm{old}})$ is a geometric-mean token-probability ratio, not the product of token ratios for the whole sequence. The old scores come from generation with temperature, top-p, top-k, and safe-logit processing; update scores come from an unwarped forward-pass log-softmax. Generation uses evaluation mode, while updates use training mode. These details prevent an exact importance-sampling identity for Equation 6. Furthermore, the update forward call does not pass the prompt attention mask, so equivalence with a fully masked conditional-likelihood implementation is not assumed.

B.2 SPAN CONSTRUCTION AND DIFFERENTIABLE REGULARIZERS

The code searches tokenizer encodings of reasoning and answer delimiters to construct spans. If detection fails, it assigns the first 70% of generated width to reasoning and the last 30% to the answer. These masks are not intersected with the non-padding generated-token mask. Equations 8 and 9 therefore describe regularization on the *implemented spans*, which can differ from linguistically meaningful reasoning and answer segments.

Writing $T(y)$ and $U(y)$ for those spans, the entropy and representation statistics are

$$H_\theta(y) = -\frac{1}{\max(|T(y)|, 1)} \sum_{t \in T(y)} \sum_v p_\theta(v \mid x, y_{<t}) \log p_\theta(v \mid x, y_{<t}), \tag{12}$$

$$\bar{h}_{\mathrm{think}} = \frac{1}{\max(|T(y)|, 1)} \sum_{t \in T(y)} h_t, \qquad \bar{h}_{\mathrm{answer}} = \frac{1}{\max(|U(y)|, 1)} \sum_{t \in U(y)} h_t. \tag{13}$$

Both depend differentiably on the current policy forward pass. Their mathematical roles are uncertainty targeting and within-response alignment. Neither term compares a response with a verified reasoning chain, a counterfactual trace, or a previous policy's output distribution.

B.3 AGREEMENT-MOMENT AUXILIARY

At task completion, a 20-bin histogram of prompt-level plurality agreement becomes the next task's prior. Its moments (μ_0, v_0) are computed by weighting points from `linspace`(0,1,20) with normalized histogram masses. Let (μ, v) be the microbatch mean and sample variance of detached agreement values, with numerical stabilization. The auxiliary is

$$\mathcal{L}_C = \frac{1}{2}\left[\log\frac{v_0}{v} + \frac{v + (\mu - \mu_0)^2}{v_0} - 1\right]. \tag{14}$$

It is zero when no prior exists. The expression compares Gaussian moments; it is not a categorical divergence between the full histograms. Agreement is repeated over a prompt's sampled completions, so a microbatch statistic should not be interpreted as four independent observations of agreement per prompt.

B.4 Proofs of Propositions 1 and 2

Proof of Proposition 1. For any prompt-specific scalar k_i,

$$(r_{ig} + k_i) - G^{-1} \sum_{g'} (r_{ig'} + k_i) = r_{ig} - G^{-1} \sum_{g'} r_{ig'}. \tag{15}$$

Translation leaves within-group variance unchanged. Therefore both the numerator and denominator in Equation 10 are unchanged, as is the clipped advantage. The implemented shift $k_i = \alpha_C(c_i - c_0)$ consequently disappears from this policy-learning signal in exact arithmetic.

For the auxiliary, the sampled agreement array is converted from numerical data and detached before entering the consistency calculation; its prior is updated without a gradient graph. Consequently $\partial\mu/\partial\theta = \partial v/\partial\theta = 0$, and the chain rule gives $\nabla_\theta \mathcal{L}_C = 0$. Multiplication by a scalar α_C cannot restore a missing path to θ. The policy optimizer excludes scalar-controller parameters, and the controller zeroes their accumulated gradients before optimizing its own objective. Hence neither auxiliary scalar gradients nor the shared reward shift supplies the asserted direct agreement path. □

The entropy result used in Section 3.2 is stated formally here.

Proposition 2 (Entropy targeting is a self-annealing, two-sided entropy regularizer)**.** *For a response y with span entropy $H_\theta(y)$, $\nabla_\theta \, \alpha_B \big(H_\theta - H_\star\big)^2 = 2\alpha_B \big(H_\theta - H_\star\big) \nabla_\theta H_\theta$. Hence, at the current parameters, the term has the same gradient as an entropy bonus $-\lambda H_\theta$ with coefficient $\lambda = 2\alpha_B(H_\star - H_\theta)$. The coefficient is positive when $H_\theta < H_\star$, shrinks linearly as entropy approaches the target, and becomes negative, turning the bonus into an entropy penalty, when $H_\theta > H_\star$.*

Proof of Proposition 2. For fixed $H_\star$ and a scalar weight α_B that the policy optimizer does not update, the chain rule gives $\nabla_\theta \, \alpha_B (H_\theta - H_\star)^2 = 2\alpha_B (H_\theta - H_\star) \nabla_\theta H_\theta$. An entropy bonus $-\lambda H_\theta$ with λ held fixed has gradient $-\lambda \nabla_\theta H_\theta$. The two gradients coincide at the current parameters when $\lambda = 2\alpha_B(H_\star - H_\theta)$, which is positive for $H_\theta < H_\star$, zero at $H_\theta = H_\star$, and negative for $H_\theta > H_\star$. Because the loss averages over responses, the same identity holds response by response, with each response weighted by its own deviation. The equivalence is local: λ changes as H_θ changes, which is what makes the regularizer self-annealing. □

B.5 Importance, replay, and projection controls

The nominal importance control attempts to use 24 sampled prompts to generate responses, forms a batch-mean reward-weighted sequence log-probability objective, and squares its gradient coordinatewise. The result is an importance proxy: squaring a gradient after averaging is generally different from averaging per-example squared gradients. At a task boundary, the current proxy and parameter anchor overwrite the previous stored state; importances are not summed across all tasks. With stored values $(F_k, \theta_k^\star)$, the penalty is

$$\mathcal{L}_{\text{imp}} = \lambda \sum_k F_k (\theta_k - \theta_k^\star)^2. \tag{16}$$

Four accumulation chunks are used, and the penalty is included only in the first chunk before division by four. Its effective full-update contribution in Equation 7 is therefore $\mathcal{L}_{\text{imp}}/4$.

This expression describes the configured estimator, not an active constraint: Appendix A documents that every estimate able to affect later training failed and that every logged ordinary penalty is zero.

Replay uses a task-keyed memory with limit 128. At the start of a new task, the scalar mixture determines how many current and stored prompts enter a fixed buffer of up to 256 prompts. Current examples precede replay examples, and ordinary updates cycle through this buffer. The mixture is not resampled independently at every update. Task finalization supplies only the first 128 entries of this buffer to the memory-addition routine, which should not be conflated with uniformly retaining 128 examples from all experiences.

The O-LoRA-inspired control records up to eight leading left singular vectors of each LoRA-B matrix at a task boundary. Stored bases are concatenated and orthogonalized, and hooks project subsequent B-matrix gradients away from their span. A-matrix gradients are not projected. This

implements a restricted orthogonal-gradient control; it does not reproduce independently selectable task adapters from Wang et al. (2023a). These details explain why the primary table names the control as inspired by O-LoRA rather than claiming a canonical replication.

B.6 Scalar controller and observed trajectory

Let b_j be the acquisition mean of task j's frozen probe, falling back to agreement when the probe is unavailable, and let $c_j(t)$ be its current mean agreement. The controller uses

$$f_t = \frac{1}{|\mathcal{S}_t|} \sum_{j \in \mathcal{S}_t} \max(0, b_j - c_j(t)), \tag{17}$$

$$a_t^\star = f_t \,\mathrm{softplus}(u), \qquad m_t^\star = \sigma(f_t \,\mathrm{softplus}(v) + o), \tag{18}$$

$$\mathcal{L}_h = (\alpha_B + \alpha_A + \alpha_C - a_t^\star)^2 + (m - m_t^\star)^2. \tag{19}$$

The targets depend on trainable scales and offset; their dependence is not stopped. One Adam step with learning rate 0.005 optimizes the scalar module. This is direct target matching, not differentiation through policy updates, held-out validation-loss minimization, or bilevel meta-learning. The sum includes all three coefficients even when an ablation mask disables a component.

Although configuration contains a nominal hyper-update period of 16, the actual call occurs after earlier-task monitoring at steps 32 and 64. There are eight calls in a five-task main run. Reconstructing f_t yields eight zeros in each of the six full/no-intervention runs. Their common loss sequence is

$$(4.37747, 4.34534, 4.31342, 4.28171,\\ 4.25020, 4.21891, 4.18783, 4.15697).$$

The three consistency coefficients start at $\log 2$ and end at 0.67337334; the replay fraction starts at 0.5 and ends at 0.49002340. Since a task buffer is constructed once, within-task changes to m take effect only when a later buffer is constructed. The fixed-α routine freezes every scalar, which further limits interpretation of the fixed-weight comparison as a single-variable ablation.

C Full algorithms

The following algorithms expand Algorithm 1. Algorithms 2 to 5 describe the supplied implementation, and Algorithms 6 and 7 describe the transparent reanalysis used in this manuscript. In particular, they retain interleaved monitoring, training-prefix evaluation, and the detached agreement path; they do not substitute a prospective corrected algorithm for the one that produced the record.

Algorithm 2 Sequential training with explicit update and monitoring gates

Input: task order $1{:}T$, condition, seed, LoRA policy, configured defaults.
Output: score matrices, diagnostic logs, trigger records, scalar snapshot.

Seed libraries; initialize the policy optimizer, scalar module, global probe buffer, and enabled retention/controller modules.

Load each 64-prompt training-prefix panel V_j and its 16-prompt prefix M_j; record initial plurality scores on M_j under the reward-row key `none`.

For task $k = 1, \ldots, T$ **do**

  If projection is enabled and $k > 1$ **then** register the saved-subspace gradient hooks. **EndIf**

  Load and shuffle the current source pool. Construct its fixed current/replay buffer using the current mixture if replay is enabled and $k > 1$; otherwise use the current pool. Set sampling-error count to zero.

  For ordinary step $s = 1, \ldots, 64$ **do**

    Select eight prompts by cycling through the buffer and attempt to sample four completions per prompt.

    If sampling raises a caught runtime error **then** increment its counter; **break** this step loop if the count exceeds eight, otherwise **continue** to the next step. **EndIf**

    Compute binary correctness and repeated prompt-level agreement; attempt Algorithm 3 with feature collection enabled.

    If the update raises a caught runtime error **then continue** to the next step. **EndIf**

    Log returned diagnostics and add returned completion features with their correctness labels to the global probe buffer.

    If $s \in \{32, 64\}$ **then**

      Request the configured background CPU probe fit.

      If $k > 1$ **then**

        Set $F \leftarrow 0$.

        For each earlier task $j < k$, in acquisition order, **do**: execute Algorithm 4, including any targeted replay after that task's measurement; accumulate $F \leftarrow F + f_j$. **EndFor**

        If the controller is enabled **then** execute Algorithm 5 with $f_t = F/(k-1)$. **EndIf**

      EndIf

    EndIf

  EndFor

  Remove and update enabled projection hooks/bases; attempt the enabled importance estimate, storing the accumulator and anchor even after a caught failure; add the prescribed buffer prefix to replay memory, in that order.

  Join a background probe fit for up to 30 seconds if present; refit synchronously and freeze a copy for task k.

  Evaluate on V_k; store bare-task acquisition entries and baseline b_k; update the agreement prior if consistency is enabled, and log agreement AUC when both correctness classes are present. Mark task k seen.

EndFor

Save matrices and diagnostics; reconstruct covered contrasts with Algorithm 6. No model checkpoint is saved by this worker.

Algorithm 3 Microbatched group-relative policy update

Input: sampled sequences, old generation scores, binary rewards, optional detached agreement, consistency mask, retention state.

1: If consistency regularization is enabled and agreement is supplied, add the same $\alpha_C(c_i - c_0)$ to every reward in prompt group i.

2: Compute standardized, clipped group advantages using Equation 10. Zero the policy optimizer's gradients.

3: Split the sampled completions into four equal accumulation chunks. For each chunk:

   a. Forward the sampled tokens through the current policy; obtain generated-token log probabilities and final-layer representations.

   b. Compute sequence-mean scores, clipped score differences d, ratios ρ, and $\mathcal{L}_{\mathrm{pg}}$ from Equation 6.

   c. Add $\beta \,\mathrm{mean}(d^2)$. If consistency and agreement are supplied, add the masked penalties in Equation 7; agreement enters detached.

   d. On the first chunk only, add the enabled importance penalty. Divide the chunk loss by four and backpropagate.

   e. Retain last non-padding generated-token features and correctness labels for the probe buffer.

4: Apply enabled projection hooks to LoRA-B gradients; clip the model-gradient norm to 1; take one AdamW step over trainable model parameters only.

5: Return collected probe features and labels. Controller scalars are updated separately by Algorithm 5.

Algorithm 4 Monitoring one earlier task and attempting targeted replay

Input: current task k, step s, earlier task j, frozen probe P_j, baseline b_j, monitoring panel M_j, replay memory.

1: Generate four completions for every prompt in M_j. Compute plurality correctness R_j, mean agreement C_j, and mean frozen-probe probability P_j^{mean} using a plurality-representative completion.

2: Store these scores under the key `task_k_s{s}` in task j's row and append the monitoring diagnostics.

3: Store $f_j = \max(0, b_j - C_j)$ as this task's contribution to the controller input; return it after the intervention branch. Notice that this uses agreement even when b_j came from a probe.

4: Define $S_j = P_j^{\mathrm{mean}}$, or $S_j = C_j$ if the probe result is unavailable.

5: If the full intervention condition is enabled and $b_j - S_j > 0.17$:

   a. Draw up to eight buffered prompts associated with task j.

   b. Attempt two additional group-relative updates, retaining the enabled importance term. The call omits agreement, so the consistency path is skipped.

   c. Catch update exceptions as in the implementation and append one trigger event. This event does not certify that both updates succeeded.

6: Continue to the next earlier task using the potentially updated policy. Consequently, a monitoring pass need not describe one frozen checkpoint.

Algorithm 5 Scalar target-matching controller

Input: averaged contribution f_t from Algorithm 4; raw scalar parameters.

1: Zero scalar gradients, discarding any accumulated gradients from policy-loss evaluation.

2: Transform raw parameters to positive consistency coefficients and a replay fraction in $(0, 1)$.

3: Compute $a_t^\star = f_t \operatorname{softplus}(u)$ and $m_t^\star = \sigma(f_t \operatorname{softplus}(v) + o)$.

4: Form Equation 19 using the unmasked sum of all three consistency coefficients.

5: Differentiate through the targets and transformed scalars, take one Adam step with learning rate 0.005, and log the scalar objective.

6: Use updated consistency coefficients in subsequent policy updates. Use the updated replay fraction when constructing the next task buffer.

Algorithm 6 Reconstructing historical contrasts and matched initial-to-final changes

Input: deduplicated run record, task order, reward/probe/agreement matrices.
Output: per-task terms, metric-specific coverage, selected endpoint keys, and seed summaries.

Let q be the final task in the recorded order. Initialize empty collections of historical contrasts, initial-to-final changes, and selected endpoints.

For each earlier task $j \neq q$ **do**

In each score row, collect finite final-task monitoring keys of the form `q_s{step}`; parse the numerical suffix and select the largest available step.

If a selected endpoint and the row's own-task acquisition entry are finite **then** store their difference as the historical task contrast. Otherwise mark that metric's task term unavailable. **EndIf**

In the reward row, **if** the selected endpoint and initial entry `none` are finite **then** store their difference as the task's initial-to-final change, along with both scores. Otherwise mark that change unavailable. **EndIf**

Retain selected keys and recorded intervention timing; annotate whether intervening updates could mix policy states within the final pass. Do not replace any unavailable term with zero.

EndFor

After the task loop, average each collection over its valid task terms and report its coverage out of $T-1$. Call the reward acquisition contrast $\Delta_{\log}$ and the corresponding probe/agreement contrasts Δ_P, Δ_C; these compare 64-prompt acquisition panels with 16-prompt monitoring panels.

For primary and component reward comparisons, average the paired initial and final scores over the covered earlier tasks and report Δ_{init} from Equation 4. All 42 primary and component records cover four tasks. The initial and final entries use the stored 16-prompt training-prefix panels. Retain the intervention-timing annotation: some component endpoints can combine policy states. This statistic measures net learning from before the sequence, not acquisition-relative BWT.

If all required terms are finite, compute the secondary endpoint aggregate from the four earlier-task endpoints and the final task's own-task score. Retain its unequal-panel and possible mixed-policy qualifications.

Aggregate only comparable launch families, models, orders, and conditions; report every seed and name the spread statistic. All 63 historical reconstructions have four valid terms with step-64 endpoints. FWT, lag, and independent intransigence remain unavailable.

Algorithm 7 Effective-objective reanalysis of the consistency ablation

Input: the 21 component records; the class map of Corollary 1.
Output: class summaries, the pooled difference, its interval, its exact permutation p-value, and the phase decomposition.

1: For each record, compute run-level Δ_{init} with Algorithm 6; each run contributes one value.
2: Assign each run to its effective objective from its consistency mask; report the mean, sample SD, and range of each class.
3: Label a run $z = 1$ if its objective contains entropy targeting and $z = 0$ otherwise; compute $\hat{\delta} = \bar{\Delta}_{z=1} - \bar{\Delta}_{z=0}$.
4: Exact test: enumerate all $\binom{21}{12} = 293{,}930$ assignments of twelve $z = 1$ labels and report the fraction whose absolute difference is at least $|\hat{\delta}|$.
5: Blocked test: form six blocks by seed label and alignment status, average the six within-block differences, and enumerate all $18^3 = 5{,}832$ assignments that preserve the number of entropy labels in each block.
6: Interval: draw 20,000 bootstrap samples (NumPy generator seeded at 0) that resample runs with replacement within each group, and report the 2.5 and 97.5 percentiles of $\hat{\delta}$.
7: Stratified and per-task checks: repeat Step 3 within each seed label, and repeat Steps 3 and 4 for each task's change separately.
8: Robustness: repeat Steps 3 and 4 with $\Delta_{\log}$ in place of Δ_{init}, and again after removing every run with an intervention during its final monitoring pass.
9: Phase decomposition: for the three earliest tasks, split the change into the part before the final task (step-64 monitor of the penultimate task minus initial), the first half of the final task (its step-32 monitor minus the previous value), and the second half (step 64 minus step 32); repeat Steps 3 and 4 for each part.
10: Description: regress Δ_{init} on entropy, alignment, and their product by least squares, and report coefficients with conventional t-test p-values as descriptive statistics only.

D DATA, OPTIMIZATION, AND EVALUATION DETAILS

D.1 TASK CONSTRUCTION AND ANSWER EVALUATION

The configured tasks span arithmetic and multiple-choice reasoning. GSM8K and SVAMP use numerical answer checking; ARC-Challenge, CommonsenseQA, and LogiQA2.0 use answer-letter checking. The LogiQA loader reads the English-filtered JSON-text variant, excluding examples with more than 10% non-ASCII characters and mapping its zero-indexed answer to a choice letter. These loader choices are part of the experiment, not interchangeable references to every version of those benchmarks.

Each loader shuffles deterministically with seed 42 before taking a prefix. The training source pool has at most 256 prompts. The worker loads each 64-prompt panel from the training split and stores its first 16 prompts for initial and monitoring evaluations; monitoring does not make a separate 16-prompt loader request. Thus all evaluation prompts belong to the training source pool, although replay selection means that a particular prompt need not have been optimized on in every condition. The paper therefore uses "training-prefix" rather than implying identical per-example optimizer exposure.

Every prompt ends with an instruction to reason inside `<think>` tags and to give the final answer inside `<answer>` tags, and the chat template is applied with the model's thinking mode enabled. The parser reads the `<answer>` span when present and otherwise falls back to an "answer is" pattern or the last line, so a response truncated before its answer usually scores as incorrect. Evaluation samples four responses per prompt and scores the plurality parsed answer. Ties follow the first-occurring answer returned by the counter routine. This is neither greedy accuracy nor pass@1, and agreement does not itself determine correctness. Decoding removes special tokens, and the span fallback in Appendix B handles failed delimiter detection. Parsing and retokenization can affect the relationship between training features and evaluation features.

D.2 PROBE CONSTRUCTION AND INTERPRETATION

The probe buffer retains up to 4,000 completion features and binary correctness labels across tasks. Features are final-layer representations at the last non-padding generated token from the training forward pass. Every 32 steps, logistic regression with $C = 1$, the LBFGS solver, and at most 300 iterations is fitted on a shuffled 80/20 completion-level partition using random seed 0. Task-end fitting is synchronous, and a frozen copy is assigned to the task. Its training buffer can include earlier tasks, so the frozen classifier is not necessarily trained only on the task named by its evaluation row.

Completions from the same prompt, repeated visits to a prompt, and neighboring policy states can occur on both sides of this split. Reported probe AUC therefore measures internal completion-split discrimination, not prompt-disjoint or unseen-task generalization (Hewitt & Liang, 2019). At evaluation, the probe scores a representative completion voting for the plurality answer after prompt/response retokenization. This differs from training on every individual completion and introduces another distribution change.

Agreement AUC uses prompt-level plurality agreement to predict plurality correctness on own-task evaluation panels. The reported probe AUC instead averages the frozen classifiers' internal holdout AUCs. Since both labels and evaluation units differ, their bars in Figure 3 are descriptive diagnostics, not a controlled monitor-selection experiment. The primary probe AUCs span approximately 0.803–0.834, while agreement AUCs span 0.522–0.620. Neither result establishes prediction of a future accuracy drop.

D.3 COMPUTATION AND STOCHASTICITY

A run nominally performs 320 ordinary policy updates and samples 10,240 ordinary training completions, before importance-estimation generations and attempted intervention updates. Failed importance attempts consume additional work despite contributing no recorded ordinary penalty. The launch scripts permit concurrent workers and asynchronous CPU probe fitting, and the configuration enables cuDNN autotuning and TF32 without requesting deterministic algorithms; deterministic libraries alone therefore would not establish identical end-to-end execution. The record does not

Table 5: Implemented defaults for the primary experiments. Effective timing and transformations are specified where the configuration name alone would be misleading.

Setting	Value or behavior
Primary / additional models	Qwen3-8B / Qwen3-14B; Llama-3.1-8B-Instruct
Ordinary updates	64 scheduled per task; 319/317 logged in two runs, 320 in 61
Prompt batch / completions	8 prompts; 4 completions per prompt
Gradient accumulation	4 chunks; model norm clipping at 1
LoRA	Rank 16, scale 32, dropout 0.05; attention and MLP projections
Policy optimizer	AdamW, 5×10^{-6}; betas $(0.9, 0.999)$; epsilon 10^{-8}; weight decay 0.01
Precision / hardware / attention	BF16; NVIDIA H200; requested FlashAttention falls back to SDPA
Prompt / completion cap	512 / 512 tokens
Generation format	Chat template with thinking mode enabled; `<think>` and `<answer>` tags requested
Sampling	Temperature 0.7, top-p 0.9, top-k 50
Current source pool	Up to 256 prompts per task
Acquisition / monitoring panel	64 / 16 prompts, nested training-split prefixes
Evaluation generations	4 per prompt; plurality answer
Replay / importance estimation	Memory limit 128 per task key; 24 importance prompts
Projection control	Up to 8 singular directions per task and LoRA-B matrix
Probe fitting	Every 32 steps; synchronous refit at task end
Controller calls	After earlier-task monitoring at steps 32 and 64
Controller optimizer	Adam, learning rate 0.005
Intervention	Signal drop > 0.17; attempt 2 extra updates
Initial consistency coefficients	Each $\mathrm{softplus}(0) = \log 2$
Initial replay fraction	$\sigma(0) = 0.5$
Initial importance coefficient	$\mathrm{softplus}(-1) \simeq 0.3133$
Clip width / score penalty	$\mathrm{softplus}(-2.5) \simeq 0.0789$ / $\mathrm{softplus}(-2) \simeq 0.1269$

justify a precise GPU-hour or energy claim. Future comparisons should preserve successful update counts, sampled tokens, device-allocation time, and evaluation cost separately rather than equating a trigger count with completed computation.

E MATCHED-PANEL CHANGES RECOVERABLE FROM THE RECORD

This section reports descriptive statistics on the matched panels. The initial evaluation and every later monitoring call use the same stored list M_j (Appendix D); because the archive keeps no example identifiers, this identity rests on the supplied execution path rather than a saved manifest, and generations are sampled afresh at each evaluation.

For the primary order, define $I_s = \frac{1}{4}\sum_{j=1}^{4} \widehat{A}_j(\theta_{0,s}; M_j)$ and $F_s = \frac{1}{4}\sum_{j=1}^{4} \widehat{A}_j(\theta_{T,s}; M_j)$. Their difference is $\Delta_{\mathrm{init}\,s} = F_s - I_s$ from Equation 4. The observation unit for the reported sample SD is the training run ($n = 3$), and all entries are accuracy fractions.

Table 6: All primary seeds on the matched 16-prompt panels. I_s and F_s average the same four task panels before the sequence and at final monitoring.

Condition	Seed	Initial I_s	Final F_s	$\Delta_{\mathrm{init}\,s}$
Group-relative	0	0.32812	0.14062	−0.18750
Group-relative	1	0.34375	0.25000	−0.09375
Group-relative	2	0.34375	0.17188	−0.17188
Nominal importance	0	0.32812	0.20312	−0.12500
Nominal importance	1	0.34375	0.20312	−0.14062
Nominal importance	2	0.31250	0.21875	−0.09375

Condition	Seed	Initial I_s	Final F_s	$\Delta_{\text{init},s}$
Replay	0	0.29688	0.20312	−0.09375
Replay	1	0.28125	0.14062	−0.14062
Replay	2	0.31250	0.17188	−0.14062
Projection control	0	0.28125	0.20312	−0.07812
Projection control	1	0.29688	0.18750	−0.10938
Projection control	2	0.31250	0.23438	−0.07812
Nominal imp. + replay	0	0.28125	0.14062	−0.14062
Nominal imp. + replay	1	0.32812	0.12500	−0.20312
Nominal imp. + replay	2	0.32812	0.20312	−0.12500
No intervention	0	0.25000	0.42188	+0.17188
No intervention	1	0.28125	0.48438	+0.20312
No intervention	2	0.35938	0.25000	−0.10938
Full	0	0.34375	0.40625	+0.06250
Full	1	0.31250	0.54688	+0.23438
Full	2	0.29688	0.39062	+0.09375

E.1 Taskwise net change and acquisition-relative contrast

Table 7 separates the matched-panel change from the historical acquisition-relative contrast. In particular, the full variant finishes LogiQA above its initial monitoring score, while its final monitoring score lies below its 64-prompt acquisition score. These are different reference checkpoints and, for the latter quantity, different panels. Neither comparison may be substituted for the other.

Table 7: Taskwise changes across three primary seeds, mean ± sample SD. The first two columns use the same 16-prompt panels at initialization and final monitoring. The last column is the full variant's historical final16-minus-acquisition64 contrast and retains that panel discrepancy.

Task	Baseline Δ_{init}	Full Δ_{init}	Full historical $\Delta_{\log}$
GSM8K	-0.1042 ± 0.0361	$+0.1667 \pm 0.1573$	$+0.1823 \pm 0.1418$
ARC	-0.3333 ± 0.0955	$+0.1042 \pm 0.1804$	$+0.1823 \pm 0.1877$
CommonsenseQA	-0.1875 ± 0.1250	$+0.1667 \pm 0.0361$	$+0.1250 \pm 0.0563$
LogiQA	$+0.0208 \pm 0.0361$	$+0.0833 \pm 0.0361$	-0.0469 ± 0.0870

E.2 Change between two later monitoring checkpoints

One further comparison uses 16 prompts at both endpoints. For task j, let $\theta_{j,s}^{\text{first}}$ be the policy at its first recorded later monitor. In every primary record this is step 32 of the next task, not the task-acquisition checkpoint. Define

$$D_s^{\text{later}} = \frac{1}{4}\sum_{j=1}^{4}\left[\widehat{A}_j(\theta_{T,s}; M_j) - \widehat{A}_j(\theta_{j,s}^{\text{first}}; M_j)\right]. \tag{20}$$

This statistic omits any change during the first 32 updates after acquisition. Its observation horizon also differs by task: the first task is followed for much longer than the fourth. It is a descriptive measure of subsequent monitoring drift, not BWT or a full-sequence retention estimate. Replay events between its endpoints are part of the recorded trajectories, not randomized treatments.

All three full-variant observations are positive, consistent with continued improvement on the monitored training prefixes; Appendix I shows that most of it arises early in the final task.

F Evaluation variability and the reasoning-budget regime

This section quantifies two properties of the evaluation that the main results depend on: how much a score varies when nothing but the decoding sample changes, and how the 512-token completion budget shapes the accuracy of each model.

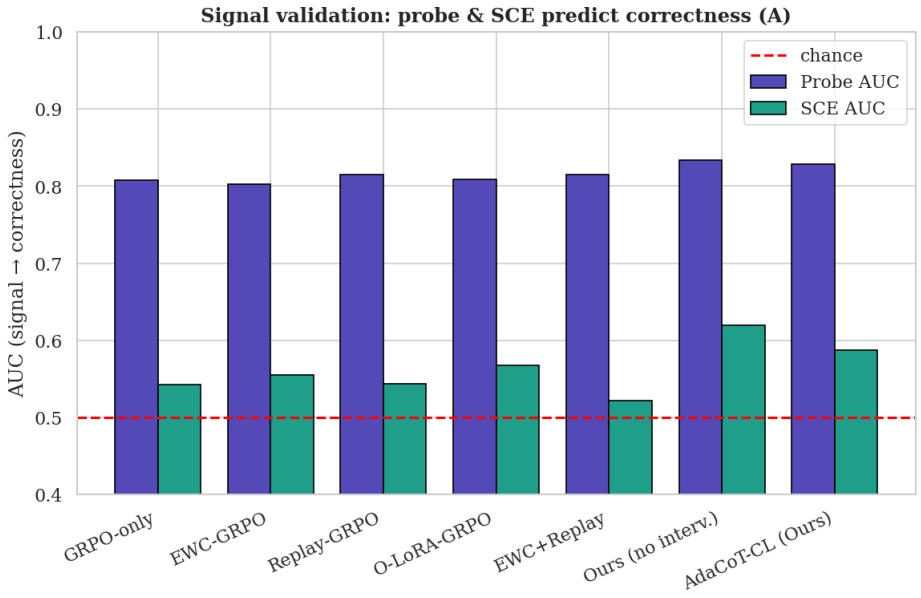


Figure 3: Original signal-discrimination plot. Probe AUC uses an internal completion-level split; SCE AUC uses agreement and plurality correctness on own-task panels. These protocols differ and share the data limitations described in Appendix D.2. The figure supports descriptive discrimination only, not a common held-out comparison or anticipatory detection.

Table 8: Subsequent drift on the same 16-prompt panels; SD is sample SD.

Condition	Seed 0	Seed 1	Seed 2	Mean ± SD
Group-relative	−0.12500	−0.03125	−0.12500	−0.0938 ± 0.0541
Nominal importance	−0.07812	−0.15625	−0.07812	−0.1042 ± 0.0451
Replay	−0.01562	−0.18750	−0.10938	−0.1042 ± 0.0861
Projection control	−0.04688	−0.14062	−0.03125	−0.0729 ± 0.0592
Nominal imp. + replay	−0.09375	−0.18750	−0.07812	−0.1198 ± 0.0592
No intervention	+0.09375	+0.12500	−0.12500	+0.0312 ± 0.1362
Full	+0.10938	+0.10938	+0.01562	+0.0781 ± 0.0541

Variability of initial evaluations. Every Qwen3-8B run begins by evaluating the same untrained policy on the same stored panels, because each loader takes a seed-42 prefix of its split and LoRA initialization leaves the policy unchanged. Table 9 summarizes these 59 initial evaluations. Their spread comes from sampling four completions per prompt at temperature 0.7 on 16 prompts, and from GPU execution that the configuration does not make deterministic (Appendices D and J). We did not detect a difference in mean initial scores across the seven primary conditions (one-way ANOVA, $p = 0.40$). Writing $\sigma = 0.033$ for the standard deviation of a four-task evaluation, a run-level Δ_{init} computed from two independent evaluations has noise standard deviation $\sqrt{2}\,\sigma \approx 0.046$. This figure describes the untrained policy only. Evaluation variability at the trained policies can differ, since it depends on each policy's per-prompt success probabilities, and it excludes the variability that training itself introduces; the figure is therefore a descriptive reference, not a significance threshold, and all inference uses between-run permutation tests.

Table 9: Initial accuracy of the untrained Qwen3-8B policy across all 59 Qwen3-8B runs (primary, component, coefficient, and order families). Each entry is a plurality-of-four accuracy on a 16-prompt training-prefix panel.

Panel	Mean	SD	Min	Max
GSM8K	0.2553	0.0584	0.1250	0.3750
ARC-Challenge	0.4968	0.0791	0.3125	0.6875
CommonsenseQA	0.4311	0.0834	0.2500	0.6250
LogiQA2.0	0.0794	0.0567	0.0000	0.1875
SVAMP	0.5466	0.0784	0.3750	0.8125
Four earlier tasks (mean)	0.3157	0.0328	0.2500	0.3906

Reasoning budget. Table 10 compares initial accuracy across the three models on the same panels, and Section 4.1 discusses what the pattern implies. Qwen3-14B, which the main text does not discuss, lies between Qwen3-8B and Llama-3.1-8B-Instruct on every task.

Table 10: Initial accuracy by model on the same 16-prompt panels (single runs; the two entries per model come from the two cross-model conditions, which share the untrained policy).

Model	GSM8K	ARC	CSQA	LogiQA	SVAMP
Qwen3-8B (mean, 59 runs)	0.255	0.497	0.431	0.079	0.547
Qwen3-14B	0.438 / 0.500	0.750 / 0.750	0.625 / 0.813	0.313 / 0.250	0.750 / 0.750
Llama-3.1-8B-Instruct	1.000 / 1.000	0.875 / 0.875	0.938 / 0.938	0.625 / 0.750	0.938 / 0.938

G EFFECTIVE-OBJECTIVE REANALYSIS

This section states and proves the corollary behind the effective-objective classes, reports the results of Algorithm 7, and gives the robustness checks cited in Section 4.3.

Corollary 1 (Effective objectives)**.** *At the same policy state, with the same sampled batch and auxiliary state, toggling the agreement mask m_C leaves the policy update unchanged in exact arithmetic. The seven consistency masks therefore implement four effective objectives: no active auxiliary (agreement only), alignment (alignment only; without entropy), entropy (entropy only; without alignment), and entropy plus alignment (without agreement; all three).*

Proof. The mask m_C multiplies $\alpha_C \mathcal{L}_C$ only. By Proposition 1, $\nabla_\theta \mathcal{L}_C = 0$, and the agreement reward shift cancels in the advantage whether or not m_C is set. At fixed controller state and feedback input, the controller update is also mask-independent, because Equation 19 sums $\alpha_B + \alpha_A + \alpha_C$ irrespective of the masks, and targeted replay updates omit agreement entirely. Two runs that differ only in m_C therefore apply the same update rule. □

Across the 21 archived component runs, the eight logged controller losses and the final scalar settings also match exactly, as the conditional argument predicts when the controller input is zero. The corollary makes such runs implementations of one objective, not independent observations: they still

differ through sampling and non-deterministic execution (Appendix J), and runs with the same seed label start from the same generator state, although their later data order can diverge once intervention branches consume different random draws. The tests below therefore include permutations that preserve seed.

Table 11: Robustness of the pooled comparison. Differences are entropy-on minus entropy-off means of run-level statistics, except in the blocked row, which averages the six seed-by-alignment block differences; p-values are exact two-sided permutation tests over the permitted relabelings for each analysis, preserving seed and alignment in the blocked row.

Analysis	Runs on / off	Mean on	Mean off	Difference	Exact p
Δ_{init}, all runs	12 / 9	$+0.132$	-0.087	$+0.218$	8.8×10^{-5}
$\Delta_{\log}$, all runs	12 / 9	$+0.124$	-0.076	$+0.200$	6.5×10^{-5}
Δ_{init}, blocked by seed and alignment	12 / 9	–	–	$+0.241$	1.7×10^{-4}
Δ_{init}, no final-pass interventions	11 / 2	$+0.136$	-0.047	$+0.183$	0.051
Δ_{init}, seed 0 only	4 / 3	$+0.047$	-0.099	$+0.146$	–
Δ_{init}, seed 1 only	4 / 3	$+0.156$	-0.089	$+0.245$	–
Δ_{init}, seed 2 only	4 / 3	$+0.191$	-0.073	$+0.264$	–

Table 12 repeats the comparison for each task. Entropy targeting has the larger mean change on all four tasks, and all four exact p-values remain below 0.05 after Holm correction for four tests. The largest differences occur on the two multiple-choice tasks, ARC and CommonsenseQA, where runs without entropy targeting decline most.

Table 12: Per-task matched-panel change Δ_{init} for the pooled comparison (12 runs with entropy targeting, 9 without).

Task	Mean on	Mean off	Difference	Exact p
GSM8K	$+0.146$	$+0.021$	$+0.125$	0.037
ARC	$+0.042$	-0.285	$+0.326$	6.6×10^{-4}
CommonsenseQA	$+0.188$	-0.104	$+0.292$	0.002
LogiQA	$+0.151$	$+0.021$	$+0.130$	0.007

The two-factor description in Table 13 uses the class without an active auxiliary as its reference. The observed entropy-on minus entropy-off contrast is $+0.315$ without alignment and $+0.167$ with alignment. The estimated interaction is negative, but these 21 runs do not establish that alignment is beneficial in only one entropy setting. The regression summarizes the recorded associations, and the permutation probabilities are retrospective and conditional on the stated exchangeability assumptions.

Table 13: Least-squares description of run-level Δ_{init} ($n = 21$). Coefficients are relative to the agreement-only class.

Term	Estimate	SE	p
Intercept (no active auxiliary)	-0.177	0.053	0.004
Entropy targeting	$+0.315$	0.065	1.5×10^{-4}
Alignment	$+0.135$	0.065	0.052
Entropy $\times$ alignment	-0.148	0.084	0.094

Blocked permutation test. The unrestricted test treats all 21 run labels as exchangeable, but runs with the same seed label share generator initialization, and the two groups contain different numbers of runs with alignment. As a sensitivity analysis, we form six blocks defined by seed and alignment status, compute the entropy-on minus entropy-off difference within each block, and average the six differences with equal weight. The observed contrast is $+0.241$. Enumerating all 5,832 assignments that preserve the number of entropy labels within each block, only the observed one reaches this absolute value, so $p = 1/5{,}832 \approx 1.7 \times 10^{-4}$ (Algorithm 7, Step 5). This analysis remains conditional on exchangeability within blocks.

The primary full series, which uses the same effective objective as the entropy-plus-alignment class, has mean $\Delta_{\text{init}} = +0.130$ against the class mean of $+0.125$, so the separately launched primary runs match the pooled class.

H Optimization diagnostics

The worker logs record, for every ordinary update, the entropy-targeting loss $(H_\theta - H_\star)^2$ averaged over the batch and the score-gap penalty $\mathbb{E}[d^2]$ from Equation 7. Neither records a sign, so they bound the size of the entropy gap and of the sampling-to-update score mismatch without fixing their direction. Table 14 reports task means.

Table 14: Logged optimization diagnostics, task means over all updates. Full runs pool the three full and three no-intervention primary runs ($n = 6$); the baseline is the group-relative condition ($n = 3$).

Task	$(H_\theta - H_\star)^2$, full	$\mathbb{E}[d^2]$, full	$\mathbb{E}[d^2]$, baseline
GSM8K	1.454	2.386	0.745
ARC	0.702	2.354	0.135
CommonsenseQA	0.481	1.195	0.021
LogiQA	0.454	1.044	0.027
SVAMP	0.315	0.529	0.006

Three observations follow, each limited by what the logs record. First, the mean squared entropy deviation falls across tasks, consistent with closer agreement with the target on the encountered inputs; because no sign is logged, it does not show from which side entropy approached the target. Second, the score-gap second moment is several times larger with entropy targeting (ablation means 1.49 with it and 0.23 without it). The sampling score is computed after temperature scaling and top-k/top-p truncation, whereas the update score uses the raw distribution, so a flatter raw distribution is one way to enlarge the gap, but other changes in the shape of the distribution can do the same. Third, the root-mean-square gap near 1.2 is large compared with the clip width $\epsilon \simeq 0.079$, so sequences outside the clip interval, for which the minimum in Equation 6 passes gradient for only one sign of advantage, may be common. Clipping frequencies were not logged, however, and a second moment alone does not describe a typical sequence. The record therefore cannot separate an effect of entropy targeting on the policy's entropy from an effect on the regime of the implemented policy term.

The cross-model runs follow the same entropy trajectory with different outcomes. The logged squared deviation falls from 3.22 to 0.18 across tasks for Qwen3-14B and from 1.24 to 0.11 for Llama-3.1-8B-Instruct, yet the first improves ($\Delta_{\text{init}} = +0.219$) and the second collapses ($\Delta_{\text{init}} = -0.766$). A falling deviation is therefore not sufficient for benefit; where the entropy term is applied, and from which starting regime, may also matter (Section 4.5).

I When the separation arises

The monitoring record gives the accuracy of every earlier task at steps 32 and 64 of each later task, always on the same 16-prompt panel, which locates when the entropy-targeted and untargeted runs diverge. In the primary order, the three earliest tasks (GSM8K, ARC, and CommonsenseQA) are monitored at the end of the LogiQA phase and at both SVAMP checkpoints, so their change splits into a part before SVAMP and the two halves of SVAMP (Algorithm 7, Step 9). Table 15 gives this decomposition for the pooled groups and four primary series, and Figure 2b plots every run. In level terms, the entropy-targeted groups climb from initial accuracies of 0.37 to 0.40 to between 0.60 and 0.65 at SVAMP step 32, while the other groups stay flat.

Buffer composition. From the second task on, each task buffer holds 128 to 130 current prompts followed by 126 to 128 replayed prompts from earlier tasks, and updates cycle through it in order. Updates 1 to 16 and 33 to 48 therefore use current-task prompts, and updates 17 to 32 and 49 to 64 use replayed prompts, apart from one or two current prompts at the boundary. The first 32 updates of every phase contain the same mixture, which lets phases be compared with one another.

Table 15: Phase decomposition of the change in mean accuracy on the GSM8K, ARC, and CommonsenseQA panels. SVAMP updates 1–32 interleave SVAMP and replayed earlier prompts; p values are exact permutation tests.

Group	Runs	Before SVAMP	SVAMP 1–32	SVAMP 33–64	Total
Entropy on (ablation)	12	-0.030	$+0.269$	-0.115	$+0.125$
Entropy off (ablation)	9	-0.086	-0.007	-0.030	-0.123
Difference		$+0.056$	$+0.276$	-0.084	$+0.248$
Exact p		0.14	3.4×10^{-6}	0.07	
Primary full	3	-0.042	$+0.243$	-0.056	$+0.146$
Primary no intervention	3	-0.007	$+0.292$	-0.229	$+0.056$
Group-relative baseline	3	-0.201	-0.007	$+0.000$	-0.208
Replay baseline	3	-0.090	-0.042	-0.014	-0.146

Robustness to interventions and replay. Some monitoring passes include intervention updates between task evaluations. Excluding every run with an intervention during either endpoint pass (the step-64 LogiQA pass or the step-32 SVAMP pass) leaves nine entropy-targeted runs with a mean rise of 0.2755 and five others with a mean change of -0.0292, so the separation persists when both endpoint passes contain no intervention updates. Replay blocks of the same size in earlier phases do not produce a comparable rise: in the entropy-targeted runs, the first halves of the CommonsenseQA and LogiQA phases change the earlier panels by -0.021 and -0.026. Because those phases also come earlier in the sequence, this comparison does not separate replay from position. The two checkpoints of the SVAMP block cannot show whether the rise within it is sudden or gradual.

Separation and generality. Beyond the 12-versus-9 comparison of Section 4.4, the rise separates all 42 primary and component runs: the 18 entropy-targeted runs rise by 0.28 more than the 24 others (approximate permutation $p < 10^{-5}$ from 200,000 relabelings), and 16 of the 18 exceed the largest rise among the 24. Without entropy targeting, the per-task rises are $+0.000$ on GSM8K, $+0.028$ on ARC, and -0.049 on CommonsenseQA, against the $+0.31$, $+0.31$, and $+0.19$ reported in Section 4.4. The size of the rise tracks the final outcome: across the 21 component runs, it correlates with Δ_{init} at $r = 0.68$.

Phase and position sensitivity. Table 16 gives the first-half change for every phase that has matched earlier panels, in all five orders (full configuration). Four of the six arithmetic phases (GSM8K or SVAMP) produce a rise of at least 0.15; none of the nine multiple-choice phases does, and the largest among them is $+0.104$. All four large rises occur in the fifth task, however, while the two earlier arithmetic phases yield $+0.0625$ and -0.0833, and the only multiple-choice phase in fifth position yields $+0.104$. Task content and sequence position are therefore not independently varied in this comparison, and the arithmetic reading of Section 4.4 is one hypothesis among several.

Table 16: First-half change on panels monitored at both the end of the preceding phase and step 32 of the current phase (full configuration, three seeds per order). The third, fourth, and fifth phases use one, two, and three earlier-task panels; the second phase lacks a matched pre-phase monitoring score. G, A, C, L, S denote GSM8K, ARC-Challenge, CommonsenseQA, LogiQA2.0, and SVAMP; arithmetic phases are G and S.

Order	Third task	Fourth task	Fifth task
G–A–C–L–S	C: -0.021	L: $+0.042$	S: $+0.243$
S–L–C–A–G	C: -0.042	A: $+0.000$	G: $+0.153$
G–C–A–S–L	A: $+0.042$	S: -0.083	L: $+0.104$
A–C–S–L–G	S: $+0.062$	L: $+0.021$	G: $+0.229$
G–L–A–C–S	A: $+0.042$	C: $+0.021$	S: $+0.153$

J EXECUTION COMPARABILITY OF NOMINALLY IDENTICAL VARIANTS

The primary full condition and the all-three ablation enable the same components and, by Corollary 1, implement the same objective as the without-agreement ablation. Their recorded trajectories nevertheless differ: seed-matched differences in $\Delta_{\log}$ (primary full minus all three) are $+0.082$, $+0.137$, and -0.070. The difference is visible before any update, because the initial evaluations already differ (Table 17).

Table 17: Initial scores before the task loop, seed 0, for two launches of the same nominal configuration. Both use the same 16-prompt panels.

Task	Primary full	All-three series	Difference
GSM8K	0.2500	0.1875	+0.0625
ARC	0.6250	0.4375	+0.1875
CommonsenseQA	0.3750	0.3125	+0.0625
LogiQA	0.1250	0.0625	+0.0625
SVAMP	0.5000	0.5000	+0.0000

The mismatch is compatible with stochastic decoding and nondeterministic execution, although the saved records cannot isolate its cause or exclude unrecorded execution differences. The two four-task initial means, 0.344 and 0.250, both lie within the range of the 59 initial evaluations of the same untrained policy (0.250 to 0.391; Table 9), and we did not detect a dependence of initial scores on condition. A shared seed fixes the Python and PyTorch generators but not the order of GPU operations: the configuration enables cuDNN autotuning and TF32, does not request deterministic algorithms, and in the primary launch shares each device between two workers. Sampled completions can therefore diverge from the first generation step. The recorded controller losses and final scalar values do agree exactly between the two series, but only because the controller input is zero in both, so this agreement says nothing about whether the sampled trajectories matched. We accordingly treat the series as separate launches of one objective, group them only in the comparison that Corollary 1 motivates, and report them separately elsewhere. A bitwise replica would require deterministic kernels, one worker per device, and persisted initial outputs.

K COMPLETE NUMERICAL RESULTS

The following tables retain all recorded seeds for the historical contrasts, avoiding selection of the most favorable run. Matched initial-to-final results appear separately in Appendix E. Values are rounded for display; aggregate reconstruction used the saved numerical precision. The notation $\bar{A}_{\log}$ denotes an unweighted average over four earlier-task monitoring scores on 16 prompts and the final task's own-task score on 64 prompts. When replay intervenes between task checks, it can combine policy states. It is not a held-out accuracy at a guaranteed common final checkpoint.

K.1 PRIMARY COMPARISONS AND UNCERTAINTY

Table 18 provides the primary seed-level record. The original plotting routine bootstraps the seed mean with 2,000 resamples and a fresh NumPy generator seeded at 42 for each group. At three seeds, the displayed 95% percentile interval endpoints coincide with the observed extrema for these groups. These intervals quantify only variation among the three logged runs; they cannot account for panel discrepancy, train/evaluation overlap, or unrecorded execution variation.

The exported significance table does not provide valid tests for this experiment. Its Wilcoxon helper returns an unavailable value for fewer than five samples; the export replaces missing values with 1 before multiplicity adjustment. Those placeholders should not be read as computed p-values, evidence of equivalence, or evidence against an effect. We consequently omit significance stars from three-seed comparisons and base inference on the pooled permutation tests of Appendix G.

Table 18: Every primary seed. All changes are unequal-panel training-prefix contrasts; $\bar{A}_{\log}$ is the recorded endpoint aggregate.

Condition	Seed	$\Delta_{\log}$	Δ_P	Δ_C	$\bar{A}_{\log}$
Group-relative	0	−0.11719	−0.08303	+0.06055	0.16875
Group-relative	1	−0.00781	−0.01689	+0.04395	0.25000
Group-relative	2	−0.13281	−0.12583	+0.03418	0.20938
Nominal importance	0	−0.07422	−0.11308	+0.00293	0.20938
Nominal importance	1	−0.08594	−0.05459	+0.03125	0.22500
Nominal importance	2	−0.08203	−0.15044	+0.03027	0.23125
Replay	0	−0.07031	−0.10866	+0.00879	0.22812
Replay	1	−0.14453	−0.12284	+0.03906	0.16875
Replay	2	−0.17969	−0.15276	−0.00391	0.19375
Projection control	0	−0.06641	−0.02560	+0.00977	0.24063
Projection control	1	−0.10547	−0.05680	−0.00293	0.21250
Projection control	2	−0.02344	−0.10559	+0.01953	0.24688
Nominal imp. + replay	0	−0.13281	−0.07709	+0.02344	0.15937
Nominal imp. + replay	1	−0.14062	−0.15725	−0.00879	0.13125
Nominal imp. + replay	2	−0.06641	−0.01300	−0.01172	0.22187
No intervention	0	+0.09375	+0.22208	−0.03516	0.43750
No intervention	1	+0.14062	+0.18751	−0.02734	0.47813
No intervention	2	−0.06250	+0.13512	−0.04883	0.27500
Full	0	+0.10156	+0.24307	−0.03711	0.38750
Full	1	+0.19922	+0.30818	−0.05859	0.54063
Full	2	+0.03125	+0.29638	+0.00488	0.40000

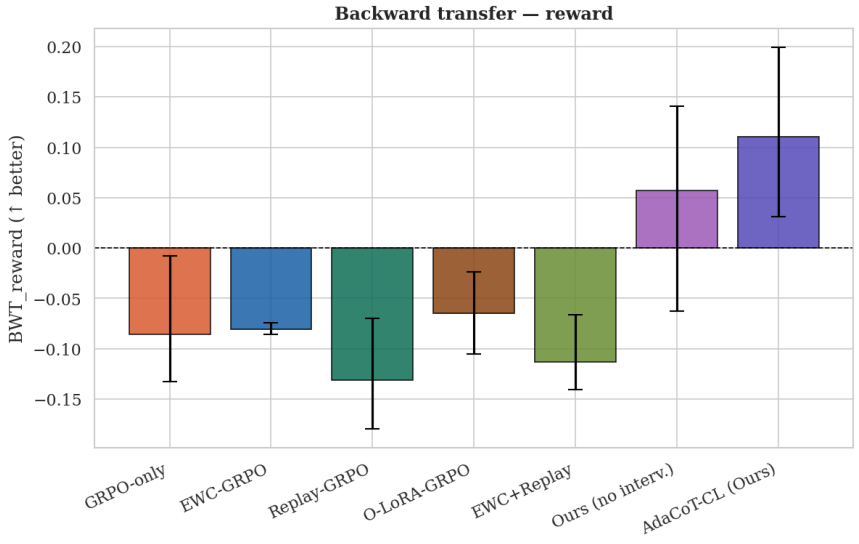


Figure 4: Primary historical contrast $\Delta_{\log}$ with the original percentile bootstrap intervals for the three-seed mean (Appendix K.1). Table 1 reports the matched change Δ_{init}.

K.2 Consistency masks

The consistency ablations retain replay, the controller, the targeted replay branch, and the nominal importance attempts, and vary only the masks in Equation 7. Figure 5 shows mean and population SD, whereas Table 19 uses sample SD; for three seeds, population SD is $\sqrt{2/3}$ times sample SD, and neither is a confidence interval.

The all-three ablation and the primary full variant are separate launches of the same effective objective; Appendix J explains why their initial scores differ, and Appendix G shows that the primary series matches the pooled class.

Table 19: Consistency ablations over three seeds, mean ± sample SD, with the effective objective of each mask (Corollary 1). Δ_{init} compares initial and final scores on the same 16-prompt training-prefix panels; $\Delta_{\log}$ is the historical unequal-panel contrast. "Only" refers to the consistency mask. A dagger marks configurations with an intervention during final monitoring in at least one seed, whose endpoint aggregate can therefore combine policy states.

Consistency configuration	Effective objective	Δ_{init}	$\Delta_{\log}$
Entropy only (BCT)	Entropy	$+0.1719 \pm 0.1491$	$+0.1641 \pm 0.1252$
Alignment only (ACT)†	Alignment	-0.0729 ± 0.0090	-0.0794 ± 0.0303
Agreement only (CoT)†	None	-0.1771 ± 0.0239	-0.1471 ± 0.0277
Without entropy†	Alignment	-0.0104 ± 0.0239	$+0.0000 \pm 0.0444$
Without alignment	Entropy	$+0.1042 \pm 0.1565$	$+0.1276 \pm 0.1232$
Without agreement†	Entropy + alignment	$+0.1719 \pm 0.0870$	$+0.1432 \pm 0.0941$
All three	Entropy + alignment	$+0.0781 \pm 0.0563$	$+0.0612 \pm 0.0410$

Table 20 reports every seed on both metrics; Δ_{init} is reconstructed with Algorithm 6, and all 21 records contain the required entries.

Table 20: Every component seed on both metrics. Δ_{init} uses the matched 16-prompt panels, $\Delta_{\log}$ is the historical contrast, and $\bar{A}_{\log}$ is the recorded endpoint aggregate (mean over seeds). Final-monitoring interventions occur in alignment-only seeds 1 and 2, agreement-only seeds 0–2, without-entropy seeds 1 and 2, and without-agreement seed 0, so those endpoints may combine policy states.

	Δ_{init}			$\Delta_{\log}$			
Configuration	Seed 0	Seed 1	Seed 2	Seed 0	Seed 1	Seed 2	$\bar{A}_{\log}$
Entropy only	$+0.03125$	$+0.15625$	$+0.32812$	$+0.04297$	$+0.15625$	$+0.29297$	0.51562
Alignment only	-0.06250	-0.07812	-0.07812	-0.07031	-0.05469	-0.11328	0.28646
Agreement only	-0.20312	-0.17188	-0.15625	-0.17188	-0.15234	-0.11719	0.18229
Without entropy	-0.03125	-0.01562	$+0.01562$	-0.03125	-0.01953	$+0.05078$	0.33333
Without alignment	-0.04688	$+0.26562$	$+0.09375$	$+0.00781$	$+0.25391$	$+0.12109$	0.47187
Without agreement	$+0.07812$	$+0.18750$	$+0.25000$	$+0.03516$	$+0.20703$	$+0.18750$	0.47500
All three	$+0.12500$	$+0.01562$	$+0.09375$	$+0.01953$	$+0.06250$	$+0.10156$	0.42708

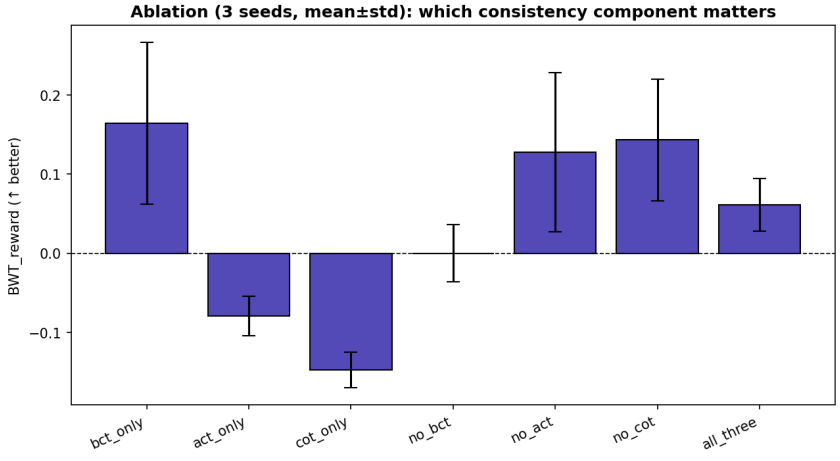


Figure 5: Historical contrast $\Delta_{\log}$ per consistency mask (three seeds, mean ± population SD). Bars that differ only in the agreement mask implement the same objective (Corollary 1). Every mask with entropy targeting is positive; every mask without it is negative or zero. Table 19 lists each mask and Table 2 gives the pooled test on Δ_{init}.

K.3 Coefficient sensitivity

Table 21 reports every fixed-weight run. The same fixed value is assigned to all three consistency coefficients, and all scalar parameters are frozen. The main enabled controller instead changes both coefficients and the replay fraction. Therefore this sweep is informative about sensitivity but does not isolate just one scalar or establish a controller comparison at equal selected hyperparameters. The 0.2 result is the largest observed value on this grid; it was not selected by an independent development protocol. Because the alignment and agreement weights move with the entropy weight, the sweep measures sensitivity to jointly changing the consistency coefficients; it is consistent with, but does not isolate, the entropy effect of the grouped analysis.

Table 21: All fixed-coefficient runs, seed 0. The same fixed value is assigned to all three consistency coefficients and every scalar is frozen. Δ_{init} uses the matched 16-prompt panels; $\Delta_{\log}$ is the historical contrast. The enabled-controller run with the same seed, whose coefficients stay near 0.67, has $\Delta_{\text{init}} = +0.0625$ and $\Delta_{\log} = +0.1016$. No training-seed uncertainty is available.

Fixed α	Δ_{init}	$\Delta_{\log}$	$\bar{A}_{\log}$
0.01	−0.17188	−0.11719	0.19687
0.05	−0.01562	+0.05078	0.29063
0.1	+0.21875	+0.22656	0.59062
0.2	+0.25000	+0.28125	0.62500
0.5	+0.10938	+0.10156	0.45312

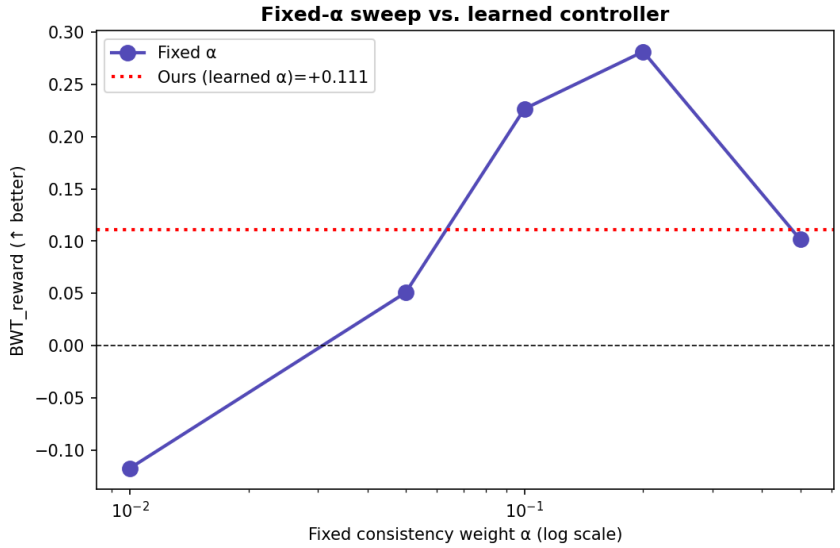


Figure 6: Single-seed fixed-coefficient sweep of Table 21; the dotted line is the primary three-seed enabled-controller mean.

K.4 Task-order sensitivity

The four alternate orders are evaluated for the full variant only. No baseline runs under these orders are available, so positive means do not establish an advantage over alternatives. In addition, Equation 2 excludes whichever task is last: changing the final task changes the set of tasks entering the mean. An order effect on this statistic therefore combines sequence effects with changes in the evaluated task set.

Matched-panel change under alternative orders. Table 23 repeats the order comparison with Δ_{init}, which uses the same panels at both endpoints. Across the 12 alternative-order runs, mean Δ_{init} is $+0.022$ and five runs are positive, compared with $+0.130$ and three of three in the primary order.

Table 22: Alternate-order contrasts. G = GSM8K, A = ARC-Challenge, C = CommonsenseQA, L = LogiQA2.0, S = SVAMP. The original order is G–A–C–L–S. Every order here uses the full variant. SD is sample SD.

Order	Seed 0	Seed 1	Seed 2	Mean ± SD	$\bar{A}_{\log}$
S–L–C–A–G	$+0.13672$	-0.09375	-0.00781	$+0.0117 \pm 0.1165$	0.35938
G–C–A–S–L	$+0.01953$	$+0.03516$	$+0.32812$	$+0.1276 \pm 0.1738$	0.48854
A–C–S–L–G	-0.08203	-0.30469	$+0.13281$	-0.0846 ± 0.2188	0.30938
G–L–A–C–S	$+0.01562$	$+0.07422$	-0.00781	$+0.0273 \pm 0.0423$	0.33854

Table 23: Matched-panel change Δ_{init} for the full variant under each order; SD is sample SD.

Order	Seed 0	Seed 1	Seed 2	Mean ± SD
G–A–C–L–S (primary)	$+0.06250$	$+0.23438$	$+0.09375$	$+0.1302 \pm 0.0916$
S–L–C–A–G	$+0.15625$	-0.07812	-0.04688	$+0.0104 \pm 0.1273$
G–C–A–S–L	$+0.00000$	$+0.06250$	$+0.39062$	$+0.1510 \pm 0.2098$
A–C–S–L–G	-0.01562	-0.25000	$+0.17188$	-0.0312 ± 0.2114
G–L–A–C–S	-0.09375	$+0.03125$	-0.06250	-0.0417 ± 0.0651

Holding the evaluated tasks fixed across orders. The unrestricted historical contrast excludes the task placed last, so changing the final task also changes which tasks enter the average. ARC and CommonsenseQA are earlier tasks in all five orders. Table 24 restricts each order to this shared pair, using the same endpoint-selection rule as Equation 2. This removes the change in task composition; it does not remove the acquisition64-versus-final16 discrepancy, differences in task age, or interventions during some final passes. The ARC-first order remains negative on average under this restriction.

Table 24: Historical contrasts restricted to the common tasks ARC and CommonsenseQA: $\frac{1}{2}\sum_{j\in\{\text{ARC,CommonsenseQA}\}}[\widehat{A}_j(\theta^{\log}_{T,j}; M_j) - \widehat{A}_j(\theta_j; V_j)]$. Entries retain unequal-panel and possible mixed-policy limitations; SD is sample SD across three seeds.

Order	Seed 0	Seed 1	Seed 2	Mean ± SD
G–A–C–L–S	$+0.17188$	$+0.25781$	$+0.03125$	$+0.1536 \pm 0.1144$
S–L–C–A–G	$+0.27344$	$+0.00000$	-0.09375	$+0.0599 \pm 0.1908$
G–C–A–S–L	$+0.10156$	$+0.16406$	$+0.33594$	$+0.2005 \pm 0.1214$
A–C–S–L–G	$+0.04688$	-0.44531	$+0.14844$	-0.0833 ± 0.3176
G–L–A–C–S	$+0.04688$	$+0.15625$	-0.11719	$+0.0286 \pm 0.1376$

The fixed-task restriction is a sensitivity analysis, not an additional independent experiment, and three seeds cannot support a reliable ranking of orders. Appendix I shows that large first-half rises occur only in the fifth phase, which confounds task content with position and does not by itself explain the variation.

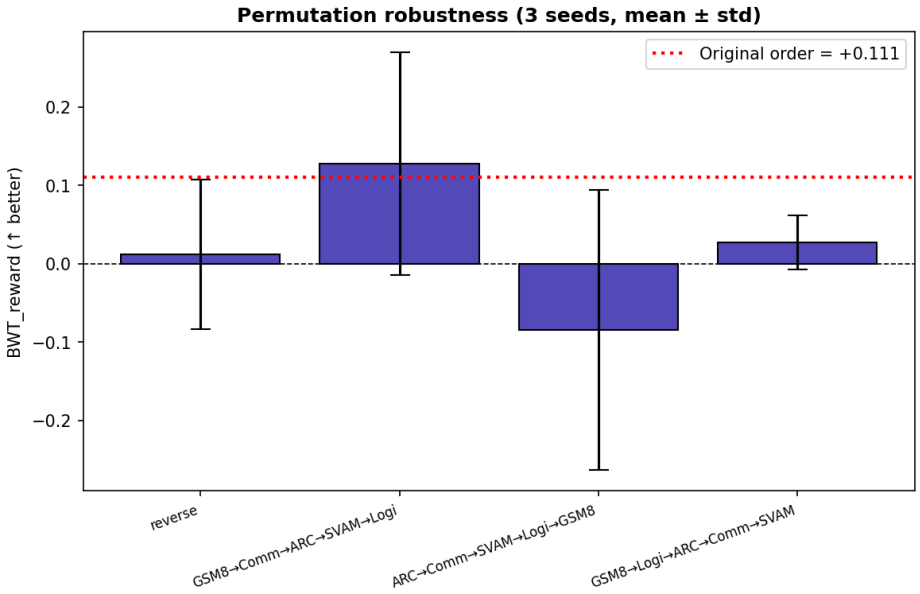


Figure 7: Alternate-order results over three seeds; error bars are population SD. The dotted line is the original-order mean. The ARC-first order has a negative mean, and variation is substantial.

K.5 Cross-model tests

The cross-model record contains one seed per condition. Architecture, tokenizer, generation behavior, and response format change together, so the logs cannot isolate the cause of the Llama failure; Section 4.5 and Appendix H give the candidate explanation and the evidence consistent with it.

The full variants on both models also have interventions during the final monitoring pass: the Qwen3-14B run triggers for GSM8K and CommonsenseQA, and the Llama run triggers for GSM8K. Thus the aggregate endpoint scores in Table 25 need not come from one common final checkpoint. The result is a warning about sensitivity of the tested pipeline, rather than a clean estimate of architecture-dependent held-out transfer.

Table 25: Cross-model runs, seed 0. The full-method final passes include intervention events.

Model	Condition	Δ_{init}	$\Delta_{\log}$	$\bar{A}_{\log}$
Qwen3-14B	Group-relative	-0.01562	$+0.01172$	0.53125
Qwen3-14B	Full	$+0.21875$	$+0.15234$	0.82500
Llama-3.1-8B-Instruct	Group-relative	-0.03125	$+0.04688$	0.84375
Llama-3.1-8B-Instruct	Full	-0.76562	-0.32422	0.13437

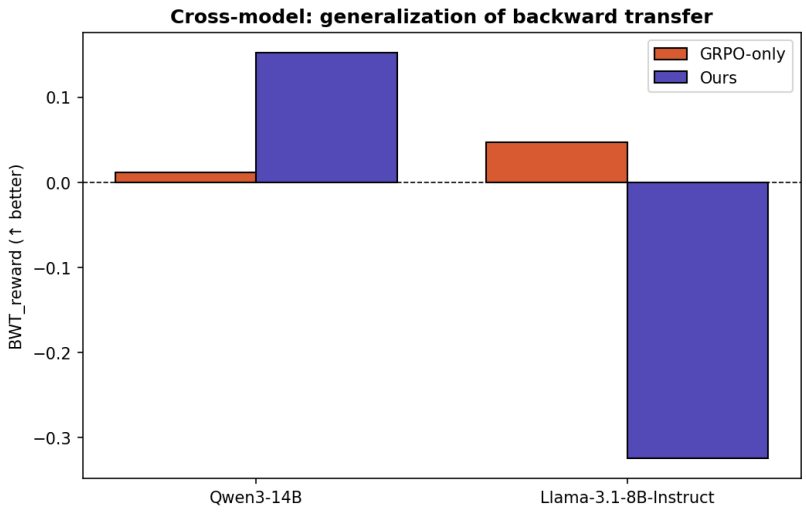


Figure 8: Single-seed cross-model contrasts. Despite the original title's reference to generalization, these are training-prefix logged contrasts, with mixed-policy endpoints possible in the full variants. The opposite directions on Qwen3-14B and Llama-3.1-8B-Instruct rule out a claim of uniform improvement in this record.

L Intervention and auxiliary-signal diagnostics

L.1 What the intervention comparison identifies

The full/no-intervention comparison changes whether a probe-threshold replay branch is enabled. It does not randomize trigger timing, match extra successful updates, or hold every stochastic execution path fixed. Table 26 makes the event evidence explicit: two triggers are recorded across the three full runs, both for ARC during LogiQA at step 64.

Table 26: Paired seed labels for the intervention variants. A trigger is recorded even when attempted replay updates raise caught exceptions; counts do not certify completed updates. The seed-2 difference occurs without a trigger in either condition.

Seed	Full $\Delta_{\log}$	No intervention $\Delta_{\log}$	Difference	Full triggers
0	$+0.1015625$	$+0.0937500$	$+0.0078125$	1
1	$+0.1992188$	$+0.1406250$	$+0.0585938$	1
2	$+0.0312500$	-0.0625000	$+0.0937500$	0
Mean	$+0.1106771$	$+0.0572917$	$+0.0533854$	$2/3$

The no-intervention seed-1 run also records only 317 ordinary updates after three sampling failures, whereas its full-condition counterpart records 320. The largest paired difference is observed in seed 2 without an intervention event. This does not explain the origin of that difference, but it does preclude interpreting the whole mean difference as an identified effect of targeted replay. A direct test would compare threshold-triggered, random, and scheduled replay with equal successful update and token budgets, using checkpoints measured before any intervention. That experiment is prospective, not contained in the present record.

L.2 Probe probability and agreement contrasts

The same matrix reconstruction can be applied to mean frozen-probe probability and mean agreement, producing Δ_P and Δ_C. The former is not an AUC change, and the latter is not an entropy change. These contrasts also inherit the 64-versus-16 panel discrepancy. In the primary full variant,

$\Delta_P = +0.2825$ while $\Delta_C = -0.0303$. The two monitors consequently should not be assumed to represent interchangeable units of forgetting.

This distinction also explains the controller-input concern in Equation 19. Comparing an acquisition probe probability with later agreement has no calibration guarantee, even if each signal separately correlates with correctness. The observed all-zero inputs are sufficient to rule out a demonstrated nontrivial feedback response in the primary runs; they do not establish that any correctly calibrated feedback controller would fail.

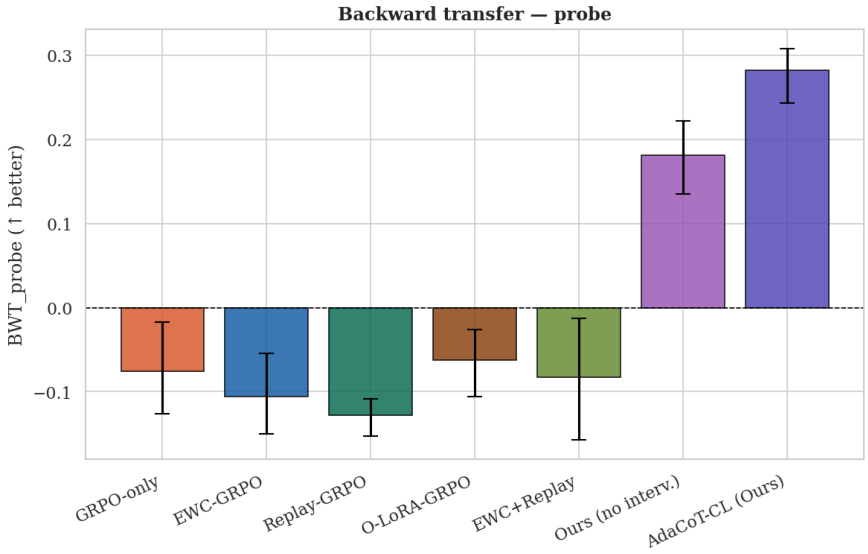


Figure 9: Recorded change in mean frozen-probe probability Δ_P, with bootstrap intervals over three seeds. Positive bars do not establish improved probe AUC, preserved reasoning mechanisms, or held-out task accuracy.

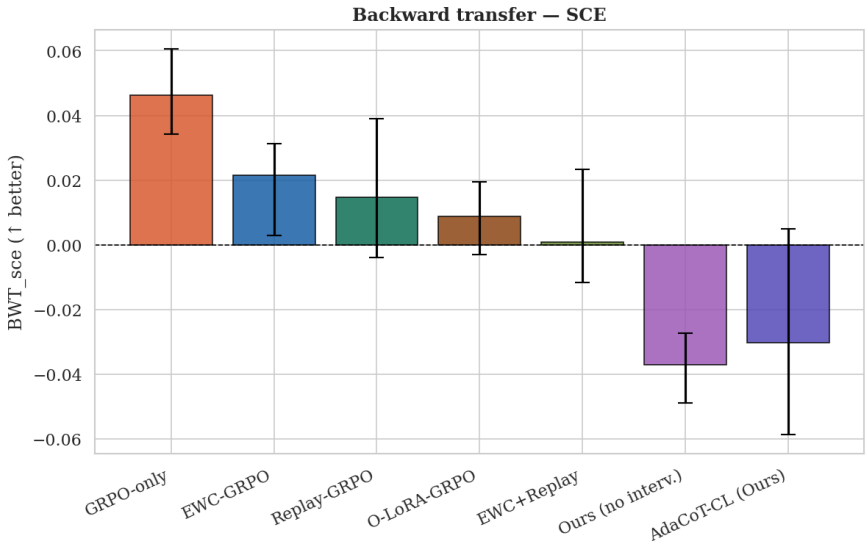


Figure 10: Recorded agreement contrast Δ_C (SCE is plurality agreement in this implementation), with bootstrap intervals. Its decrease for the two consistency variants, alongside positive reward contrasts, shows that agreement and correctness are distinct diagnostics.

M MEASUREMENT VALIDITY AND UNAVAILABLE QUANTITIES

M.1 ENDPOINT-KEY RECONSTRUCTION

Own-task evaluation writes a bare task-name key, whereas later monitoring writes keys with step suffixes. The original metric helper looks for the bare final-task key in earlier-task rows, finds no matching comparisons, and returns zero. Postprocessing instead selects the maximum numerical step suffix for the final task and recomputes reward, probe, and agreement contrasts. All 63 archived records match the latter reconstruction and have full four-task coverage.

Repairing this lookup resolves a bookkeeping problem. It does not resolve the statistical difference between acquisition and monitoring panels. Equation 3 makes the distinction exact: without the acquisition score on M_j, the saved aggregates cannot separate matched-prefix retention from within-checkpoint subset discrepancy. Because evaluations sample new completions, Monte Carlo variation is an additional source of difference even when prompts match; Appendix F measures its size. No confidence interval over three training seeds removes either issue. The unknown acquisition score on M_j can reverse the sign of the matched acquisition-relative change, so a positive historical contrast alone does not identify positive retention even on the training prefix. In contrast, Δ_{init} uses initial and final evaluations on the stored M_j panel and avoids this particular 64-versus-16 discrepancy; its reference is pre-sequence initialization, not acquisition.

M.2 A FROZEN EVALUATION PASS IS A SEPARATE REQUIREMENT

Algorithm 4 evaluates an earlier task and may update the policy before evaluating the next task. If this happens in the final pass, the endpoint vector is not an evaluation of one frozen policy. This occurs in the full cross-model runs, several component runs, the fixed-$\alpha = 0.01$ run, and five alternate-order runs. It does not occur in the main full variant's final SVAMP pass. A correct reanalysis preserves this distinction instead of applying the mixed-policy caveat indiscriminately to every run.

The endpoint aggregate $\bar{A}_{\log}$ (Appendix K) also weights tasks equally while mixing 16- and 64-prompt panels, so it neither matches the acquisition panels nor resolves intervening updates. For these reasons the main paper relies on Δ_{init} and reports $\Delta_{\log}$ and $\bar{A}_{\log}$ only as secondary logged quantities.

M.3 FORWARD TRANSFER, LAG, AND INTRANSIGENCE

A forward-transfer calculation requires evaluating a future task after relevant earlier learning but before training on that future task. Those entries are not recorded here. The original helper returns zero for an empty set of valid comparisons. Likewise, each within-transition monitoring series has only two time points, steps 32 and 64, which is insufficient for the implemented lag calculation; its empty aggregate also defaults to zero. Both quantities are therefore *unavailable*, not measured zero.

The postprocessor sets the displayed intransigence value to negative reconstructed reward contrast. Standard intransigence requires an appropriate reference for learning the current task, rather than negating a retention statistic (Chaudhry et al., 2018). No independent joint-training reference is available in the record, so the supplied forward-transfer and intransigence exports are excluded from every result claim.

N PROSPECTIVE VALIDATION PROTOCOL

This section specifies experiments needed to test the main hypothesis more decisively. None of the measurements described here is presented as completed. The proposal follows from the implemented paths and observed sensitivity, so it prioritizes identifying the entropy effect before increasing system complexity.

Matched, unseen evaluation. For each task, construct disjoint training, development, and test panels with persisted example identifiers. Save a frozen policy or adapter after every task and evaluate every checkpoint on the same test panel with a fixed decoding budget. Keep monitoring and

intervention outside that frozen evaluation pass. Report the full task-by-checkpoint accuracy matrix, matched-panel BWT, final task-averaged accuracy, and valid pre-learning entries for forward transfer. If intransigence is reported, train the appropriate reference rather than deriving it from BWT.

An isolated regularization comparison. Begin with a group-relative learner whose policy term uses a proper importance ratio, either a tokenwise clipped objective or a sequence likelihood ratio from summed log probabilities, with update scores computed under the same attention mask, decoding transformations, and mode as sampling; then add entropy targeting restricted to a detected reasoning span, with and without replay. Compare it with established entropy controls under the same budget: a fixed entropy bonus (Mnih et al., 2016), decoupled clipping (Yu et al., 2025), and covariance-based entropy control (Cui et al., 2025). Log the signed span entropy, completion length, the fraction of completions that reach the token budget, reward variance, the frequency of zero-advantage groups, and gradient norms, so that the effect of entropy targeting can be separated from changes in termination behavior and from the clipping regime described in Appendix H. Calibrate the target per model from the initial entropy of the base policy, and select targets and weights on a development split with the same tuning budget for every method. Before evaluating an importance-based baseline, verify successful estimates, nonzero stored importance, and nonzero penalty gradients, and stop the run rather than silently replacing a failed estimate with zero. Compare fixed weights, the common schedule, and a responsive controller only after each has a verified feedback path.

Testing when and why the rise occurs. One hypothesis motivated by the timing pattern (Appendix I) is that entropy targeting, arithmetic-task training, and replay jointly improve completion within the token budget; the saved records do not establish which factors are necessary. Test it by evaluating held-out panels of all earlier tasks at dense checkpoints within the first arithmetic phase, logging completion lengths and answer-tag rates, and repeating the phase with and without replay and with entropy targeting switched on only for that phase. Add an evaluation after update 16 to separate the new-task updates from the replay block, and vary the arithmetic task's position to distinguish task content from accumulated training. Repeating the phase with a budget large enough that truncation is rare then tests the budget reading: attenuation would support a budget-related explanation, and persistence would weaken a truncation-only account, but neither outcome alone would establish transfer of reasoning behavior.

Identifiable monitoring and intervention. Fit probes with prompt-disjoint partitions and test whether a score at time t predicts a subsequent accuracy loss beyond current accuracy and agreement. For replay intervention, randomize trigger selection or use paired checkpoint branches with equal successful-update and token budgets. Compare targeted, random, scheduled, and no-extra-update conditions. Distinguish branch entry, attempted updates, and completed updates in the logs. These controls would address the gap between predictive monitoring and an effective intervention policy.

Coverage and execution integrity. Repeat the selected configurations across models and task orders, including the observed Llama and ARC-first failure cases, and run baselines under every order. Report a common task set when comparing orders, alongside order-specific transfer matrices. Increase seed coverage enough to resolve the effect sizes of interest rather than interpreting three-seed intervals as definitive. Pin dependencies, disable silent synthetic fallback, request deterministic kernels with one worker per device, save data hashes and sampled example identifiers, persist adapter checkpoints and predictions, and record hardware and successful token/update counts. A fail-fast loader and a complete manifest make dataset fidelity testable rather than assumed.