DENSER ≠ BETTER: LIMITS OF ON-POLICY SELF-DISTILLATION FOR CONTINUAL POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual post-training enables foundation models to acquire new knowledge while preserving existing capabilities. Recent work suggests that on-policy learning can mitigate forgetting, with self-distillation as a particularly attractive approach. We revisit this optimistic claim through self-distillation policy optimization (SDPO). Our experiments show that SDPO accelerates in-domain specialization when teacher signals are stable and well aligned, but struggles to generalize out of distribution. In continual post-training, SDPO exhibits greater forgetting and can even collapse, whereas GRPO, the more established on-policy reinforcement learning method, adapts more conservatively and better preserves prior capabilities. Further analyses link these failures to increased drift in parameter and response space, and to amplification of high-frequency artifacts through a self-reinforcing teacher–student loop. Thus, on-policy data alone is insufficient for continual learning. Self-distillation is effective when teacher targets are stable and token-level supervision is reliable, but should not be treated as a default stabilizer for continual post-training.

1 INTRODUCTION

Reinforcement learning with verifiable rewards has emerged as an effective post-training paradigm for large language models (LLMs) (Lambert et al., 2024; Guo et al., 2025a). However, practical systems often require continual post-training across domains and skills (Team, 2025; NVIDIA, 2025), raising a central challenge: models must acquire new capabilities without forgetting prior ones. Existing work suggests an optimistic picture for continual post-training: on-policy reinforcement fine-tuning (RFT) forgets less than supervised fine-tuning (SFT) (Lai et al., 2025; Chen et al., 2025; Shenfeld et al., 2025; Zhang et al., 2025b), and this benefit has often been attributed to the use of on-policy data (Chen et al., 2025; Shenfeld et al., 2025; Zhu et al., 2025).

Recently, on-policy distillation has been claimed to be an attractive alternative to reward-based reinforcement learning. By converting model-generated rollouts into dense token-level supervision, these methods promise better credit assignment, higher sample efficiency, and reduced dependence on external teachers or explicit reward models (Agarwal et al., 2024; Gu et al., 2024; Thinking Machines Lab, 2025). A particularly appealing variant is on-policy self-distillation, where the model itself acts as a teacher by conditioning on demonstrations, feedback, solutions, or other auxiliary context, and the resulting behavior is distilled back into the same model. For example, self-distillation fine-tuning (SDFT) uses a demonstration-conditioned model as its own teacher and demonstrates that such on-policy distillation can acquire new skills while preserving prior capabilities (Shenfeld et al., 2026). Self-distillation policy optimization (SDPO) similarly uses a better-informed, feedback-conditioned teacher to provide dense token-level supervision to the student policy (Hübotter et al., 2026). Together, these works suggest that on-policy self-distillation may be a practical recipe for continual learning, combining the on-policy nature of RFT that mitigates forgetting with the efficiency of dense supervised updates for learning new knowledge.

However, this view conflates two distinct ingredients: where the data come from and which objective is used to update the model. On-policy sampling determines the histories on which the model is trained, but the training objective determines which continuation distribution is reinforced at each history. This distinction is especially important for dense self-distillation. When every token becomes a training target, teacher drift, noisy rationales, formatting conventions, and domain-specific artifacts can be reinforced as repeatedly as useful behavior (Li et al., 2026; Kim et al., 2026). Thus, on-policy

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

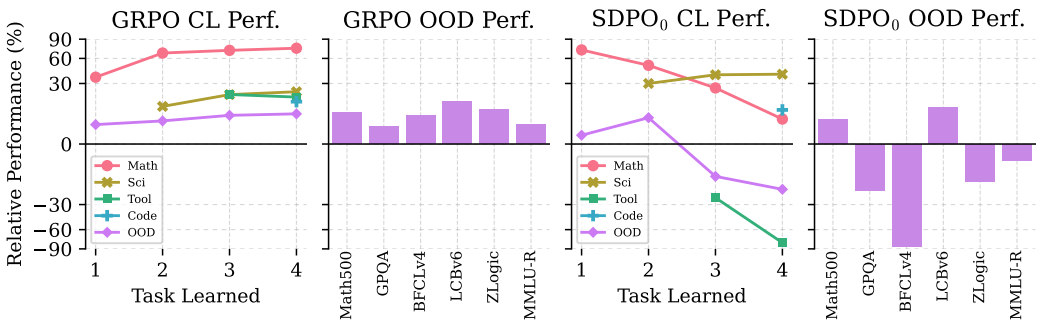


Figure 1: GRPO versus SDPO relative performance to base model (%) across sequentially learned tasks (1–4) and out-of-distribution (OOD) benchmarks. The line plots track performance retention across domains (MATH, SCIENCE, TOOLUSE, CODING, and macro OOD average) as new tasks are acquired. The bar charts break down final relative performance across specific OOD benchmarks.

self-distillation may avoid the off-policy mismatch of SFT, but it does not automatically inherit the conservative update bias of sequence-level on-policy RFT.

This paper studies whether on-policy self-distillation enables reliable continual post-training. We focus on self-distillation policy optimization (SDPO, Hübotter et al., 2026) and compare it with sequence-level reward optimization (GRPO, Shao et al., 2024) from two perspectives:

- What are the key factors that make on-policy self-distillation beneficial? (§ 2)
- Does SDPO outperform GRPO in generalization or continual learning? (§ 3)

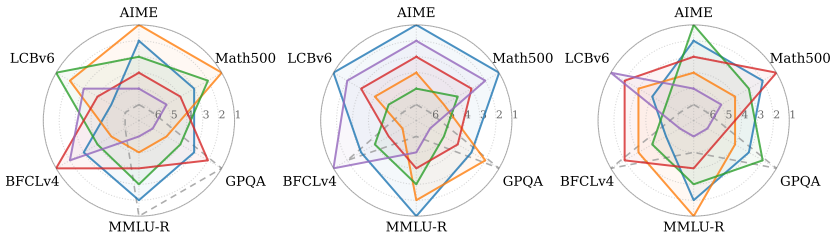
To answer these questions, we conduct post-training experiments in single-domain and multi-domain continual settings. We vary the supervision density by comparing GRPO, standard SDPO, and SDPO variants with additional teacher strategies (§ 2.1) or supervision tweaks (§ 2.2), and evaluate on in-distribution and general benchmarks to assess specialization, retention, and transfer (§ 3.1 and 3.2). Finally, we connect behavior to diagnostics—parameter and response drift (§ 4.1), fixes to mitigate forgetting (§ 4.2), and theoretical analysis (§ 4.3) to interpret our observations.

Our results show that on-policy self-distillation offers a powerful yet fragile training signal (Figure 1). SDPO can substantially improve performance on the current training domain, but it also increases the risk of drift, interference, and even collapse. Our observations suggest that self-distillation introduces a trade-off: it strengthens the local learning signal, but also increases sensitivity, domain mismatch, and accumulated artifacts. Our main contributions are:

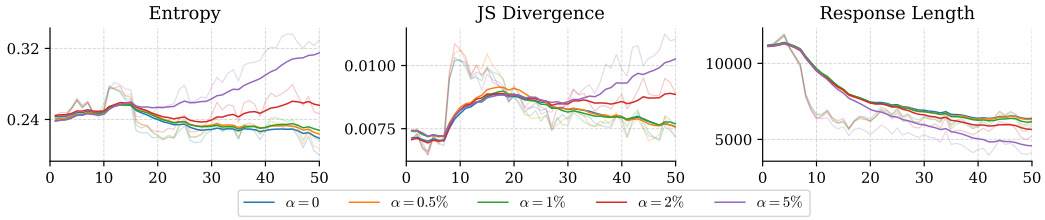
- We separate two factors often conflated in continual post-training: on-policy data and the objective used to turn on-policy rollouts into updates. This lets us ask whether on-policy self-distillation inherits the retention benefits attributed to on-policy RL.
- We show that SDPO is a strong specialization method but a fragile continual-learning signal. Its benefits depend on teacher stability and token reliability; its forgetting concentrates on neighboring-yet-misaligned tasks and accumulates across domains.
- We diagnose how SDPO's forgetting behavior arises via Parameter drift, response drift, collapse analysis, and an excess-KL theoretical view.

2 TOKEN-LEVEL DENSE SUPERVISION IS POWERFUL BUT SENSITIVE

First, we argue that SDPO's benefits depend on the quality of the dense signal. SDPO distills token logits from the teacher model, which improves sample efficiency because every generated token contributes supervision. However, the same mechanism reinforces teacher noise, temporal mismatch, and formatting artifacts many times within a single response. We therefore begin by asking: *how do token-level reward stability (§ 2.1) and density (§ 2.2) affect self-distillation?*



(a) Performance ranking across six benchmarks after MATH, TOOLUSE, and SCIENCE training.



(b) Training dynamics of MATH-SDPO under different teacher EMA rates $\alpha \in \{0, 0.5\%, 1\%, 2\%, 5\%\}$.

Figure 2: Effect of SDPO teacher EMA update rate α: no single α is globally optimal. **(a)** Ranking-based radar plots; a larger area is better. **(b)** Training dynamics during MATH-SDPO.

2.1 TOKEN-LEVEL SUPERVISION REQUIRES FRESH BUT TEMPORALLY STABLE TEACHERS

A natural intuition in SDPO is that the teacher should be updated frequently: a fresher teacher should better reflect the student's latest improvements and therefore provide denser and stronger token supervision. To test this intuition, we sweep the EMA update rate $\alpha \in \{0, 0.5\%, 1\%, 2\%, 5\%\}$:

$$\Theta^{(k)}_{\text{teacher}} = (1-\alpha)\Theta^{(k-1)}_{\text{teacher}} + \alpha\Theta^{(k)}_{\text{student}}.$$

Surprisingly, freshness alone does not predict performance. As shown in Figure 2a, no single α is uniformly optimal: in MATH-SDPO, $\alpha = 1\%$ gives the best overall profile, while $\alpha = 5\%$ performs worst; in TOOLUSE training, the frozen teacher is best, although $\alpha = 5\%$ is competitive. This raises a question: *if a larger EMA rate makes the teacher fresher, why can it hurt SDPO so severely?*

We argue that EMA entangles two different properties of the teacher: *freshness* and *temporal stability*. A larger α makes the teacher weights closer to the current student, but it also makes the teacher a more rapidly moving target. This is particularly risky for distillation: because SDPO applies supervision at every generated token, small teacher fluctuations can be reinforced repeatedly within a single response. The training curves in Figure 2b support this view. Large EMA rates, especially $\alpha = 2\%$ and $\alpha = 5\%$, observe late-stage increases in entropy and JS divergence, indicating growing teacher–student mismatch and less reliable token targets. Smaller or frozen teachers avoid this instability, but they introduce the opposite problem: the teacher can become stale and provide a weakening optimization signal as the student evolves.

To separate freshness from per-step teacher dynamics, we introduce a simple *restart-and-freeze* strategy. Instead of updating the teacher continuously, we periodically refresh it from the current student and then keep it frozen within the next interval:

$$\Theta_{\text{teacher}} \leftarrow (1-\gamma)\Theta^{\text{old}}_{\text{teacher}} + \gamma\Theta^{(k)}_{\text{student}}, \quad 0 < \gamma \leq 1.$$

This design preserves teacher freshness at refresh points while removing the step-to-step volatility of EMA updates. For comparison with EMA, we match the effective k-step refresh amount by $\gamma \approx 1-(1-\alpha)^k$; for example, $\alpha = 5\%$ over $k = 10$ steps corresponds to $\gamma \approx 0.40$.

The results in Table 1 support this decomposition. Replacing continuous EMA with matched restart-and-freeze substantially repairs the destructive behavior of $\alpha = 5\%$: on AIME, performance improves from 34.38% to 55.00%, and the average score improves from 65.34% to 71.84%, a gain of 6.50 points. Importantly, the restart strategy also improves over the frozen teacher: hard restart with $\gamma = 1$ outperforms the frozen teacher on all five benchmarks, improving the average score from 71.88% to

Table 1: MATH-SDPO performance (Acc@8). Restart-and-freeze offers a middle ground: it periodically re-initializes the teacher but prevents per-step noise from being distilled repeatedly.

Teacher Strategy	AIME	Math500	GPQA	ZLogic	MMLU-R	Avg
–	32.71	73.85	62.94	81.62	86.02	67.43
EMA $\alpha = 0$	56.42	84.29	58.72	74.50	85.47	71.88
Restart $\gamma = 1$	**57.50**	**85.05**	**60.86**	**74.68**	**86.60**	**72.94**$_{\uparrow 1.06}$
EMA $\alpha = 5\%$	34.38	78.97	55.87	**73.68**	83.82	65.34
Restart $\gamma = 40\%$	**55.00**	**84.62**	**59.03**	73.50	**87.05**	**71.84**$_{\uparrow 6.50}$

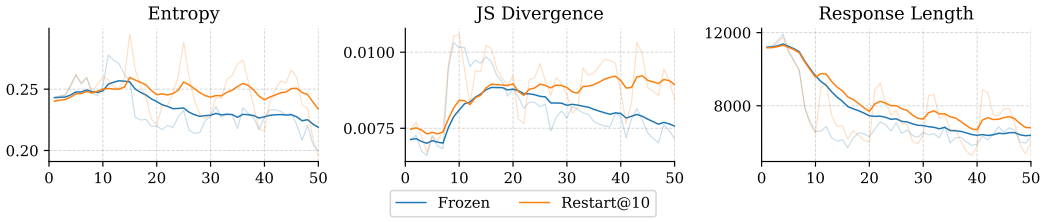


Figure 3: Training dynamics during MATH-SDPO with teacher *frozen* versus 10-step-hard-restart.

72.94%. The problem is not freshness per se: freshness is useful only when delivered through a stable teacher signal. The training dynamics in Figure 3 further illustrate this point. The frozen teacher produces smooth curves, but its entropy and JS divergence gradually decline, suggesting that the teacher becomes increasingly outdated. In contrast, restart-and-freeze produces bounded oscillations aligned with refresh events: each restart renews the teacher signal, and avoids noisy per-step target changes. The JS divergence remains bounded, with neither late-stage blow-up nor collapse.

Takeaways 1

Teacher design is central to on-policy self-distillation, with a *stability–freshness* dilemma: a frozen teacher is stable but stale in the long term; a fast EMA teacher is fresh but can amplify online noise. Periodic refresh-and-freeze provides more reliable targets.

2.2 CoT Distillation: More Supervision Is Not Always Better

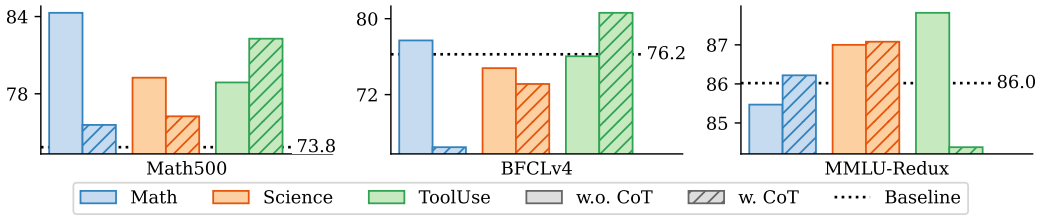
We next consider the supervision density of SDPO, by additionally distilling Chain-of-Thought tokens. This is a stronger test of the dense-supervision hypothesis: if more token-level targets were uniformly beneficial, CoT distillation should consistently improve performance. Figure 4a shows the opposite: CoT helps TOOLUSE training but hurts MATH and SCIENCE training.

We attribute this contrast to the reliability of intermediate tokens. CoT distillation does not merely add more supervision; it changes the target from answer-producing behavior to a particular reasoning trajectory. In long-form MATH and SCIENCE tasks, such trajectories are underdetermined and often contain redundant self-checks, uncertain interpretations, formatting habits, or incorrect intermediate claims. These tokens are weakly tied to final correctness, so distilling them can dilute credit assignment and encourage imitation of reasoning artifacts. This is reflected in MATH training by longer responses, higher entropy, and larger JS divergence (Figure 4b).

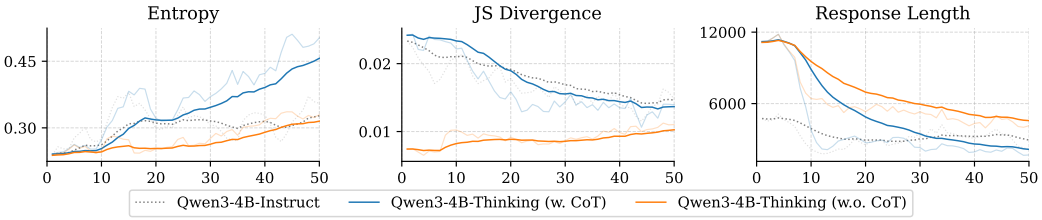
By contrast, TOOLUSE traces are short and schema-constrained. CoT therefore acts as useful process supervision in TOOLUSE training, yielding stable dynamics and improved BFCLv4 performance (Figure 4c). Additional EMA ablations further show that long-CoT supervision becomes especially harmful when the teacher is updated rapidly (Table E.2), consistent with a confirmation-bias loop in which noisy reasoning patterns are copied into the teacher and reinforced again.

Takeaways 2

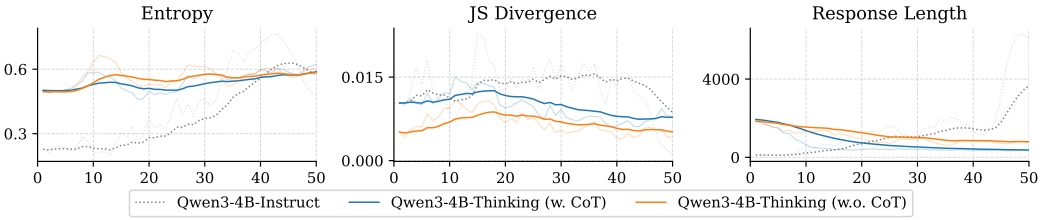
More supervision is useful only when reliable. CoT distillation benefits short and structured traces, but can harm long, noisy, or artifact-prone rationales.



(a) SDPO Performance (Acc@8) trained on MATH, SCIENCE, and TOOLUSE use against baseline (before training).



(b) SDPO training dynamics on MATH (which requires long CoT).



(c) SDPO training dynamics on TOOLUSE (which requires short CoT).

Figure 4: With and without CoT SDPO Performance (Acc@8) and training dynamics.

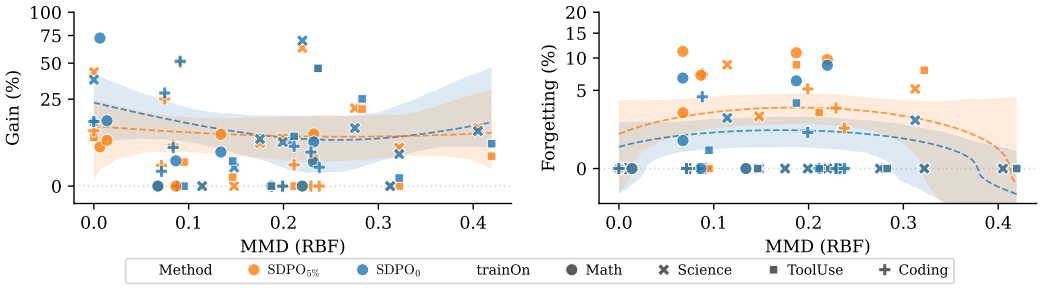


Figure 5: Relative performance change in terms of gain G and forgetting F versus Maximum Mean Discrepancy with a Radial Basis Function kernel.

3 SDPO TRADES RETENTION FOR SPECIALIZATION

The previous section shows that SDPO's dense signal is conditional: it helps only when the teacher is stable and the distilled tokens are reliable. We now ask whether these local benefits translate into broader training advantages. *Does token supervision improve out-of-distribution (OOD) generalization (§ 3.1), or mitigate forgetting in continual post-training (§ 3.2)?*

3.1 RETHINKING OOD GENERALIZATION: INTERMEDIATE-DISTANCE INTERFERENCE

SDPO's main advantage is specialization, rather than uniformly better generalization (Table 2). Under MATH training, SDPO_0 is the strongest math specialist, raising AIME from 32.71% to 56.42% and Math500 from 73.85% to 84.29%. Similarly, CODING-SDPO_0 improves LCBv6 by 13.39%. These results indicate that SDPO can strongly exploit source-like evaluation distributions.

Table 2: Performance (Acc@8) of models trained on a single domain. Rows indicate training domain and method; columns indicate evaluation benchmarks. SDPO_0 and $\text{SDPO}_{5\%}$ denote SDPO with teacher EMA 0 and 5%, respectively. $\text{SDPO}_0^{\text{CoT}}$ denotes SDPO_0 with CoT distilled.

		MATH		SCIENCE		TOOLUSE		CODING		LOGIC	KNOW.
	Method	AIME	Math500	SCIENCE	GPQA	TOOLUSE	BFCLv4	CODING	LCBv6	ZLogic	MMLU-R
	–	32.71	73.85	50.54	62.94	48.90	76.25	56.39	46.73	81.62	86.02
MATH	GRPO	44.67	76.84	51.33	62.83↓	50.48	78.33	60.73	52.93	80.82↓	85.74↓
MATH	$\text{SDPO}_{5\%}$	34.38	78.97	45.00↓	55.87↓	53.31	78.12	52.37↓	50.86	73.68↓	83.82↓
MATH	SDPO_0	56.42	84.29	47.36↓	58.72↓	52.06	77.71	57.58	48.52	74.50↓	85.47↓
SCIENCE	GRPO	35.00	73.47↓	59.05	62.50↓	48.71↓	75.00↓	55.53↓	45.90↓	87.40	87.50
SCIENCE	$\text{SDPO}_{5\%}$	53.34	78.42	72.26	57.39↓	51.29	72.29↓	60.66	56.11	90.68	84.10↓
SCIENCE	SDPO_0	55.63	79.25	69.46	61.62↓	50.55	74.79↓	60.01	51.94	89.80	87.00
TOOLUSE	GRPO	38.55	76.60	49.11↓	62.25↓	57.90	79.58	57.88	47.69	77.35↓	84.95↓
TOOLUSE	$\text{SDPO}_{5\%}$	47.71	78.57	46.55↓	57.39↓	52.76	77.71	54.94↓	55.90	84.02	86.25
TOOLUSE	SDPO_0	47.71	78.88	50.65	60.73↓	55.51	76.04↓	61.00	58.48	86.48	87.82
CODING	GRPO	33.54	73.70↓	51.96	62.63↓	50.37	75.42↓	57.84	46.81	81.90	85.52↓
CODING	$\text{SDPO}_{5\%}$	49.17	77.98	47.92↓	58.33↓	49.63	73.96↓	62.10	58.47	80.53↓	87.25
CODING	SDPO_0	49.59	77.50	50.00↓	60.29↓	51.47	79.17	64.18	60.12	82.58	86.65

However, this specialization does not translate into uniform retention. We decompose relative performance change into positive *gain* and negative *forgetting* (Equations (B.8) and (B.9)) and show that the SDPO-induced forgetting pattern is non-monotonic, as Figure 5 plots gain and forgetting against embedding distance measured by Maximum Mean Discrepancy (MMD, Gretton et al., 2012, Equation (B.4)). Source-like tasks (e.g., MMD $\lesssim 0.05$) and very distant tasks (e.g., MMD $\gtrsim 0.4$) mostly improve, while the largest drops concentrate in intermediate-distance tasks.

GPQA is the clearest example of intermediate-forgetting: it is not the farthest, yet it consistently degrades after SDPO training from multiple source domains. We interpret this pattern as evidence that GPQA may lie in an interference region: it is close enough to be affected by source-domain updates, yet insufficiently aligned to benefit from them. More broadly, we observe SDPO degradation on MATH-to-SCIENCE, MATH-to-ZLogic, and SCIENCE-to-BFCLv4 (additional cases listed in Table E.4).

These results suggest that SDPO's OOD failures are consistent with an intermediate-distance interference pattern, rather than a simple monotonic function of distributional distance. One explanation is that distillation specializes the model toward the source-domain response manifold. When an OOD task is very close, this specialization may transfer; when it is very far, the perturbation may not strongly affect it. The most vulnerable tasks are partially aligned OOD tasks: close enough to be changed, but misaligned enough not to benefit. This non-monotonic pattern is consistent with prior theory of continual learning with linear models (Lin et al., 2023; Goldfarb et al., 2024), and our results suggest that a similar interference regime emerges in LLM continual post-training.

Takeaways 3

For OOD generalization, SDPO improves source-aligned tasks and leaves distant tasks relatively unaffected or even improved, but interferes most strongly with partially aligned tasks that are close enough to be shifted and yet misaligned enough not to benefit.

3.2 CONTINUAL LEARNING EXPOSES THE ACCUMULATED COST OF DENSE SUPERVISION

Single-domain interference becomes more severe when updates are composed across domains. We therefore train models sequentially on MATH → SCIENCE → TOOLUSE → CODING and evaluate after each stage.

The continual setting reverses the single-domain picture. SDPO_0 is a strong early specializer: it reaches 56.42% on MATH and 65.70% on SCIENCE, exceeding GRPO's 44.67% and 56.39%, respectively. However, these gains do not survive later domain shifts: after the full sequence, SDPO_0 drops to 34.38% on MATH and 9.93% on TOOLUSE, and also falls below the base model on LOGIC and KNOWLEDGE. $\text{SDPO}_0^{\text{CoT}}$ degrades even earlier, consistent with the instability of long-CoT token targets. The most severe case is $\text{SDPO}_{5\%}$, which collapses after the third stage and remains near zero. By contrast, GRPO

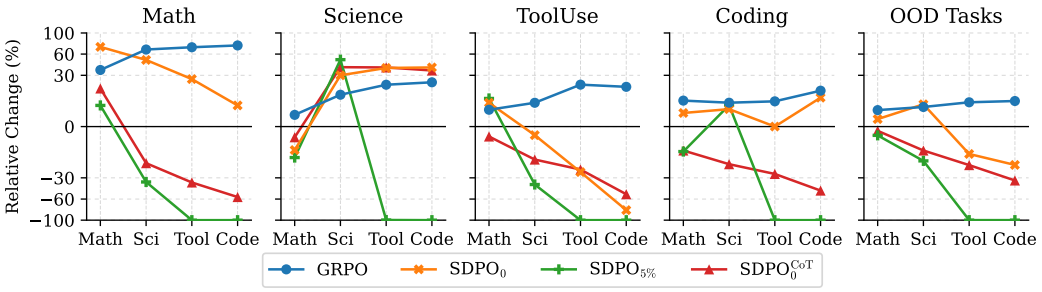


Figure 6: Normalized accuracy percentage change (y-axis) across continual training stages (x-axis). GRPO improves almost monotonically, whereas SDPO degrades more often. Each subplot denotes one evaluation benchmark: MATH, SCIENCE, TOOLUSE, CODING, and out-of-domain tasks.

adapts more conservatively but retains capabilities more reliably, achieving final improvements over the base model across all evaluated categories (Figure 6).

Takeaways 4

For continual post-training, dense token-level supervision improves local adaptation but is not inherently retention-preserving; when composed across tasks, it can accumulate drift and interference that result in forgetting.

4 UNDERSTANDING SDPO IN CONTINUAL POST-TRAINING

4.1 DENSE SUPERVISION CAUSES LARGER PARAMETER AND RESPONSE DRIFT

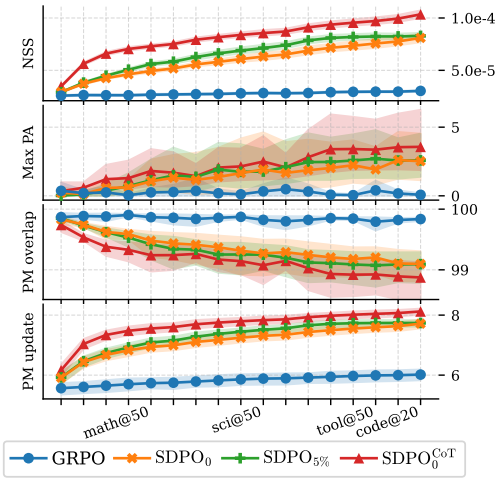
In our benchmarks, SDPO adapts more aggressively than GRPO. To understand why, we look inside the model and ask whether SDPO's greater behavioral changes correspond to larger parameter shifts. For each LLM weight matrix, we compute the compact singular value decomposition (Equation (B.5)) to inspect the input-output directions and their spectral strengths. We quantify how training changes the geometry, spectral magnitude, subspace orientation, and dominant parameter regions of the model. Specifically, we compute the following metrics between the initial $\mathbf{W}^{(0)}$ and step k $\mathbf{W}^{(k)}$:

- *Normalized spectral shift*: ℓ_2-normalized drift of the singular-value profile per Equation (B.11). Larger values indicate substantial strength changes in dominant directions.
- *Principal-angle rotation*: maximum rotation within top-k left singular subspaces, as shown in Equation (B.14). A small value implies better preservation of dominant subspaces.
- *Principal-mask overlap*: intersection-over-union of principal masks before and after training as defined in Equation (B.15). High values mean dominant entries remain important.
- *Principal-mask update sparsity*: intersection-over-union of the initial principal mask and the changed-parameter mask as defined in Equation (B.17). A high value indicates that training directly modifies structurally important parameters.

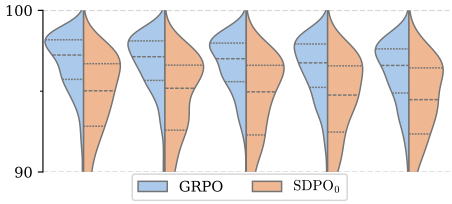
Figure 7 reveals a consistent parameter and response drift hierarchy. GRPO changes the model least: its spectral shift and subspace rotation remain small, and its principal-mask overlap stays near 99.9%. SDPO variants drift farther, with $\text{SDPO}_0^{\text{CoT}}$ producing the largest subspace change. This pattern is consistent with an accumulated-drift explanation of continual-learning failure: SDPO produces notably larger spectral shifts, more principal-parameter modifications, and response embedding distribution changes than GRPO. The ordering $\text{SDPO}_0^{\text{CoT}} > \text{SDPO}_{5\%} > \text{SDPO}_0 > \text{GRPO}$ matches the aggressiveness of the supervision signal: CoT increases token coverage, whereas fast EMA increases target non-stationarity. This suggests that denser token targets drive stronger internal updates.

Takeaways 5

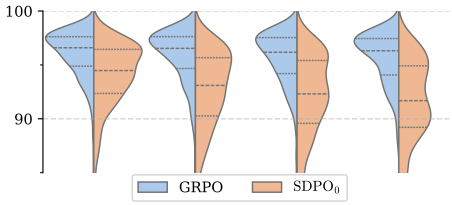
Faster adaptation is accompanied by larger drift. SDPO reshapes parameters and responses more than GRPO, which aids specialization but undermines continual-learning stability.



(a) Shift of self-Attention V projection during continual learning.



(b) Embedding similarity shift during MATH training; x-axis denotes checkpoints at steps $k = 10, 20, \ldots, 50$.



(c) Embedding similarity shift during continual learning; x-axis denotes checkpoints after tasks MATH, SCIENCE, TOOLUSE, and CODING.

Figure 7: **(a)** Model parameter drift (y-axis) against training steps (x-axis). Metrics are averaged across all layers with 95% confidence intervals. NSS, PA, and PM denote normalized spectral shift, principal angle, and principal mask, respectively. **(b, c)** Cosine similarity (%) between AIME rollout(@8) embeddings before and after GRPO (left half) versus SDPO$_0$ (right half) training.

Table 3: Comparison of linear and spectral weight editing for mitigating forgetting in SDPO$_0$ (Acc@8). Both methods merge MATH-SDPO$_0$with the pre-update weights at 50% scale; spectral merging additionally retains only the top $r = 16$ singular components. Δ' denotes changes on non-MATH tasks.

	MATH		SCIENCE		TOOLUSE		CODING		LOGIC	KNOW.	Avg	Δ'
Method	AIME	Math500	SCIENCE	GPQA	TOOLUSE	BFCLv4	CODING	LCBv6	ZLogic	MMLU-R		
Baseline	32.71	73.85	50.54	62.94	48.90	76.25	56.39	46.73	81.62	86.02	61.60	–
MATH-GRPO	44.67	76.84	51.33	62.83	50.48	78.33	60.73	52.93	80.82	85.74	64.47	↑ 1.73
MATH-SDPO$_0$	56.42	84.29	47.36	58.72	52.06	77.71	57.58	48.52	74.50	85.47	64.26	↓ 0.93
+Linear$^{50\%}$	51.67	84.25	48.81	62.25	50.92	77.29	56.47	46.65	76.42	85.65	64.04	↓ 0.62
+Spectral$^{50\%}_{r=16}$	50.21	82.05	52.26	61.62	48.90	75.21	57.95	46.61	79.38	86.40	64.06	↓ 0.13

4.2 MITIGATING SDPO FORGETTING WITH WEIGHT EDITS

The preceding analysis suggests that continual self-distillation may accumulate both beneficial task adaptation and undesirable behavioral shifts, which motivates weight-space editing to reduce forgetting. We investigate two merging strategies between the pre-update weights W_0 and the post-trained weights W_1, with task vector $\Delta = W_1 - W_0$. *Linear merging* interpolates the full task vector as $W = W_0 + s\Delta = (1 - s)W_0 + sW_1$. *Spectral merging* instead retains only the r largest singular components of each matrix-valued task vector, yielding $W = W_0 + s\ \mathrm{trim}_r(\Delta)$. This design tests whether dominant low-rank directions preserve task-specific gains while suppressing weaker directions that may contribute to interference and out-of-domain forgetting.

Results in Table 3 (full results in Table E.6) show that both strategies mitigate several behavioral shifts induced by SDPO$_0$ while largely preserving its overall performance. At a 50% merge ratio, linear merging improves GPQA from 58.72 to 62.25 and ZLogic from 74.50 to 76.42, bringing them closer to the GRPO results of 62.83 and 80.82, respectively. Spectral merging with $r = 16$ provides stronger recovery on ZLogic (79.38) and also improves SCIENCE to 52.26 and MMLU-R to 86.40, surpassing both the unmerged SDPO$_0$ and GRPO variants on the latter two benchmarks. These improvements involve some reduction in SDPO$_0$'s large mathematics gains, but the average remains nearly unchanged. Thus, merging produces a more balanced model, with spectral trimming in

particular moving SDPO_0 toward the lower-forgetting behavior of GRPO without materially sacrificing aggregate performance.

4.3 Theoretical View on Forgetting: Excess KL Drift

Existing studies have shown that on-policy RL is biased toward solutions that solve the new task while staying close to the base policy in KL (Shenfeld et al., 2025). However, SDPO does not inherit this bias: its rollouts are on-policy, but its local target is a teacher distribution conditioned on augmented context, rather than the KL-minimal distribution induced by a sequence-level reward.

Fix a prompt x with base sequence distribution $P_0(\cdot \mid x)$, and let S_x be the set of successful responses under a binary sequence reward. Among all sequence distributions Q with $Q(S_x) = \alpha$, the KL-minimal one—the *matched-success Razor policy* $Q^{\dagger}_{\alpha,x}$, with token kernel $q^{\dagger}_{\alpha,x}(\cdot \mid h_t)$ at history $h_t = (x, y_{<t})$—reweights the base policy only as needed to change the probability of eventual success. In contrast, SDPO distills a teacher $T_c(\cdot \mid h_t) = \pi_T(\cdot \mid x, c, y_{<t})$ that observes augmented context c, such as a solution or rationale. For a strict distillation divergence D, its population token target is the teacher projection $B(\cdot \mid h_t) \in \arg\min_{u \in \Delta(\mathcal{V})} \mathbb{E}_{c|h_t} D\big(u, T_c(\cdot \mid h_t)\big)$, which equals T_c when c is deterministic. Let $P_B(\cdot \mid x)$ be the sequence distribution induced by B and set $\alpha_x = P_B(S_x \mid x)$.

Theorem 1 (Teacher projection induces excess KL drift)**.** *Assume* $0 < \alpha_x < 1$. *Then*

$$\begin{aligned} &D_{\mathrm{KL}}\big(P_B(\cdot \mid x) \,\|\, P_0(\cdot \mid x)\big) - D_{\mathrm{KL}}\big(Q^{\dagger}_{\alpha_x,x} \,\|\, P_0(\cdot \mid x)\big) \\ &\quad = \mathbb{E}_{y \sim P_B(\cdot|x)} \sum_{t=1}^{|y|} D_{\mathrm{KL}}\big(B(\cdot \mid h_t) \,\|\, q^{\dagger}_{\alpha_x,x}(\cdot \mid h_t)\big) \geq 0. \end{aligned} \tag{1}$$

The inequality is strict whenever the SDPO teacher target differs from the sequence-reward Razor kernel on a set of histories with positive P_B-probability.

Equality is therefore restrictive: the teacher target must coincide with the Razor kernel at every visited history, which is not implied by teacher–student parameter sharing. Because the teacher conditions on c, it may select a particular reasoning trace, formatting pattern, or artifact even when many continuations are equally successful. Thus, SDPO's on-policy sampling controls where supervision is applied, but the distribution selected at each token is the teacher projection, not the KL-minimal sequence-reward projection. Combined with the KL-forgetting law, the excess drift in Equation (1) predicts additional forgetting; Appendix C provides full proofs, a matched forward-KL analogue, and a corollary formalizing this prediction.

Takeaways 6

On-policy data alone does not guarantee forgetting mitigation. Although SDPO trains on on-policy histories, each token is projected toward a privileged teacher rather than a KL-minimal successful policy; this can amplify artifacts and induce excess drift across continual post-training.

5 Conclusion

This paper studied whether on-policy self-distillation improves continual post-training. Our results indicate that SDPO can effectively specialize when teacher signals are stable and aligned with the target domain. However, SDPO is not uniformly beneficial for continual learning. Its token supervision mechanism can amplify both useful signals and harmful artifacts, making it sensitive to teacher quality, update schedules, and token structure. Compared with GRPO, SDPO variants exhibit greater forgetting, instability, parameter drift, and response-manifold shifts.

The main lesson is not that on-policy self-distillation methods are ineffective, but that they are not automatically safe. Token supervision can be a rapid specialization signal, but it can also be dangerous: each token can reinforce useful signals and harmful artifacts. Sequence-level methods such as GRPO learn more slowly, but in our experiments, they preserve capabilities more reliably. Thus, the future of on-policy self-distillation for continual learning likely depends not on making the signal denser, but on safeguarding it through token weighting, teacher controls, or data balancing strategies, thereby making it more selective, stable, and artifact-aware.

AI USE STATEMENT

In this work, we used generative AI tools for proving mathematical claims, namely to help formulate and verify mathematical claims and to assist in the writing of proofs. Additionally, we used generative AI tools to aid and polish writing, in particular to edit the paper to improve readability. We have not used generative AI tools for the other tasks with required disclosure (such as generating synthetic data sets, implementing methods, or interpreting results), and the remaining required disclosure tasks are not applicable to this work.

We have reviewed all AI-assisted work: LLM-assisted mathematical claims and proofs were checked and verified for correctness by the authors, and all AI-polished text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

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

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

Kehua Feng, Keyan Ding, Weijie Wang, Xiang Zhuang, Zeyuan Wang, Ming Qin, Yu Zhao, Jianhua Yao, Qiang Zhang, and Huajun Chen. Sciknoweval: Evaluating multi-level scientific knowledge of large language models, 2024.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are we done with mmlu?, 2024.

Daniel Goldfarb, Itay Evron, Nir Weinberger, Daniel Soudry, and PAul HAnd. The joint effect of task similarity and overparameterization on catastrophic forgetting–an analytical model. In *The Twelfth International Conference on Learning Representations*, 2024.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Haiyang Guo, Fanhu Zeng, Fei Zhu, Jiayi Wang, Xukai Wang, Jingang Zhou, Hongbo Zhao, Wenzhuo Liu, Shijie Ma, Xu-Yao Zhang, et al. A comprehensive survey on continual learning in generative models. *arXiv e-prints*, pp. arXiv–2506, 2025b.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

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

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

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

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025.

Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning. In *International Conference on Machine Learning*, pp. 21078–21100. PMLR, 2023.

Wenzhuo Liu, Fei Zhu, Longhui Wei, and Qi Tian. C-clip: Multimodal continual learning for vision-language model. In *The Thirteenth International Conference on Learning Representations*, 2025.

MAA. American invitational mathematics examination (aime), 2024–2025. URL `https://maa.org/maa-invitational-competitions/`.

NVIDIA. Nvidia nemotron 3: Efficient and open intelligence, 2025. URL `https://arxiv.org/abs/2512.20856`. White Paper.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

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

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

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

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

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases, 2023.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Thinking Machines Lab. On-policy distillation. `https://thinkingmachines.ai/blog/on-policy-distillation/`, 2025. Blog post.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025a.

Zhihao Zhang, Qiaole Dong, Qi Zhang, Jun Zhao, Enyu Zhou, Zhiheng Xi, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Mingqi Wu, et al. Why reinforcement fine-tuning enables mllms preserve prior knowledge better: A data perspective. *arXiv preprint arXiv:2506.23508*, 2025b.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5871–5880, 2021.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, et al. The path not taken: Rlvr provably learns off the principals. *arXiv preprint arXiv:2511.08567*, 2025.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Related Work

A.1 Post-Training of Large Language Models

Reinforcement Learning for Post-Training. Reinforcement learning is widely used to post-train large language models with preference signals or reward functions derived from human feedback, automated evaluators, or verifiable correctness criteria (Stiennon et al., 2020; Ouyang et al., 2022; Ziegler et al., 2019; Bai et al., 2022). Recent methods, such as direct preference optimization (Rafailov et al., 2023), recast preference optimization as a supervised objective, while on-policy algorithms like group relative policy optimization (Shao et al., 2024) optimize relative advantages over sampled response groups, often improving stability and scalability in large-scale settings.

On-Policy Distillation. On-policy distillation (OPD) methods construct training targets from samples generated by the current policy or its improved variants, thereby converting reinforcement signals into supervised learning updates (Agarwal et al., 2024; Gu et al., 2024). Recent self-distillation methods remove the need for a separate teacher by running the same model under different information conditions: the student observes only the original prompt, while the teacher is augmented with privileged context c. The privileged signal can come from demonstrations or examples (Snell et al., 2022; Shenfeld et al., 2026), correct solutions or textual feedback (Hübotter et al., 2026), or revision contexts, hints, and task-level feedback transformations (Zhao et al., 2026; Ye et al., 2026).

Chain-of-Thought Reasoning. Chain-of-thought (CoT) prompting and training encourage LLMs to generate intermediate reasoning before final answers, often improving performance on arithmetic, symbolic, and commonsense tasks (Wei et al., 2022; Kojima et al., 2022). Under final-answer supervision, this lengthens the credit-assignment horizon; with step-level or process rewards, it offers denser feedback and improves interpretability (Uesato et al., 2022; Lightman et al., 2023). CoT also has drawbacks: it increases inference cost, may propagate errors across steps, and the generated rationales need not faithfully reflect the model's actual decision process (Turpin et al., 2023).

Continual Post-Training. Continual learning (Guo et al., 2025b) studies how models incorporate new knowledge without catastrophic forgetting of those already acquired. Common approaches use regularization, replay, or reference-model constraints to mitigate catastrophic forgetting across training stages (Kirkpatrick et al., 2017; Zhu et al., 2021; Liu et al., 2025). A growing line of work shows that RFT can substantially mitigate forgetting in continual post-training (Lai et al., 2025; Shenfeld et al., 2025; Chen et al., 2025). For example, Lai et al. (2025) find that this benefit is not mainly due to explicit KL penalties or chain-of-thought prompting, but rather to the implicit regularization induced by reward-guided policy updates. Complementarily, Shenfeld et al. (2025) argue that online RL is biased toward KL-minimal solutions among the policies that solve a new task, thereby reducing the distributional shift from the base model that is predictive of forgetting. Similarly, Chen et al. (2025) identify the use of on-policy data as a key factor behind RL's robustness.

A.2 Group Relative Policy Optimization

Group relative policy optimization (GRPO, Shao et al., 2024) assigns rewards at the sequence level. Given a prompt $x \sim \mathcal{D}$, GRPO samples a group of G responses $\{y_i\}_{i=1}^{G}$ from the current policy, i.e., $y_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, evaluates each response with a reward model to obtain rewards $\{r_i\}_{i=1}^{G}$, and computes group-relative advantages by normalizing rewards within the group, e.g., $A_i = (r_i - \text{mean}(\{r_j\}_{j=1}^{G}))/(\text{std}(\{r_j\}_{j=1}^{G}) + \epsilon)$. Training optimizes a PPO-style clipped objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\big(\rho_{i,t}A_i, \text{clip}(\rho_{i,t}, 1{-}\varepsilon, 1{+}\varepsilon)A_i\big) - \lambda D_{\text{KL}}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\big),$$

where $\rho_{i,t} = \pi_\theta(y_{i,t}|x, y_{i,<t}) / \pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})$ and π_{ref} is a reference policy used for KL regularization. In essence, GRPO is a critic-free policy optimization method that stabilizes reinforcement learning by using relative rewards among multiple sampled responses to the same prompt.

A.3 On-Policy Self-Distillation

Self-distillation policy optimization (SDPO, Hübotter et al., 2026) instead provides dense token-level rewards using the same model as both teacher and student. For each student-generated response $y \sim \pi_\theta(\cdot \mid x)$, the student is trained to align with the more-informed teacher $\pi_\theta(\cdot \mid x, c)$ having richer context c (including solutions or other auxiliary signals). Training minimizes the generalized Jensen–Shannon divergence (JSD, Equation (B.2)) between student and teacher distributions:

$$\mathcal{L}_{\mathrm{SDPO}}(\theta) = \sum_t D_{\mathrm{JSD}}(\beta)\left(\pi_\theta(\cdot \mid x, y_{<t}) \parallel \mathrm{stopgrad}\left(\pi_\theta(\cdot \mid x, c, y_{<t})\right)\right).$$

This yields dense supervision: instead of a single reward for the whole sequence, SDPO encourages the student distribution at each generated token to align with the more informed teacher distribution.

B Math Notations

Kullback–Leibler (KL) divergence. For two discrete distributions $P(\mathcal{X})$ and $Q(\mathcal{X})$,

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \tfrac{P(x)}{Q(x)}. \tag{B.1}$$

Generalized Jensen–Shannon (JS) divergence. A weighted average of two KL divergences from P and Q to their mixture with $\beta \in (0, 1)$ ($\beta = 0.5$ recovers the standard symmetric JS divergence):

$$D_{\mathrm{JSD}}(\beta)(P \parallel Q) = \beta D_{\mathrm{KL}}(P \parallel \beta P + (1-\beta)Q) + (1-\beta) D_{\mathrm{KL}}(Q \parallel \beta P + (1-\beta)Q). \tag{B.2}$$

Maximum Mean Discrepancy. It measures the difference between distributions P and Q by comparing their kernel mean embeddings in a reproducing kernel Hilbert space (RKHS):

$$\mathrm{MMD}^2(P, Q) = \|\mathbb{E}_{x\sim P}[\phi(x)] - \mathbb{E}_{y\sim Q}[\phi(y)]\|^2_{\mathcal{H}} \tag{B.3}$$

$$= \mathbb{E}_{x,x'\sim P}[k(x, x')] + \mathbb{E}_{y,y'\sim Q}[k(y, y')] - 2\mathbb{E}_{x\sim P, y\sim Q}[k(x, y)]. \tag{B.4}$$

Singular Value Decomposition (SVD). For each LLM weight matrix at training step k: $\mathbf{W}^{(k)} \in \mathbb{R}^{m\times n}$, we compute the compact singular value decomposition:

$$\mathbf{W}^{(k)} = \mathbf{U}^{(k)} \mathbf{\Sigma}^{(k)} {\mathbf{V}^{(k)}}^\top, \tag{B.5}$$

where $\mathbf{U}^{(k)} \in \mathbb{R}^{m\times d}$, $\mathbf{V}^{(k)} \in \mathbb{R}^{n\times d}$, and $d = \min(m, n)$.

Performance Change. We measure absolute (Δ) and relative (δ) change against the base model:

$$\Delta_{m,\mathrm{train},\mathrm{eval}} = P_{m,\mathrm{train},\mathrm{eval}} - P_{\mathrm{base},\mathrm{eval}}, \tag{B.6}$$

$$\delta_{m,\mathrm{train},\mathrm{eval}} = {}^{\Delta_{m,\mathrm{train},\mathrm{eval}}}\!/_{P_{\mathrm{base},\mathrm{eval}}} \times 100\%, \tag{B.7}$$

where m is the training method, train is the training domain, and eval is the evaluation benchmark. We define *gain* G and *forgetting* F as the positive and negative part of the relative change:

$$G_{m,\mathrm{train},\mathrm{eval}} = \max(0, \delta_{m,\mathrm{train},\mathrm{eval}}), \tag{B.8}$$

$$F_{m,\mathrm{train},\mathrm{eval}} = \max(0, -\delta_{m,\mathrm{train},\mathrm{eval}}). \tag{B.9}$$

Intersection-over-Union. For two masks A and B (e.g., predicted mask and ground-truth mask),

$$\mathrm{IoU}(A, B) = {}^{|A\cap B|}\!/_{|A\cup B|}, \tag{B.10}$$

where $|\cdot|$ denotes cardinality. The quantity lies in $[0, 1]$, with 1 for a perfect overlap.

Normalized spectral shift. It is defined as the ℓ_2-normalized drift of the singular-value profile:

$$\mathrm{NSS}\left(\mathbf{W}^{(0)}, \mathbf{W}^{(k)}\right) = \frac{\left\|\sigma(\mathbf{W}^{(0)}) - \sigma(\mathbf{W}^{(k)})\right\|_2}{\left\|\sigma(\mathbf{W}^{(0)})\right\|_2}. \tag{B.11}$$

Principal-mask (PM) and Principal-angles (PAs). Given $\mathbf{W}_r^{(k)} = \mathbf{U}_r^{(k)}\mathbf{\Sigma}_r^{(k)}{\mathbf{V}_r^{(k)}}^\top$ as the rank-r approximation, PM is

$$\mathbf{M}_{\text{principal}}{}^{(k)}[i,j] = \mathbb{I}\left(\left|\mathbf{W}_r^{(k)}[i,j]\right| \geq \tau_\eta^{(k)}\right), \tag{B.12}$$

where $\tau_\eta^{(k)}$ is a threshold controlled by the hyperparameter η. Principal angles are

$$\theta_i(\mathbf{W}^{(0)}, \mathbf{W}^{(k)}) = \arccos \sigma_i\left({\mathbf{U}_r^{(0)}}^\top \mathbf{U}_r^{(k)}\right), \qquad i = 1, \ldots, k, \tag{B.13}$$

where $\sigma_i(\cdot)$ denotes the i-th singular value.

Maximum principal-angle rotation. Max PA of the top-r left singular vectors:

$$\text{maxPA}(\mathbf{W}^{(0)}, \mathbf{W}^{(k)}) = \max_{i \leq r} \theta_i(\mathbf{W}^{(0)}, \mathbf{W}^{(k)}). \tag{B.14}$$

Principal-mask overlap. Intersection-over-Union between the initial and current PM masks:

$$\text{PMOverlap}(\mathbf{W}^{(0)}, \mathbf{W}^{(k)}) = \text{IoU}(\mathbf{M}_{\text{principal}}{}^{(0)}, \mathbf{M}_{\text{principal}}{}^{(k)}). \tag{B.15}$$

Changed-parameter-mask. Per-parameter mask with change threshold $\eta > 0$:

$$\mathbf{M}_{\text{param}}{}^{(k)}[i,j] = \mathbb{I}\left(\left|(\mathbf{W}^{(k)} - \mathbf{W}^{(0)})[i,j]\right| \geq \eta\right). \tag{B.16}$$

Principal-mask update sparsity. Intersection-over-Union between the initial principal mask and the changed-parameter mask:

$$\text{PMUpdateSparsity}(\mathbf{W}^{(0)}, \mathbf{W}^{(k)}) = \text{IoU}\left(\mathbf{M}_{\text{principal}}{}^{(0)}, \mathbf{M}_{\text{param}}{}^{(k)}\right). \tag{B.17}$$

C PROOFS

We provide the proofs for results in § 4.3. All distributions are over finite spaces. We assume that the base sequence distribution $P_0(\cdot \mid x)$ has full support. When the prompt x is fixed, we suppress it from notation.

Matched-success Razor policy. Let $S = S_x$ be the set of successful responses and let $z = P_0(S \mid x) \in (0,1)$. For a target success level $\alpha \in (0,1)$, define

$$\mathcal{Q}_\alpha = \{Q \in \Delta(\mathcal{Y}) : Q(S) = \alpha\}.$$

Lemma 2 (Closed form of the matched-success Razor policy)**.** *The solution of*

$$Q^\dagger_{\alpha,x} = \arg\min_{Q \in \mathcal{Q}_\alpha} D_{\text{KL}}\big(Q \parallel P_0(\cdot \mid x)\big)$$

is

$$Q^\dagger_{\alpha,x}(y) = P_0(y \mid x)\left[\tfrac{\alpha}{z}\mathbf{1}\{y \in S_x\} + \tfrac{1-\alpha}{1-z}\mathbf{1}\{y \notin S_x\}\right]. \tag{C.1}$$

Equivalently,

$$Q^\dagger_{\alpha,x} = \alpha P_0(\cdot \mid x, S_x) + (1-\alpha)P_0(\cdot \mid x, S_x^c).$$

Proof. Fix x and write $p = P_0(\cdot \mid x)$. Let $p^+ = p(\cdot \mid S)$ and $p^- = p(\cdot \mid S^c)$. Any $Q \in \mathcal{Q}_\alpha$ can be decomposed as

$$Q = \alpha Q^+ + (1-\alpha)Q^-,$$

where $Q^+ = Q(\cdot \mid S)$ and $Q^- = Q(\cdot \mid S^c)$. Then

$$D_{\text{KL}}(Q \parallel p) = \alpha \log \frac{\alpha}{z} + (1-\alpha)\log\tfrac{1-\alpha}{1-z} + \alpha D_{\text{KL}}(Q^+ \parallel p^+) + (1-\alpha)D_{\text{KL}}(Q^- \parallel p^-).$$

The first two terms are constant over $\mathcal{Q}_\alpha$. The remaining terms are nonnegative and vanish if and only if $Q^+ = p^+$ and $Q^- = p^-$. Thus the unique minimizer is

$$Q^\dagger_{\alpha,x} = \alpha p^+ + (1-\alpha)p^-,$$

which is exactly Equation (C.1). □

Lemma 3 (Token kernel induced by the Razor policy)**.** *Let $q^{\dagger}_{\alpha,x}(\cdot \mid h_t)$ be the autoregressive token kernel induced by $Q^{\dagger}_{\alpha,x}$ at history $h_t = (x, y_{<t})$. Define*

$$C_\alpha(h) = \mathbb{E}_{y\sim P_0(\cdot|h)}\left[\tfrac{\alpha}{z}\mathbf{1}\{y \in S_x\} + \tfrac{1-\alpha}{1-z}\mathbf{1}\{y \notin S_x\}\right]. \tag{C.2}$$

Then

$$q^{\dagger}_{\alpha,x}(a \mid h_t) = \pi_0(a \mid h_t)\frac{C_\alpha(h_t a)}{C_\alpha(h_t)}. \tag{C.3}$$

In the exact-success limit $\alpha \to 1$,

$$q^{\dagger}_{1,x}(a \mid h_t) = \pi_0(a \mid h_t)\frac{\Pr_{P_0}(y \in S_x \mid h_t a)}{\Pr_{P_0}(y \in S_x \mid h_t)}. \tag{C.4}$$

Proof. By Lemma 2,

$$Q^{\dagger}_{\alpha,x}(y) = P_0(y \mid x)w_\alpha(y),$$

where

$$w_\alpha(y) = \frac{\alpha}{z}\mathbf{1}\{y \in S_x\} + \frac{1-\alpha}{1-z}\mathbf{1}\{y \notin S_x\}.$$

For a history $h_t = (x, y_{<t})$,

$$q^{\dagger}_{\alpha,x}(a \mid h_t) = \Pr_{Q^{\dagger}_{\alpha,x}}(y_t = a \mid h_t) = \frac{\sum_{y_{>t}} P_0(a, y_{>t} \mid h_t)w_\alpha(y)}{\sum_{\tilde{y}_{\geq t}} P_0(\tilde{y}_{\geq t} \mid h_t)w_\alpha(\tilde{y})}.$$

The numerator equals

$$\pi_0(a \mid h_t)\mathbb{E}_{y\sim P_0(\cdot|h_t a)}[w_\alpha(y)] = \pi_0(a \mid h_t)C_\alpha(h_t a),$$

and the denominator equals $C_\alpha(h_t)$. This proves Equation (C.3). Letting $\alpha \to 1$ gives Equation (C.4). □

Teacher projection and excess reverse KL. For a strict distillation divergence D, the population SDPO token target is

$$B(\cdot \mid h_t) \in \arg\min_{u\in\Delta(\mathcal{V})} \mathbb{E}_{c|h_t} D\big(u, T_c(\cdot \mid h_t)\big).$$

When c is deterministic and D is generalized JSD, the minimizer is $B(\cdot \mid h_t) = T_c(\cdot \mid h_t)$ because generalized JSD is nonnegative and equals zero if and only if its two arguments are identical.

Let $P_B(\cdot \mid x)$ be the sequence distribution induced by $B(\cdot \mid h_t)$, and let

$$\alpha = P_B(S_x \mid x).$$

The corresponding matched-success Razor policy is $Q^{\dagger}_{\alpha,x}$.

Proof of Theorem 1. Fix x and suppress x from notation. Let $p = P_0(\cdot \mid x)$, $S = S_x$, $z = p(S)$, and $Q^{\dagger} = Q^{\dagger}_{\alpha,x}$. By Lemma 2,

$$Q^{\dagger}(y) = p(y)w_\alpha(y),$$

where $w_\alpha(y) = \alpha/z$ on S and $w_\alpha(y) = (1-\alpha)/(1-z)$ on S^c.

Since $P_B(S) = Q^{\dagger}(S) = \alpha$,

$$\sum_y P_B(y)\log w_\alpha(y) = \sum_y Q^{\dagger}(y)\log w_\alpha(y). \tag{C.5}$$

Therefore,

$$\begin{aligned} D_{\mathrm{KL}}(P_B \,\|\, p) - D_{\mathrm{KL}}(Q^{\dagger} \,\|\, p) &= \sum_y P_B(y)\log\frac{P_B(y)}{p(y)} - \sum_y Q^{\dagger}(y)\log\frac{Q^{\dagger}(y)}{p(y)} \\ &= \sum_y P_B(y)\log\frac{P_B(y)}{p(y)} - \sum_y Q^{\dagger}(y)\log w_\alpha(y) \\ &= \sum_y P_B(y)\log\frac{P_B(y)}{p(y)w_\alpha(y)} \\ &= D_{\mathrm{KL}}(P_B \,\|\, Q^{\dagger}), \end{aligned} \tag{C.6}$$

where we used Equation (C.5) in the third line.

Now write both sequence distributions autoregressively:

$$P_B(y \mid x) = \prod_{t=1}^{H} B(y_t \mid h_t), \qquad Q^\dagger(y \mid x) = \prod_{t=1}^{H} q^\dagger_{\alpha,x}(y_t \mid h_t).$$

Then the autoregressive KL chain rule gives

$$\begin{aligned} D_{\mathrm{KL}}(P_B \,\|\, Q^\dagger) &= \mathbb{E}_{y\sim P_B} \log \frac{P_B(y \mid x)}{Q^\dagger(y \mid x)} \\ &= \mathbb{E}_{y\sim P_B} \sum_{t=1}^{H} \log \frac{B(y_t \mid h_t)}{q^\dagger_{\alpha,x}(y_t \mid h_t)} \\ &= \mathbb{E}_{y\sim P_B} \sum_{t=1}^{H} D_{\mathrm{KL}}\big(B(\cdot \mid h_t) \,\|\, q^\dagger_{\alpha,x}(\cdot \mid h_t)\big). \end{aligned} \tag{C.7}$$

Combining Equation (C.6) and Equation (C.7) proves the theorem. The right-hand side is nonnegative, and it is strictly positive whenever the two token kernels differ on a set of histories with positive P_B-probability. □

Forward KL counterpart. The previous theorem uses reverse KL because RL's Razor is naturally stated as a reverse-KL projection. The empirical forgetting law is often measured by forward KL from the base policy to the fine-tuned policy, so we also give the matched-success forward-KL analogue.

Proposition 4 (Forward-KL excess at matched success)**.** *Fix x and let $p = P_0(\cdot \mid x)$. Let $S = S_x$, $z = p(S)$, and let $P_B(S) = \alpha \in (0,1)$. Define*

$$p^+ = p(\cdot \mid S), \qquad p^- = p(\cdot \mid S^c),$$

and

$$B^+ = P_B(\cdot \mid S), \qquad B^- = P_B(\cdot \mid S^c).$$

Then

$$D_{\mathrm{KL}}(p \,\|\, P_B) - D_{\mathrm{KL}}(p \,\|\, Q^\dagger_{\alpha,x}) = zD_{\mathrm{KL}}(p^+ \,\|\, B^+) + (1-z)D_{\mathrm{KL}}(p^- \,\|\, B^-) \geq 0. \tag{C.8}$$

Proof. Decompose P_B as

$$P_B = \alpha B^+ + (1-\alpha)B^-.$$

Then

$$D_{\mathrm{KL}}(p \,\|\, P_B) = z \log \frac{z}{\alpha} + (1-z) \log \frac{1-z}{1-\alpha} + zD_{\mathrm{KL}}(p^+ \,\|\, B^+) + (1-z)D_{\mathrm{KL}}(p^- \,\|\, B^-).$$

For $Q^\dagger_{\alpha,x} = \alpha p^+ + (1-\alpha)p^-$, the two conditional KL terms vanish:

$$D_{\mathrm{KL}}(p \,\|\, Q^\dagger_{\alpha,x}) = z \log \tfrac{z}{\alpha} + (1-z) \log \tfrac{1-z}{1-\alpha}.$$

Subtracting gives Equation (C.8). □

Connection to forgetting. We use the KL-forgetting law as an empirical bridge between distributional drift and prior-task forgetting.

Assumption 5 (KL-forgetting law)**.** *For policies matched in new-task success, let $\mathsf{F}(\pi)$ denote the drop in prior-task performance. Suppose there exists a strictly increasing function ω with $\omega(0) = 0$ such that*

$$\mathcal{K}_{\mathrm{fwd}}(\pi_1) - \mathcal{K}_{\mathrm{fwd}}(\pi_2) \geq \varepsilon \quad \Longrightarrow \quad \mathsf{F}(\pi_1) - \mathsf{F}(\pi_2) \geq \omega(\varepsilon),$$

where

$$\mathcal{K}_{\mathrm{fwd}}(\pi) = \mathbb{E}_{x\sim\mathcal{D}} D_{\mathrm{KL}}\big(P_0(\cdot \mid x) \,\|\, P_\pi(\cdot \mid x)\big).$$

Corollary 6 (Teacher mismatch implies additional forgetting)**.** *Let P_B be the SDPO teacher-projected sequence distribution and let $Q^{\dagger}_{\alpha,x}$ be the matched-success Razor policy with $\alpha = P_B(S_x \mid x)$. Define*

$$\Delta_{\mathrm{fwd}} = \mathbb{E}_{x\sim\mathcal{D}}\Big[z_x D_{\mathrm{KL}}\big(P_0(\cdot \mid x, S_x) \,\|\, P_B(\cdot \mid x, S_x)\big) + (1 - z_x) D_{\mathrm{KL}}\big(P_0(\cdot \mid x, S^c_x) \,\|\, P_B(\cdot \mid x, S^c_x)\big)\Big].$$

If $\Delta_{\mathrm{fwd}} > 0$, then under Assumption 5,

$$\mathsf{F}(P_B) - \mathsf{F}(Q^{\dagger}_{\alpha}) \geq \omega(\Delta_{\mathrm{fwd}}) > 0.$$

Proof. Taking expectation over $x \sim \mathcal{D}$ in Proposition 4 gives

$$\mathcal{K}_{\mathrm{fwd}}(P_B) - \mathcal{K}_{\mathrm{fwd}}(Q^{\dagger}_{\alpha}) = \Delta_{\mathrm{fwd}}.$$

The result follows directly from Assumption 5. □

D Training Details

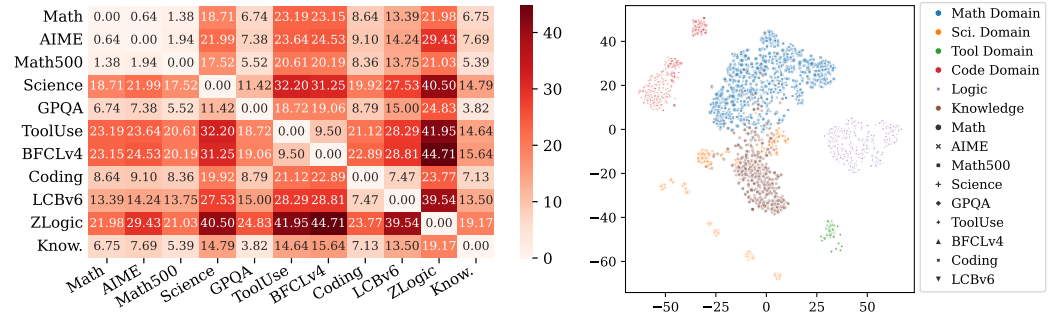
D.1 Models and Datasets

We adopt Qwen3 series (Qwen3-4B-Instruct-2507 and Qwen3-4B-Thinking-2507, Team, 2025), Olmo-3 series (Olmo-3-7B-Instruct-DPO and Olmo-3-7B-Think-DPO, Olmo et al., 2025) as base models[1] and train[2] on 4 diverse tasks—Math, Science, ToolUse, and Coding, with in-distribution and out-of-distribution[3] test sets to evaluate model performance measured in averaged accuracy (i.e., Acc@k). Data sources and training hyperparameters (learning rates according to Hübotter et al., 2026) are disclosed in Tables D.1 and D.2, respectively. We visualize the embeddings of each benchmark using Qwen3-Embedding-4B (Zhang et al., 2025a) in Figure D.1.

Table D.1: Data sources for training or evaluation purposes across domains. *AIME* denotes AIME24 and AIME25 combined. Abbreviations *ZLogic*, *MMLU-R* denote ZebraLogicBench, MMLU-Redux 2.0, respectively.

		Evaluation	
Domain	**Training**	In-Distribution	Out-Of-Distribution
Math	DAPO-Math-17k (Yu et al., 2026)	AIME (MAA, 2024–2025)	Math500 (Lightman et al., 2023)
Science	SciKnowEval (Feng et al., 2024)		GPQA (Rein et al., 2023)
ToolUse	ToolAlpaca (Tang et al., 2023)		BFCLv4 (Patil et al., 2025)
Coding	Dolci-Think-RL-7B (Olmo et al., 2025)		LCBv6 (Jain et al., 2025)
Logic	–		ZLogic (Lin et al., 2025)
Knowledge	–		MMLU-R (Gema et al., 2024)

[1] For Instruct models, SDPO and SDPO^{CoT} are equivalent due to absence of think tags.
[2] Training framework from SDPO official implementation `https://github.com/lasgroup/SDPO`.
[3] In-distribution and out-of-distribution are pragmatic labels, not strict measures of inter-dataset distance.



(a) Pairwise MMD (multiplier 10^{-2}) between benchmarks. (b) Visualization using t-SNE.

Figure D.1: Embedding visualization of benchmarks by domain.

Table D.2: Hyperparameters for SDPO and GRPO training with Qwen3-4B-Thinking-2507.

	Parameters	SDPO	GRPO
Data	Max. prompt length	2048	2048
	Max. response length	14336	14336
Batching	Question batch size	32	8
	Mini batch size	32	8
	Number of rollouts	8	8
Rollout	Inference engine	vLLM	vLLM
	Temperature	1.0	1.0
	Top-p	-1	-1
	Top-k	-1	-1
SDPO	Distillation Top-K	100	–
	Distillation divergence	Jensen–Shannon	–
	Teacher-EMA update rate (%)	$\alpha \in \{0, 0.5, 1, 2, 5\}$	–
	Rollout importance-sampling clip	2	–
PPO	KL coefficient (λ)	–	0.0
	ϵ-low	0.20	0.20
	ϵ-high	0.28	0.28
Training	Optimizer	AdamW	AdamW
	Learning rate	10^{-5}	10^{-6}
	Training steps	50 (20 for CODING)	50 (20 for CODING)
	Warmup steps	10	10
	Weight decay	0.01	0.01
	Gradient Clip Norm	1.0	1.0

D.2 DATA SAMPLES

AIME

```
Every morning Aya goes for a $9$-kilometer-long walk and stops at a coffee shop afterwards. When she walks
↪  at a constant speed of $s$ kilometers per hour, the walk takes her 4 hours, including $t$ minutes
↪  spent in the coffee shop. When she walks $s+2$ kilometers per hour, the walk takes her 2 hours and 24
↪  minutes, including $t$ minutes spent in the coffee shop. Suppose Aya walks at $s+\frac{1}{2}$
↪  kilometers per hour. Find the number of minutes the walk takes her, including the $t$ minutes spent in
↪  the coffee shop.
Please reason step by step, and put your final answer within \boxed{}.
```

BFCLv4

```
Your task is to answer the user's question using available tools.
You have access to the following tools:
Name: Tools
Description: A set of functions/tools that you can use to answer the user's question.
Documentation:
math.triangle_area_heron: Calculates the area of a triangle using Heron's formula, given the lengths of
↪  its three sides.
Parameters: {"side1": "integer. Length of the first side of the triangle.", "side2": "integer. Length of
↪  the second side of the triangle.", "side3": "integer. Length of the third side of the triangle."}
Output: Successful response.
 - Format: application/json
math.circle_area: Calculates the area of a circle given its radius.
Parameters: {"radius": "float. The radius of the circle."}
Output: Successful response.
 - Format: application/json
math.triangle_area_base_height: Calculates the area of a triangle using the formula (1/2)base*height.
Parameters: {"base": "float. The base length of the triangle.", "height": "float. The height of the
↪  triangle."}
Output: Successful response.
 - Format: application/json

Use the following format:
Thought: you should always think about what to do
Action: the action to take, should be one of the tool names.
Action Input: the input to the action, must be in JSON format. All of the action input must be realistic
↪  and from the user.

Begin!
Question: Calculate the area of a triangle, given the lengths of its three sides: 3, 4, and 5.
```

CODING

```
You are tasked with creating a function that retrieves the details of a specific assignment based on its
↪  ID. The function should take a string representing the assignment ID as input and return a dictionary
↪  containing the assignment details. For the purpose of this question, you can assume that the
↪  assignment details will be represented as a fixed dictionary for demonstration. Implement a function
↪  named `get_assignment_details` that takes an assignment ID and returns the corresponding assignment
↪  details as a dictionary. If the assignment ID does not exist, return an empty dictionary. The
↪  assignment details for valid IDs are predefined as follows: '101': {'title': 'Math Homework',
↪  'due_date': '2023-11-01'}, '102': {'title': 'Science Project', 'due_date': '2023-12-15'}, '103':
↪  {'title': 'History Essay', 'due_date': '2023-10-30'}.
```

GPQA

```
Two quantum states with energies E1 and E2 have a lifetime of 10^-9 sec and 10^-8 sec, respectively. We
↪  want to clearly distinguish these two energy levels. Which one of the following options could be their
↪  energy difference so that they can be clearly resolved?

A. 10^-8 eV
B. 10^-4 eV
C. 10^-11 eV
D. 10^-9 eV
Please reason step by step, and put your final answer within \boxed{}.
```

MATH

```
In triangle $ABC$, $\sin \angle A = \frac{4}{5}$ and $\angle A < 90^\circ$. Let $D$ be a point outside
↪  triangle $ABC$ such that $\angle BAD = \angle DAC$ and $\angle BDC = 90^\circ$. Suppose that $AD = 1$
↪  and that $\frac{BD}{CD} = \frac{3}{2}$. If $AB + AC$ can be expressed in the form
↪  $\frac{a\sqrt{b}}{c}$ where $a, b, c$ are pairwise relatively prime integers, find $a + b + c$.
Please reason step by step, and put your final answer within \boxed{}.
```

Math500

```
Convert the point $(0,3)$ in rectangular coordinates to polar coordinates.  Enter your answer in the form
↪  $(r,\theta),$ where $r > 0$ and $0 \le \theta < 2 \pi.$
Please reason step by step, and put your final answer within \boxed{}.
```

MMLU-R

```
Determine whether the polynomial in Z[x] satisfies an Eisenstein criterion for irreducibility over Q. x^2 -
↪  12
A. Yes, with p=2.
B. Yes, with p=3.
C. Yes, with p=5.
D. No.
Please reason step by step, and put your final answer within \boxed{}.
```

SCIENCE

```
Which of the following molecules is the only correct reactant used in the synthesis of the product
↪  "Nc1ccn(Cc2ccc(Cl)cc2C(F)(F)F)n1" ?

A: O=C1c2ccccc2C(=O)N1c1ccn(Cc2ccc(Cl)cc2C(F)(F)F)n1
B: O=C1c2ccccc2C(Cl)N1c1ccc2ccc(Cl)nc2n1
C: O=C1c2ccccc2C(=O)N1Cc1ccc(Oc2ccc(Cl)c(C(F)(F)F)c2)cc1
D: O=C1c2ccccc2C(=O)N1CC=C(Cc1ccccc1)C(F)(F)F
Please reason step by step, and put your final answer within \boxed{}.
```

TOOLUSE

```
Your task is to answer the user's question using available tools.
You have access to the following tools:
Name: Axolotl
Description: Collection of axolotl pictures and facts
Documentation:
getRandomAxolotlImage: Retrieve a random axolotl image with information on the image source.
Parameters: {}
Output: Successful response.
 - Format: application/json
 - Structure: Object{url, source, description}
searchAxolotlImages: Search for axolotl images based on specific criteria such as color, gender, and size.
Parameters: {"color": "string. One of: [wild, leucistic, albino]. The color of the axolotl (e.g., 'wild',
↪  'leucistic', 'albino', etc.).", "gender": "string. One of: [male, female]. The gender of the axolotl
↪  ('male', 'female').", "size": "string. One of: [small, medium, large]. The size of the axolotl
↪  ('small', 'medium', 'large').", "page": "integer. The page number for pagination purposes."}
Output: Successful response.
 - Format: application/json
 - Structure: Object{results: Array[Object{url, source, description}], pagination: Object{current_page,
 ↪  total_pages, total_results}}
getAxolotlFacts: Retrieve interesting facts about axolotls such as their habits, habitats, and physical
↪  characteristics.
Parameters: {"category": "string. One of: [habits, habitat, physical characteristics]. The category of
↪  facts to retrieve (e.g., 'habits', 'habitat', 'physical characteristics').", "limit": "integer. The
↪  maximum number of facts to return."}
Output: Successful response.
 - Format: application/json
 - Structure: Array[Object{fact, source}]

Use the following format:
Thought: you should always think about what to do
Action: the action to take, should be one of the tool names.
Action Input: the input to the action, must be in JSON format. All of the action input must be realistic
↪  and from the user.

Begin!
Question: Hey, can you show me a random picture of an axolotl?
```

ZLogic

```
There are 6 houses, numbered 1 to 6 from left to right, as seen from across the street. Each house is
↪  occupied by a different person. Each house has a unique attribute for each of the following
↪  characteristics:
 - Each person has a unique name: `Arnold`, `Peter`, `Eric`, `Alice`, `Bob`, `Carol`
 - People have unique favorite book genres: `biography`, `science fiction`, `fantasy`, `mystery`,
 ↪  `romance`, `historical fiction`
 - People have unique favorite sports: `baseball`, `basketball`, `swimming`, `volleyball`, `tennis`,
 ↪  `soccer`
 - People own unique car models: `honda civic`, `ford f150`, `tesla model 3`, `chevrolet silverado`, `bmw 3
 ↪  series`, `toyota camry`

## Clues:
1. Eric is the person who loves mystery books.
2. The person who loves tennis is the person who loves fantasy books.
3. The person who loves soccer is directly left of the person who loves science fiction books.
4. There is one house between the person who owns a Honda Civic and the person who loves biography books.
5. Peter is somewhere to the right of Carol.
6. The person who loves tennis is in the first house.
7. The person who owns a Tesla Model 3 is somewhere to the right of the person who loves baseball.
8. Eric is somewhere to the left of the person who loves romance books.
9. The person who owns a Toyota Camry is somewhere to the right of the person who loves romance books.
10. The person who owns a BMW 3 Series is Peter.
11. The person who owns a BMW 3 Series is the person who loves basketball.
12. The person who owns a Tesla Model 3 is directly left of Arnold.
13. Alice and the person who loves volleyball are next to each other.
14. The person who loves historical fiction books is the person who loves soccer.
15. The person who owns a Chevrolet Silverado is not in the first house.
16. The person who loves science fiction books is directly left of the person who loves swimming.

What is Name of the person who lives in House 5?
A. Eric
B. Bob
C. Alice
D. Peter
E. Carol
F. Arnold
Please reason step by step, and put your final answer within \boxed{}.
```

D.3 Model Chat Templates

Qwen3-4B-Thinking-2507

```
<|im_start|>user\n{prompt}<|im_end|>
<|im_start|>assistant\n<think>
```

Qwen3-4B-Instruct-2507

```
<|im_start|>user\n{prompt}<|im_end|>
<|im_start|>assistant\n
```

OLMo-3-7B-Instruct

```
<|im_start|>system\nYou are a helpful function-calling AI assistant. You do not currently
↪  have access to any functions. <functions></functions><|im_end|>
<|im_start|>user\n{prompt}<|im_end|>
<|im_start|>assistant\n
```

OLMo-3-7B-Think

```
<|im_start|>system\nYou are OLMo, a helpful function-calling AI assistant built by Ai2.
↪  Your date cutoff is November 2024, and your model weights are available at
↪  https://huggingface.co/allenai. You do not currently have access to any functions.
↪  <functions></functions><|im_end|>
<|im_start|>user\n{prompt}<|im_end|>
<|im_start|>assistant\n<think>
```

D.4 Prompt Templates

Prompt for Math & AIME & Math500 & Science & GPQA

```
{question}
Please reason step by step, and put your final answer within \boxed{}.
```

Prompt for ToolUse & BFCLv4

```
Your task is to answer the user's question using available tools.
You have access to the following tools:
{tool}

Use the following format:
Thought: you should always think about what to do
Action: the action to take, should be one of the tool names.
Action Input: the input to the action, must be in JSON format. All of the action input must
↪  be realistic and from the user.

Begin!
Question: {question}
```

SDPO Regeneration Prompt

```
{question}
Please reason step by step, and put your final answer within \boxed{}.
Correct solution: {previously correct solution}
Correctly solve the original question.
```

E Supplementary Results

Table E.1 presents the performance scores used to plot Figure 2a, and displays a configuration matrix of EMA teachers $\alpha \in \{0, 0.5\%, 1\%, 2\%, 5\%\}$ and training datasets Math, ToolUse, and Science. Table E.2 extends Figure 4 for SDPO versus SDPO^{CoT} under more teacher EMA settings and evaluation benchmarks. The large degradation at $\alpha = 1\%, 5\%$ further verifies *confirmation bias*.

Table E.1: SDPO performance (Acc@8) against teacher EMA α. No α is globally best.

Trained on	$\alpha(\%)$	AIME	Math500	GPQA	MMLU-R	BFCLv4	LCBv6	Avg
–	–	32.71	73.85	62.94	86.02	76.25	46.73	63.08
MATH	0	56.42	84.29	58.72	**85.47**	77.71	48.52	68.61
	0.5	**57.92**	**84.52**	57.01	84.65	76.46	51.10	68.61
	1	56.04	84.35	58.21	85.31	77.71	**51.79**	**69.06**
	2	52.71	84.02	**59.53**	85.00	**78.75**	50.35	68.39
	5	34.38	78.97	55.87	83.82	78.12	50.86	63.67
TOOLUSE	0	**47.71**	**78.88**	60.73	**87.82**	76.04	**58.02**	**68.20**
	0.5	43.13	77.08	**61.49**	87.40	72.08	51.68	65.48
	1	41.04	77.18	58.78	87.18	76.04	48.62	64.81
	2	45.00	77.70	58.90	86.75	74.79	53.51	66.11
	5	**47.71**	78.57	57.39	86.25	**77.71**	55.55	67.20
SCIENCE	0	55.63	79.25	61.62	87.00	74.79	51.94	68.37
	0.5	53.96	79.10	61.05	**87.42**	75.21	52.93	68.28
	1	**55.84**	79.20	**62.31**	86.92	75.21	51.68	68.53
	2	55.00	**80.97**	59.47	86.40	**76.04**	54.92	**68.80**
	5	53.34	78.42	57.39	84.10	72.29	**56.11**	66.94

Table E.2: SDPO with and without CoT Performance (Acc@8). CoT enhances TOOLUSE training while impairing MATH and SCIENCE. Larger teacher EMA α generally worsens CoT performance.

$\alpha(\%)$	CoT	Trained on	AIME	Math500	GPQA	MMLU-R	BFCLv4	LCBv6
–	–	–	32.71	73.85	62.94	86.02	76.25	46.73
0	×	MATH	56.42	84.29	58.72	85.47	77.71	48.52
0	✓	MATH	38.13	75.58	58.59	86.22	66.46	56.82
1	×	MATH	56.04	84.89	58.62	85.31	77.71	51.79
1	✓	MATH	37.71	75.58	56.25	86.20	66.88	54.45
5	×	MATH	34.38	78.97	55.87	83.82	78.12	50.86
5	✓	MATH	8.13	69.50	40.53	80.95	34.38	41.76
0	×	SCIENCE	55.63	79.25	61.62	87.00	74.79	51.94
0	✓	SCIENCE	42.08	76.25	55.05	87.08	73.12	57.08
1	×	SCIENCE	55.84	79.20	62.31	86.92	75.21	51.68
1	✓	SCIENCE	37.29	75.60	55.11	86.65	71.67	60.46
5	×	SCIENCE	53.34	78.42	57.39	84.10	72.29	56.11
5	✓	SCIENCE	15.63	71.50	39.96	81.85	60.00	47.77
0	×	TOOLUSE	47.71	78.88	60.73	87.82	76.04	58.48
0	✓	TOOLUSE	53.34	82.28	56.69	84.38	80.62	56.66
1	×	TOOLUSE	41.04	77.18	58.78	87.18	76.04	49.69
1	✓	TOOLUSE	53.55	82.90	56.38	84.55	80.42	59.11
5	×	TOOLUSE	47.71	78.57	57.39	86.25	77.71	55.90
5	✓	TOOLUSE	53.34	83.00	50.06	83.30	77.08	52.50

Table E.5: $\text{SDPO}_{5\%}$ performance (Acc@8) continually trained with various data at 2nd & 3rd stage. Altering order only partially mitigates collapse; SDPO still degrades significantly under data shifts.

		MATH		SCIENCE		TOOLUSE		CODING		LOGIC	KNOW.
	Trained on	AIME	Math500	SCIENCE	GPQA	TOOLUSE	BFCLv4	CODING	LCBv6	ZLogic	MMLU-R
	–	32.71	73.85	50.54	62.94	48.90	76.25	56.39	46.73	81.62	86.02
$\text{SDPO}_{5\%}$	1. MATH	34.38	78.97	45.00	55.87	53.31	78.12	52.37	50.86	73.68	83.82
	2. SCIENCE	21.25	73.95	76.37	47.98	30.15	51.25	59.33	48.82	61.88	82.12
	3. TOOLUSE	0.00	0.00	0.18	0.13	0.00	0.00	0.00	0.00	0.10	0.12
	2. SCIENCE	21.25	73.95	76.37	47.98	30.15	51.25	59.33	48.82	61.88	82.12
	3. CODING	17.09	69.53	74.52	42.93	0.74	0.00	54.79	54.01	58.00	80.50
	2. TOOLUSE	22.29	73.82	38.63	42.36	12.68	9.79	–	–	56.95	77.35
	3. SCIENCE	10.42	66.50	72.56	36.55	13.60	20.62	–	–	49.20	77.75

Table E.3: Performance (Acc@8) trained on a single domain. Rows indicate training domain and method; columns indicate evaluation benchmarks. GRPO outperforms SDPO at generalization.

	Method	AIME	Math500	SCIENCE	GPQA	TOOLUSE	BFCLv4	ZLogic	MMLU-R
	Olmo-3-7B-Instruct-DPO								
	–	19.17	61.30	32.50	30.24	42.46	76.46	60.48	69.92
MATH	GRPO	18.96	64.82	35.24	35.23	40.62	71.88	64.95	72.00
MATH	SDPO_0	13.75	51.24	21.93	24.94	34.38	57.08	34.31	58.96
	Olmo-3-7B-Think-DPO								
	–	46.04	71.53	30.89	36.30	9.19	3.96	59.18	79.35
MATH	GRPO	51.67	74.95	37.56	44.32	9.38	3.33	61.58	79.47
MATH	$\text{SDPO}_{5\%}$	41.67	72.90	28.15	33.71	8.09	0.00	43.88	72.20
MATH	SDPO_0	51.25	75.95	32.98	38.45	8.46	0.62	54.40	79.00
MATH	$\text{SDPO}_0^{\text{CoT}}$	54.59	74.52	41.79	43.81	7.17	0.00	61.78	78.53

Table E.4: Representative gain, retention, and forgetting cases. Intermediate-distance targets exhibit the clearest forgetting, while source-like and very distant targets are often stable or improve.

	Train	Test	MMD·10^{-2}	$\Delta_{\text{SDPO}_{5\%}}$	Δ_{SDPO_0}	Gain	Retention	Forgetting
Near	MATH	AIME	0.64	↑ 1.67	↑ 23.71	✓		
Near	MATH	Math500	1.38	↑ 5.12	↑ 10.44	✓		
Near	CODING	LCBv6	3.74	↑ 11.74	↑ 13.39	✓		
Near	TOOLUSE	BFCLv4	4.75	↑ 1.46	↓ 0.21		✓	
Intermediate	MATH	GPQA	6.74	↓ 7.07	↓ 4.22			✓
Intermediate	SCIENCE	GPQA	11.42	↓ 5.55	↓ 1.32			✓
Intermediate	TOOLUSE	GPQA	18.89	↓ 5.55	↓ 2.21			✓
Intermediate	CODING	GPQA	11.90	↓ 4.61	↓ 2.65			✓
Intermediate	MATH	ZLogic	21.98	↓ 7.94	↓ 7.12			✓
Intermediate	SCIENCE	BFCLv4	25.16	↓ 3.96	↓ 1.46			✓
Distant	SCIENCE	ZLogic	32.66	↑ 9.06	↑ 8.18	✓		
Distant	TOOLUSE	ZLogic	43.33	↑ 2.40	↑ 4.86	✓		
Distant	CODING	ZLogic	31.66	↓ 1.09	↑ 0.96		✓	

For single-domain training, Table E.3 examines SDPO and GRPO for Olmo-3 series. For continual training, Table E.7 presents detailed scores for Figure 6 across each stage, with a data-ordering ablation on collapsed $\text{SDPO}_{5\%}$ in Table E.5. We supplement diagnostics (Figure 7) with additional matrices (Figure E.2), more training (Figure E.3), and more benchmarks (Figure E.4).

We also identify one extreme failure mode as artifact amplification in self-distillation. $\text{SDPO}_{5\%}$ collapses with a visible generation artifact: endlessly repeated "\boxed" tokens. This behavior coincides with a change in final-answer formatting requirement at the third stage, where TOOLUSE differs from earlier MATH and SCIENCE stages (data samples in Appendix D.2). The collapsed model still over-amplifies the high-frequency formatting pattern from earlier training stages. More experiments show that this collapse can be partially mitigated by reordering data or masking tokens (Tables E.5 and E.7).

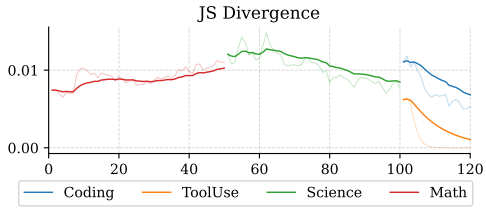


Figure E.1: JS divergence during continual-$\text{SDPO}_{5\%}$ with TOOLUSE or CODING as third stage.

This failure can be attributed to the token supervision: frequent tokens can receive repeated reinforcement. As the teacher is derived from the student, artifacts can enter a self-reinforcing loop (*confirmation bias*): the student emits the artifact token, the teacher assigns it token-level support, and the next student update makes it more likely, explaining the JS divergence plummeting in the

Table E.6: Performance (Acc@8) comparison of linear and spectral weight editing for mitigating forgetting in SDPO$_0$. @s denotes the scale of update between the post-trained model and pre-update baseline. r denotes the top-r singular components to retain, which takes $r = 64$ unless otherwise specified.

	MATH		SCIENCE		TOOLUSE		CODING		LOGIC	KNOW.	Avg
Method	AIME	Math500	SCIENCE	GPQA	TOOLUSE	BFCLv4	CODING	LCBv6	ZLogic	MMLU-R	
Baseline	32.71	73.85	50.54	62.94	48.90	76.25	56.39	46.73	81.62	86.02	61.60
MATH-GRPO	44.67	76.84	51.33	62.83	50.48	78.33	60.73	52.93	80.82	85.74	64.47
MATH-SDPO$_0$	56.42	84.29	47.36	58.72	52.06	77.71	57.58	48.52	74.50	85.47	64.26
+Linear@0.3	48.34	81.42	50.71	63.38	49.45	75.00	56.71	44.01	79.20	85.65	63.39
+Linear@0.5$_{\lambda=3e-5}$	44.80	80.82	49.64	61.93	49.26	77.71	56.55	44.94	79.97	85.72	63.13
+Linear@0.5	51.67	84.25	48.81	62.25	50.92	77.29	56.47	46.65	76.42	85.65	64.04
+Linear@0.7	55.00	84.82	46.07	59.41	52.57	76.67	57.59	48.36	72.10	84.12	63.67
+Linear@1$_{\lambda=3e-5}$	55.63	84.90	47.86	60.10	50.55	77.08	56.81	45.98	75.32	85.60	63.98
+Linear@1$_{\lambda=5e-5}$	56.67	84.58	44.23	57.20	51.84	77.92	56.75	47.88	72.62	84.35	63.40
+Spectral@0.3	41.46	79.10	50.77	62.18	48.71	74.58	55.79	45.25	80.88	85.95	62.47
+Spectral@0.5$_{r=16}$	50.21	82.05	52.26	61.62	48.90	75.21	57.95	46.61	79.38	86.40	64.06
+Spectral@0.5$_{r=256}$	50.42	82.65	49.64	61.87	49.45	77.50	56.24	44.25	78.70	85.98	63.67
+Spectral@0.5	49.59	81.88	48.81	60.54	49.63	76.25	57.44	45.35	78.72	85.65	63.39
+Spectral@0.7	54.17	83.15	49.70	60.10	50.74	76.67	57.45	47.72	76.60	86.18	64.25
+Spectral@1$_{\lambda=3e-5}$	49.38	82.18	49.58	62.88	49.45	75.42	56.46	47.01	77.88	86.38	63.66

third stage (Figure E.1). A frequently refreshed teacher or denser supervision can worsen this process and, in severe cases, can lead to near-complete collapse.

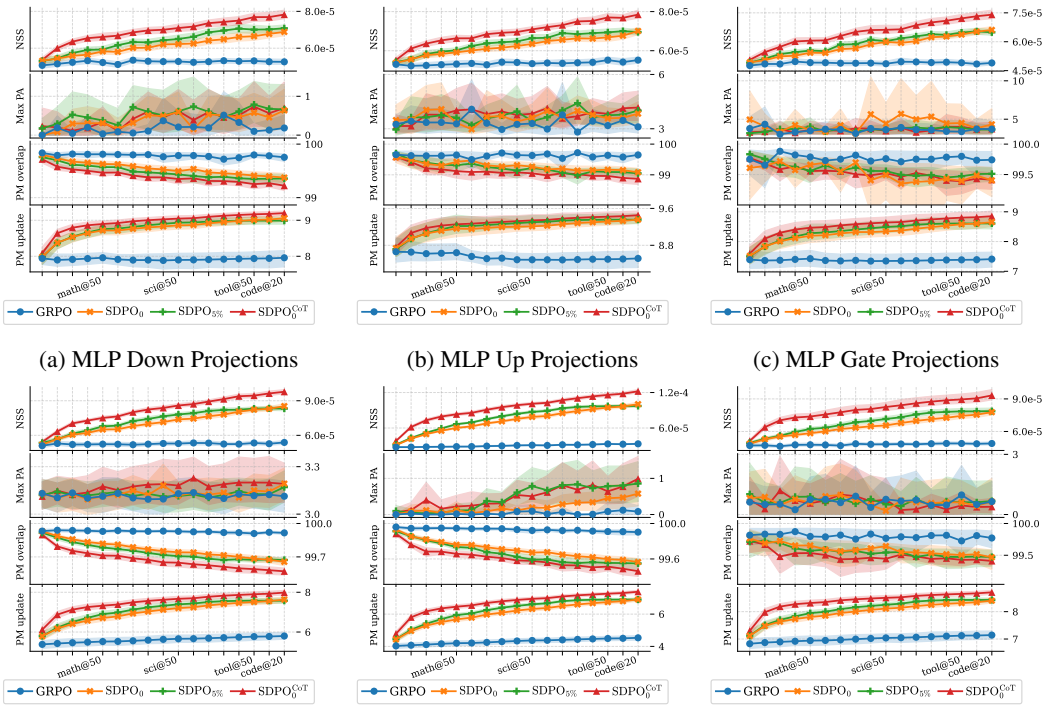
In weight editing, we merge a post-trained checkpoint back into its base model through a task-vector formulation designed to mitigate forgetting. For each weight tensor, let W_0 and W_1 denote the base and post-trained weights and $\Delta = W_1 - W_0$ the task vector; the merged weight takes the form $W = W_0 + \lambda\Delta$, where the effective per-tensor scale λ composes three mechanisms. First, a global scale $s \in [0, 1]$ interpolates between the base (s=0) and post-trained (s=1) endpoints in the spirit of WiSE-FT, trading in-domain gains against out-of-distribution (OOD) retention. Second, in the *spectral* variant, the task vector is truncated to its top-r singular components, $\Delta_r = U_{:r}S_{:r}V_{:r}^\top$ with $\Delta = USV^\top$, before scaling: the discarded spectral tail is hypothesized to carry noisy, overfit directions that drive OOD forgetting, whereas the principal subspace preserves most in-domain gains. Third, each task vector is optionally anchored to a spectral-drift budget: with

$$d_t = \frac{\left\|\boldsymbol{\sigma}(W_1) - \boldsymbol{\sigma}(W_0)\right\|_2}{\left\|\boldsymbol{\sigma}(W_0)\right\|_2}$$

the normalized drift of the singular-value spectrum, a runtime-measurable proxy for posttraining-induced forgetting, the per-tensor scale $\lambda_t = \min(1, \tau/d_t)$ shrinks over-drifted tensors to the tolerated level τ while leaving within-budget tensors untouched; in the first-order regime of observed drifts ($d_t \sim 10^{-5}$), $d(W_0, W_0 + \lambda\Delta) \approx \lambda\, d_t$, so the anchor is exact up to $O(d_t^2)$. An additional relative-norm guard $\|\lambda\Delta\|_F \le g\,\|W_0\|_F$ caps the per-tensor update magnitude. All computations are carried out in float32 per tensor with tied weights merged once; one-dimensional tensors (biases, normalization parameters), for which the drift metric is undefined, are exempt from anchoring, and the choice $r \ge \min(m, n)$ recovers plain linear interpolation.

Table E.7: SDPO versus GRPO performance (Acc@8) continually trained. SDPO_0 and $\text{SDPO}_{5\%}$ denote SDPO with teacher EMA 0 and 5%, respectively. $\text{SDPO}_0^{\text{CoT}}$ denotes SDPO_0 with CoT distilled. $\text{SDPO}_{5\%}^{\text{M25}}$ denotes $\text{SDPO}_{5\%}$ randomly masked out 25% tokens. S-SDPO_0 denotes SDPO with restart-and-freeze teacher strategy ($\gamma = 1$).

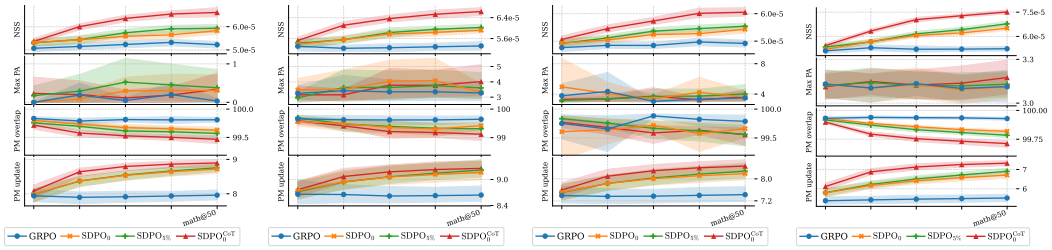
		MATH		SCIENCE		TOOLUSE		CODING		LOGIC	KNOW.	Avg
	Trained on	AIME	Math500	SCIENCE	GPQA	TOOLUSE	BFCLv4	CODING	LCBv6	ZLogic	MMLU-R	
	Qwen3-4B-Thinking-2507											
	–	32.71	73.85	50.54	62.94	48.90	76.25	56.39	46.73	81.62	86.02	61.60
GRPO	1. MATH	44.67	76.84	51.33	62.83	50.48	78.33	60.73	52.93	80.82	85.74	64.47
GRPO	2. SCIENCE	54.88	78.28	56.39	64.23	52.06	76.67	60.06	50.17	88.23	87.91	66.89
GRPO	3. TOOLUSE	56.21	79.49	60.64	63.99	58.71	80.83	60.49	52.23	90.10	88.54	69.12
GRPO	4. CODING	57.29	79.88	61.85	64.58	57.72	81.25	64.68	53.56	89.52	88.70	69.90
SDPO_0	1. MATH	56.42	84.29	47.36	58.72	52.06	77.71	57.58	48.52	74.50	85.47	64.26
SDPO_0	2. SCIENCE	49.29	81.75	65.70	59.03	48.49	76.67	58.35	56.88	87.17	86.38	66.97
SDPO_0	3. TOOLUSE	41.13	80.02	70.37	54.71	37.35	30.83	56.39	54.25	80.03	84.92	59.00
SDPO_0	4. CODING	34.38	77.40	70.71	51.83	9.93	11.25	61.77	51.89	72.15	84.20	52.55
$\text{SDPO}_0^{\text{CoT}}$	1. MATH	38.13	75.58	49.88	58.59	48.35	66.46	52.70	56.82	77.00	86.22	60.97
$\text{SDPO}_0^{\text{CoT}}$	2. SCIENCE	27.71	74.65	70.89	54.10	42.83	61.67	47.28	46.28	76.65	85.95	58.80
$\text{SDPO}_0^{\text{CoT}}$	3. TOOLUSE	21.04	71.88	70.71	49.37	38.60	36.04	41.94	44.88	66.22	84.38	52.51
$\text{SDPO}_0^{\text{CoT}}$	4. CODING	14.17	69.53	68.69	43.24	23.35	13.54	30.01	24.32	59.95	81.62	42.84
$\text{SDPO}_{5\%}$	1. MATH	34.38	78.97	45.00	55.87	53.31	78.12	52.37	50.86	73.68	83.82	60.64
$\text{SDPO}_{5\%}$	2. SCIENCE	21.25	73.95	76.37	47.98	30.15	51.25	59.33	48.82	61.88	82.12	55.31
$\text{SDPO}_{5\%}$	3. TOOLUSE	0.00	0.00	0.18	0.13	0.00	0.00	0.00	0.00	0.10	0.12	0.05
$\text{SDPO}_{5\%}$	4. CODING	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.18	0.02
$\text{SDPO}_{5\%}^{\text{M25}}$	1. MATH	37.09	79.88	42.32	56.12	52.39	74.58	54.71	51.51	71.80	84.35	60.48
$\text{SDPO}_{5\%}^{\text{M25}}$	2. SCIENCE	18.54	72.38	72.32	45.08	23.16	23.96	57.73	48.75	54.12	81.90	49.79
$\text{SDPO}_{5\%}^{\text{M25}}$	3. TOOLUSE	0.00	0.00	5.42	1.96	0.00	0.00	0.00	0.00	1.50	9.65	1.85
$\text{SDPO}_{5\%}^{\text{M50}}$	1. MATH	39.17	78.53	43.99	55.81	50.00	76.67	54.01	51.75	75.20	84.72	60.98
$\text{SDPO}_{5\%}^{\text{M50}}$	2. SCIENCE	25.84	76.10	77.98	51.52	35.48	44.58	48.18	49.56	72.05	82.98	56.43
$\text{SDPO}_{5\%}^{\text{M50}}$	3. TOOLUSE	0.00	0.12	34.17	18.62	0.00	0.00	0.09	0.00	55.47	43.38	15.18
$\text{SDPO}_{5\%}^{\text{M75}}$	1. MATH	37.50	79.05	43.15	56.12	53.68	80.00	53.53	51.93	75.22	84.85	61.50
$\text{SDPO}_{5\%}^{\text{M75}}$	2. SCIENCE	18.55	73.05	70.95	44.00	42.83	63.75	59.02	50.39	56.53	81.73	55.31
$\text{SDPO}_{5\%}^{\text{M75}}$	3. TOOLUSE	0.63	28.48	54.70	28.22	0.00	0.00	8.18	7.31	45.55	62.02	23.51
	Qwen3-4B-Instruct-2507											
	–	50.00	77.10	24.23	14.71	59.01	83.12	–	–	48.00	56.75	51.61
GRPO	1. MATH	56.00	79.94	25.94	17.23	58.97	83.33	–	–	49.40	57.60	53.55
GRPO	2. SCIENCE	56.88	81.12	59.35	58.84	57.72	82.92	–	–	84.22	86.52	70.95
GRPO	3. TOOLUSE	58.34	81.88	60.54	59.91	65.99	75.42	–	–	84.80	87.85	71.84
SDPO_0	1. MATH	35.84	71.15	12.20	5.49	59.74	82.29	–	–	9.15	2.97	34.85
SDPO_0	2. SCIENCE	27.08	64.53	11.79	8.90	57.90	82.92	–	–	11.70	9.48	34.29
SDPO_0	3. TOOLUSE	20.63	54.25	7.14	4.17	0.00	0.00	–	–	1.48	5.35	11.63
SFT	1. MATH	0.42	20.78	27.38	23.99	52.94	78.12	–	–	2.58	66.55	34.09
SFT	2. SCIENCE	0.00	0.00	72.74	30.74	0.00	7.92	–	–	30.22	69.30	26.36
SFT	3. TOOLUSE	0.00	0.00	0.00	0.00	0.00	0.00	–	–	0.00	0.00	0.00
	Olmo-3-7B-Think											
	–	42.29	72.55	31.90	34.09	18.01	27.71	50.79	39.64	63.10	80.20	46.03
GRPO	1. MATH	52.08	74.50	36.01	39.96	14.71	20.62	55.39	43.25	63.18	80.53	48.02
GRPO	2. SCIENCE	52.30	75.38	49.88	42.42	13.05	23.12	56.92	45.66	62.22	81.25	50.22
GRPO	3. TOOLUSE	51.88	74.85	53.04	43.75	52.57	74.79	59.45	46.10	63.82	80.70	60.09
GRPO	4. CODING	54.17	75.60	53.51	44.26	50.18	71.67	60.88	45.16	63.92	79.50	59.88
SDPO_0	1. MATH	52.08	75.12	34.76	37.50	10.29	25.42	52.51	41.90	56.92	78.25	46.48
SDPO_0	2. SCIENCE	43.96	74.50	57.38	35.54	9.38	11.67	56.96	45.57	57.18	76.40	46.85
SDPO_0	3. TOOLUSE	41.25	72.38	52.56	34.47	16.73	30.83	59.53	46.08	52.20	74.68	48.07
SDPO_0	4. CODING	37.29	71.60	53.39	32.83	14.89	20.21	60.95	44.98	46.62	73.40	45.62



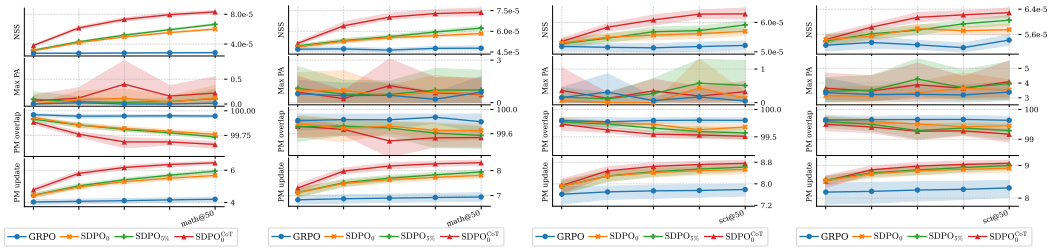
(a) MLP Down Projections (b) MLP Up Projections (c) MLP Gate Projections

(d) Self-Attention Qs (e) Self-Attention Ks (f) Self-Attention Os

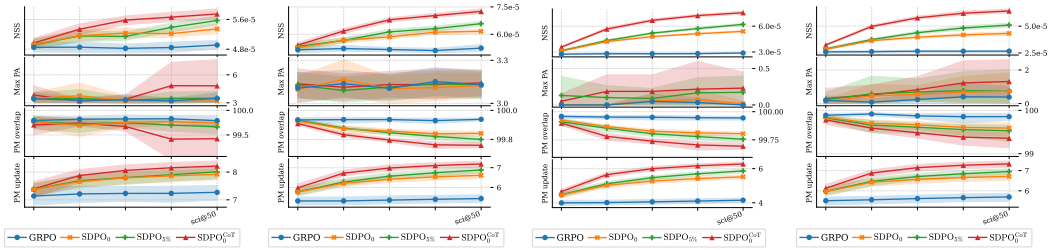
Figure E.2: Parameter drift (y-axis) against training steps (x-axis), with metrics averaged across all layers with a 95% confidence interval. **(a-f)** MATH training. **(g-l)** continual training.



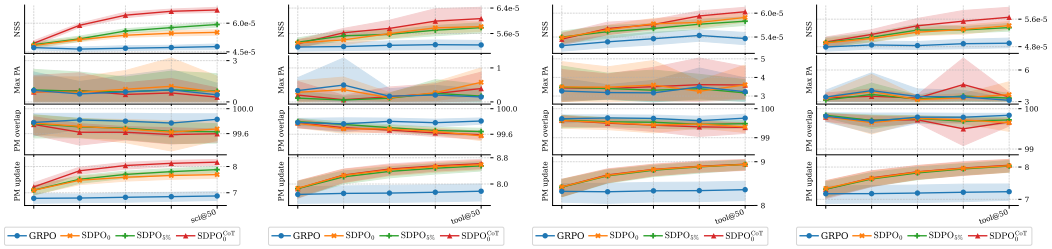
(a) MLP Down Projections (b) MLP Up Projections (c) MLP Gate Projections (d) Self-Attention Qs



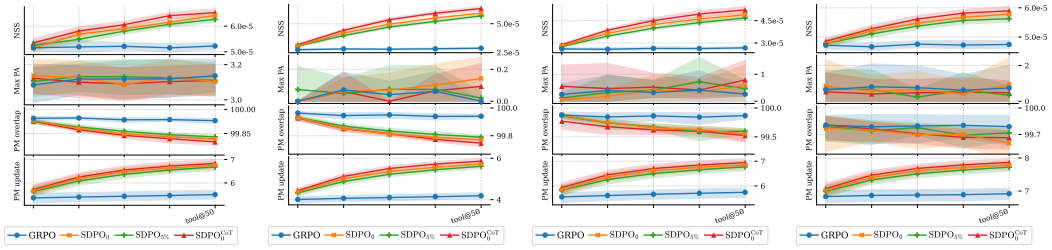
(e) Self-Attention Ks (f) Self-Attention Os (g) SCIENCE MLP Down (h) SCIENCE MLP Up



(i) SCIENCE MLP Gate (j) SCIENCE Self-Attn Qs (k) SCIENCE Self-Attn Ks (l) SCIENCE Self-Attn Vs

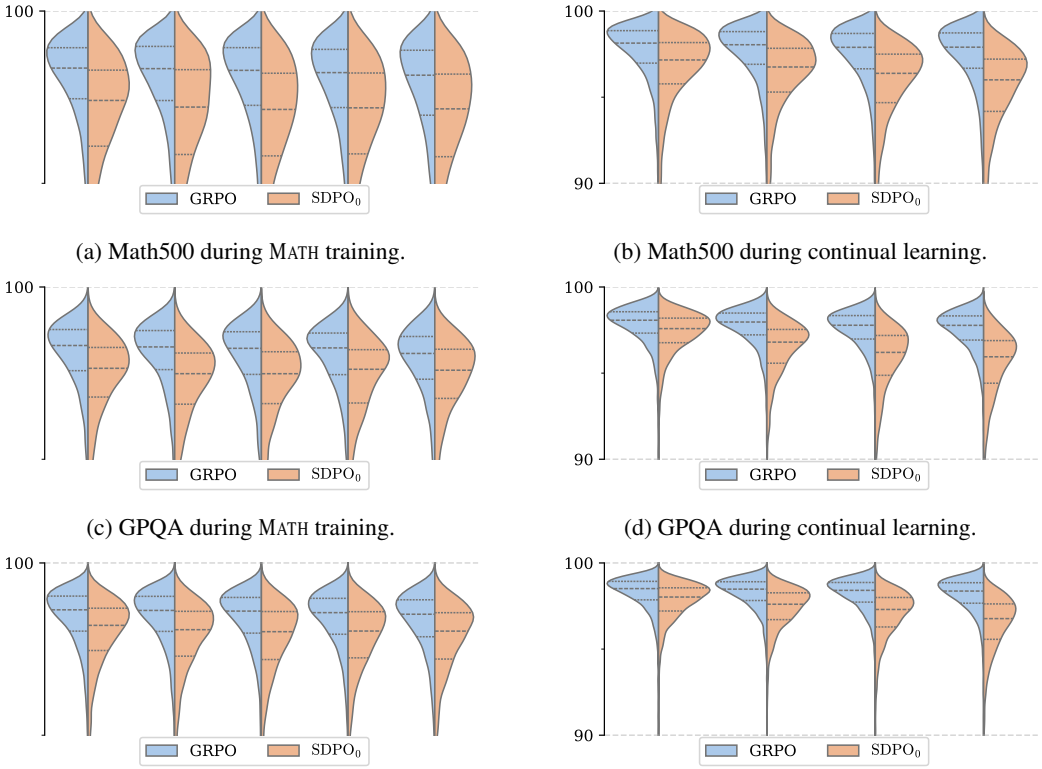


(m) SCIENCE Self-Attn Os (n) TOOLUSE MLP Down (o) TOOLUSE MLP Up (p) TOOLUSE MLP Gate



(q) TOOLUSE Self-Attn Qs (r) TOOLUSE Self-Attn Ks (s) TOOLUSE Self-Attn Vs (t) TOOLUSE Self-Attn Os

Figure E.3: Parameter drift (y-axis) against MATH, SCIENCE, and TOOLUSE training steps (x-axis, $t = 10, 20, \ldots, 50$), with metrics averaged across all layers with a 95% confidence interval.



(a) Math500 during MATH training.

(b) Math500 during continual learning.

(c) GPQA during MATH training.

(d) GPQA during continual learning.

(e) MMLU-R during MATH training.

(f) MMLU-R during continual learning.

Figure E.4: Cosine similarity (%) between rollout(@8) embeddings before and after GRPO (in blue) versus SDPO$_0$ (in orange) training. For MATH training, x-axis denote checkpoints at steps $k = 10, 20, \ldots, 50$ from left to right; for continual learning, x-axis denote checkpoints after tasks MATH, SCIENCE, TOOLUSE, and CODING from left to right.

F GENERATION SAMPLES

AIME Qwen3-4B-Thinking-2507 Generation

```
This is a complex or challenging question, and it is difficult to provide a direct and correct answer. I need to think about it.\nWell, let's tackle this problem step by step. So, first, let's make sure I understand what's given and what's being asked. Aya does a 9 km walk every morning, but wait, hold on\u2014wait, no, wait: "a 9-kilometer-long walk"\u2014does that mean the walk itself is 9 km, not including the coffee shop? Wait, the problem says "the walk takes her 4 hours, including t minutes spent in the coffee shop." Hmm, okay, so the "walk" here is the time from when she starts walking until she gets to the coffee shop? Wait, no, maybe better to parse the problem carefully.

"Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards." So, the walk is 9 km long, then she stops at a coffee shop (so the coffee shop is the endpoint of the walk? Maybe the walk is the 9 km journey to the coffee shop, and then she spends t minutes there? Wait, but the problem says "the walk takes her 4 hours, including t minutes spent in the coffee shop." Wait, that's confusing. If the walk is the 9 km, then why is the time including time spent in the coffee shop? Maybe the problem is using "walk" to mean the entire activity? Wait, no, let's check the problem statement again:

"When she walks at a constant speed of s kilometers per hour, the walk takes her 4 hours, including t minutes spent in the coffee shop." Oh! Wait, maybe "the walk" here is a misnomer, but actually, the total time from when she starts walking until she finishes at the coffee shop (i.e., the time to walk the 9 km plus the time she spends at the coffee shop) is 4 hours? Wait, but the problem says "the walk takes her 4 hours, including t minutes spent in the coffee shop." Hmm, maybe the problem is translated or phrased a bit awkwardly, but let's think in terms of variables.

Let's define:

Let\u2019s let the time she spends walking (i.e., the time actually moving) be W hours, and the time spent at the coffee shop be T hours? Wait, no, the problem says t minutes, so maybe better to use minutes for the coffee shop time to avoid decimals, but maybe keep it in hours first.

Wait, let's try to set up equations based on the two scenarios.

First scenario: speed = s km/h, total time (walk + coffee shop) = 4 hours, which includes t minutes at coffee shop. Wait, but hold on\u2014if the walk is 9 km long, then the walking time should be distance over speed, right? So maybe the "walk" here is the 9 km, so the time she spends walking (moving) is 9/s hours, and then she spends t minutes at the coffee shop, so total time from start to finish (including coffee shop) is (9/s) + (t/60) hours. The problem says "the walk takes her 4 hours, including t minutes spent in the coffee shop." Wait, maybe the problem is using "walk" to mean the entire trip? Like, "the walk" is the activity from when she starts walking until she stops at the coffee shop, so the duration of that activity is 4 hours, which consists of the time walking (9 km) plus time at coffee shop (t minutes). That might be the right interpretation.

Yes, that makes sense. So "the walk takes her 4 hours"\u2014the entire walk activity (which includes walking the 9 km and then stopping at coffee shop) takes 4 hours total, with t minutes of that time spent at the coffee shop. So the rest of the time (4 hours minus t minutes) is spent actually walking the 9 km.

Similarly, second scenario: speed = s + 2 km/h, total walk activity time = 2 hours 24 minutes, which includes t minutes at coffee shop, so walking time is 2h24m - t minutes.

Okay, that seems to be the key interpretation. Let's formalize this.

Let me convert all times to hours to make the equations easier, or maybe keep t in minutes and convert the total times to minutes? Let's see. Let's try hours first, then convert t to hours.

First scenario:

Total time (including coffee shop) = 4 hours = 240 minutes.

Time spent at coffee shop = t minutes = t/60 hours.

Therefore, time spent walking = total time - coffee shop time = 240 - t minutes = (240 - t)/60 hours.

But she walks 9 km at speed s km/h, so walking time = 9 / s hours.

Therefore, we have equation 1: 9 / s = (240 - t)/60.

Second scenario:

Total time (including coffee shop) = 2 hours 24 minutes = 2 + 24/60 = 2.4 hours = 144 minutes.

Time spent at coffee shop = still t minutes (the problem says "including t minutes spent in the coffee shop" in both cases, so t is the same in both scenarios\u2014she spends t minutes at coffee shop regardless of walking speed, which is good, that's a constant we need to find or relate).

Therefore, time spent walking = total time - coffee shop time = 144 - t minutes = (144 - t)/60 hours.

She walks 9 km at speed s + 2 km/h, so walking time = 9 / (s + 2) hours.

Therefore, equation 2: 9 / (s + 2) = (144 - t)/60.

Okay, great! So now we have two equations with two variables: s (km/h) and t (minutes). We need to solve for s and t, then find the total time when speed is s + 1/2 km/h, which would be (walking time) + t minutes, so total time in minutes is (9 / (s + 1/2) * 60) + t, right? Wait, let's confirm: walking time at speed s + 1/2 is 9 / (s + 1/2) hours, which is 60*(9 / (s + 1/2)) minutes, then add t minutes for coffee shop, so total time in minutes is 540 / (s + 1/2) + t. Maybe better to keep everything in minutes to avoid confusion later.
```

```
Let's write equations 1 and 2 again, but let's multiply both sides by 60 to eliminate denominators, that
↪  might make it cleaner.

Equation 1: 9 / s = (240 - t)/60 \u21d2 multiply both sides by 60s: 540 = s(240 - t) \u21d2 540 = 240s -
↪  st. Let's call this Eq1.

Equation 2: 9 / (s + 2) = (144 - t)/60 \u21d2 multiply both sides by 60(s + 2): 540 = (s + 2)(144 - t).
↪  Let's call this Eq2.

Okay, now we have two linear equations in terms of s and t? Wait, no, actually, Eq1 and Eq2 are both
↪  linear in s and t, yes! Let's check:

Eq1: 240s - st = 540 \u21d2 s(240 - t) = 540

Eq2: (s + 2)(144 - t) = 540

Oh, nice! Both equal 540, so we can set them equal to each other to eliminate the 540.

So, s(240 - t) = (s + 2)(144 - t)

Let's expand the right-hand side: s(144 - t) + 2(144 - t) = 144s - st + 288 - 2t

Left-hand side: 240s - st

So set equal: 240s - st = 144s - st + 288 - 2t

Wait, look, we can subtract (-st) from both sides? Wait, let's subtract 144s - st from both sides to
↪  simplify.

Left side minus (144s - st) = (240s - st) - 144s + st = 96s

Right side minus (144s - st) = (144s - st + 288 - 2t) - 144s + st = 288 - 2t

So we get 96s = 288 - 2t

Oh, that's a nice simplification! The st terms canceled out, which is good, we didn't want to deal with
↪  quadratic equations. Let's write that as:

96s = 288 - 2t \u21d2 let's divide both sides by 2 to simplify: 48s = 144 - t \u21d2 t = 144 - 48s. Let's
↪  call this Eq3.

Okay, great, now we can plug Eq3 into one of the original equations to solve for s. Let's pick Eq1: s(240 -
↪  t) = 540.

First, compute 240 - t using Eq3: 240 - (144 - 48s) = 240 - 144 + 48s = 96 + 48s.

So Eq1 becomes: s(96 + 48s) = 540.

Wait, hold on: is that right? Wait, Eq1 was s(240 - t) = 540, yes, and we found 240 - t = 96 + 48s, so
↪  yes, s*(96 + 48s) = 540.

Let's write that as 48s\u00b2 + 96s - 540 = 0. Maybe we can simplify this equation by dividing all terms
↪  by 24? Let's see: 48/24=2, 96/24=4, 540/24=22.5, which is not integer. Divide by 12: 4s\u00b2 + 8s -
↪  45 = 0. Wait, 48\u00f712=4, 96\u00f712=8, 540\u00f712=45, yes! Perfect, that's integer coefficients.

So 4s\u00b2 + 8s - 45 = 0. Now let's solve for s using quadratic formula.

Quadratic in standard form: as\u00b2 + bs + c = 0, solutions s = [-b \u00b1 \u221a(b\u00b2 - 4ac)]/(2a)

Here, a=4, b=8, c=-45.

Discriminant D = b\u00b2 - 4ac = 64 - 4*4*(-45) = 64 + 720 = 784. Oh, nice, 784 is a perfect square!
↪  \u221a784 = 28.

So s = [-8 \u00b1 28]/(8). Now, speed can't be negative, so we take the positive root: (-8 + 28)/8 = 20/8
↪  = 5/2 = 2.5 km/h? Wait, hold on, wait: 20 divided by 8 is 2.5, yes, that's 2.5 km/h. Wait, but let's
↪  check if that makes sense with the times.

Wait, but hold on, let's make sure we didn't make a mistake in setting up the equations, because if s is
↪  2.5 km/h, then walking time for 9 km is 9 / 2.5 = 3.6 hours = 216 minutes, so total time including
↪  coffee shop is 216 + t minutes = 4 hours = 240 minutes, so t = 24 minutes. Let's check the second
↪  scenario: speed s + 2 = 4.5 km/h, walking time = 9 / 4.5 = 2 hours = 120 minutes, total time including
↪  coffee shop is 120 + t = 120 + 24 = 144 minutes = 2 hours 24 minutes, which matches the problem
↪  statement! Oh, that's perfect! So that checks out.

Wait, so why did I doubt? Let's go through the algebra again to make sure I didn't skip a step wrong.

We had Eq1: s(240 - t) = 540 (from 9/s = (240 - t)/60 \u21d2 9*60 = s(240 - t) \u21d2 540 = s(240 - t),
↪  yes, that's correct, 9*60 is 540, right).

Eq2: (s + 2)(144 - t) = 540, same way: 9/(s+2) = (144 - t)/60 \u21d2 9*60 = (s+2)(144 - t) \u21d2 540 =
↪  (s+2)(144 - t), correct.

Set equal: s(240 - t) = (s + 2)(144 - t). Expanded RHS: s(144 - t) + 2(144 - t) = 144s - st + 288 - 2t.
↪  LHS: 240s - st. Subtract 144s - st from both sides: 96s = 288 - 2t \u21d2 48s = 144 - t \u21d2 t = 144
↪  - 48s, correct.

Then plug into Eq1: s(240 - (144 - 48s)) = s(96 + 48s) = 540 \u21d2 48s\u00b2 + 96s - 540 = 0 \u21d2
↪  divide by 12: 4s\u00b2 + 8s - 45 = 0, correct.

Quadratic formula: s = [-8 \u00b1 \u221a(64 + 720)]/8 = [-8 \u00b1 \u221a784]/8 = [-8 \u00b1 28]/8.
↪  Positive solution: 20/8 = 2.5, yes, 2.5 km/h. Then t = 144 - 48s = 144 - 48*(2.5) = 144 - 120 = 24
↪  minutes. Perfect, matches the check we did earlier.
```

```
Wait, so in the check, when s=2.5, walking time is 9/2.5=3.6 hours=216 minutes, total time=216 +24=240
↪  minutes=4 hours, correct. Second scenario: s+2=4.5, walking time=9/4.5=2 hours=120 minutes, total
↪  time=120+24=144 minutes=2h24m, correct. So all checks out.

Now, the problem asks: Suppose Aya walks at s + 1/2 kilometers per hour. Find the number of minutes the
↪  walk takes her, including the t minutes spent in the coffee shop.

So first, s + 1/2 = 2.5 + 0.5 = 3 km/h. Wait, that's nice! 2.5 is 5/2, so 5/2 + 1/2 = 6/2 = 3 km/h. Good,
↪  so speed is 3 km/h.

Now, walking time at 3 km/h for 9 km: 9 / 3 = 3 hours. Then add t minutes for coffee shop, which we found
↪  t=24 minutes.

Wait, hold on! Wait, the problem says "the number of minutes the walk takes her, including the t minutes
↪  spent in the coffee shop." So total time in minutes is (walking time in hours * 60) + t.

Walking time at 3 km/h: 9 km / 3 km/h = 3 hours = 180 minutes. Then add t=24 minutes, so total time=180
↪  +24=204 minutes.

Wait, is that it? Let me make sure we didn't misinterpret "the walk takes her...". Wait, in the problem
↪  statement, when she walks at s km/h, the walk takes 4 hours including t minutes at coffee shop. So in
↪  our interpretation, "the walk" (the activity) takes total time = walking time + coffee time. So yes,
↪  for the new speed, it's walking time (at s+1/2) + t minutes, which is what we did.

But let's just go through the problem statement again to be 100% sure we didn't flip something.

"Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards." So the walk is
↪  9 km, then she stops at coffee shop (so coffee shop is after the walk? Wait, maybe that's the
↪  confusion! Maybe "the walk" is 9 km, then she goes to coffee shop, so the time to walk the 9 km is
↪  separate from the time at coffee shop, and the problem says "the walk takes her 4 hours, including t
↪  minutes spent in the coffee shop"\u2014wait, that would be odd, because if the walk is the 9 km, it
↪  shouldn't include coffee shop time. Wait, but in our check, it worked out perfectly, so maybe the
↪  problem statement is a bit poorly worded, but the math checks out with the interpretation that total
↪  time from start of walk (when she begins walking) until she finishes at coffee shop is 4 hours (which
↪  includes the time she's walking the 9 km plus time at coffee shop).

Wait, another way: suppose "the walk" refers to the 9 km journey, so time to walk 9 km is some time, then
↪  she spends t minutes at coffee shop, so total time from when she starts walking until she leaves
↪  coffee shop is (walking time) + t minutes. The problem says "the walk takes her 4 hours, including t
↪  minutes spent in the coffee shop"\u2014maybe "takes her" is the duration of the entire activity
↪  (walking + coffee), so yes, that's how we took it, and the check worked, so probably that's the right
↪  interpretation.

Let's see, if we had misinterpreted and thought that the coffee shop time is part of the walk time, but
↪  no, the problem says "including t minutes spent in the coffee shop", so the t minutes are added to the
↪  walk time to get the total time.

Wait, in our check, with s=2.5 km/h, walking time (just walking) is 3.6 hours=216 minutes, total time with
↪  coffee is 240 minutes, so t=24 minutes. Second scenario: walking time 2 hours=120 minutes, total time
↪  144 minutes, so t=24 minutes, same t, which the problem states, so that's consistent.

So when speed is 3 km/h, walking time is 9/3=3 hours=180 minutes, total time=180 +24=204 minutes.

Wait, but let's try to do it without plugging in numbers early, to verify.

We had s=2.5, t=24, so s + 1/2=3, walking time=9/3=3 hours=180 min, total=180+24=204.

Alternatively, let's see if we can find the total time without finding s and t individually, maybe as a
↪  check.

Let\u2019s denote for a given speed v (km/h), walking time is 9/v hours, total time T(v) = 9/v + t/60
↪  hours (since t minutes is t/60 hours).

In first case, v=s, T(s)=4 hours \u21d2 9/s + t/60 = 4 \u21d2 9/s = 4 - t/60 \u21d2 which is what we had
↪  (since 4 - t/60 = (240 - t)/60 hours).

Second case, v=s+2, T(s+2)=2 + 24/60=2.4 hours \u21d2 9/(s+2) + t/60=2.4 \u21d2 9/(s+2)=2.4 - t/60=(144 -
↪  t)/60, same as before.

So we can write the two equations as:

9/s = 4 - t/60 ...(1a)

9/(s+2) = 2.4 - t/60 ...(2a)

Let\u2019s let u = t/60, so u is time in hours at coffee shop. Then equations become:

9/s = 4 - u ...(1b)

9/(s+2) = 2.4 - u ...(2b)

Now, subtract equation (2b) from equation (1b):

9/s - 9/(s+2) = (4 - u) - (2.4 - u) = 1.6

Simplify left side: 9[ (s+2 - s)/(s(s+2)) ] = 9[2/(s(s+2))] = 18/(s(s+2))

So 18/(s(s+2)) = 1.6 = 8/5 (converting 1.6 to fraction: 8/5)

Therefore, s(s+2) = 18 * (5/8) = 90/8 = 45/4 = 11.25

Wait, s(s+2)=45/4 \u21d2 s\u00b2 + 2s - 45/4=0 \u21d2 multiply by 4: 4s\u00b2 +8s -45=0, which is exactly
↪  the quadratic we had earlier! Good, so that's a good consistency check.
```

```
Then from (1b): u=4 - 9/s. We know s=2.5=5/2, so 9/s=9/(5/2)=18/5=3.6, so u=4 - 3.6=0.4 hours=24 minutes,
↪  correct.

Now, for speed v=s + 1/2=5/2 +1/2=3 km/h, total time T(v)=9/v + u=9/3 + 0.4=3 + 0.4=3.4 hours. Convert to
↪  minutes: 3.4*60=204 minutes. Perfect, same answer.

So that's reassuring. The key was setting up the equations correctly, which we verified with two methods
↪  and a check, so the total time is 204 minutes.

Wait, just to make sure there isn't a trick in the problem statement: "the walk takes her 4 hours,
↪  including t minutes spent in the coffee shop". Does "the walk" refer to the coffee shop trip? No, the
↪  coffee shop is where she stops after the walk, so the walk is the 9 km, then coffee shop is separate.
↪  But the problem says the time for the walk (activity) includes coffee shop time, which is a bit
↪  confusing, but since the math checks out with the problem's given times (4 hours total with t minutes,
↪  2h24m total with t minutes), and we found consistent t, it's probably the right interpretation.

Another possible misinterpretation: maybe t minutes is the time she spends walking? But no, the problem
↪  says "t minutes spent in the coffee shop", so t is coffee time.

Wait, let's suppose for a second that we misread and thought the 9 km is the time or something, but no, it
↪  says "9-kilometer-long walk", so distance is 9 km.

Wait, let's see if the problem says "the walk takes her 4 hours"\u2014if "the walk" is the 9 km, then
↪  walking time is 4 hours, but then she spends t minutes at coffee shop, so total time is 4h + t min,
↪  but the problem says "the walk takes her 4 hours, including t minutes...", so "including" implies that
↪  the t minutes are part of the 4 hours, so the walk's duration is 4 hours, which comprises walking and
↪  coffee time. So yes, our original interpretation is correct.

I think the key was not overcomplicating it and setting up the equations with total time = walking time +
↪  coffee time, which gave us two equations that simplified nicely, and the check confirmed the values of
↪  s and t, so we didn't have to worry about extraneous solutions or wrong interpretations.

So to recap:

- Found s=2.5 km/h, t=24 minutes via equations, verified with time calculations.

- New speed: 2.5 + 0.5=3 km/h.

- Walking time for 9 km at 3 km/h: 3 hours=180 minutes.

- Total time including coffee: 180 +24=204 minutes.

All steps check out, multiple methods confirm the answer. I think that's solid.

I think sometimes when problems say "the walk takes X time including Y time", it's easy to get confused
↪  about what's included, but by testing with the numbers we got (which matched the problem's time
↪  descriptions), we could be confident in the setup.

Just to write the final answer clearly: 204 minutes.

**Final Answer**\n\boxed{204}\n</think>

We are given that Aya walks a **9-kilometer-long walk** and stops at a coffee shop after. The problem
↪  states:

- When she walks at a constant speed of $ s $ km/h, the **total time** (including $ t $ minutes spent in
↪  the coffee shop) is **4 hours**.\n- When she walks at $ s + 2 $ km/h, the **total time** (including $
↪  t $ minutes at the coffee shop) is **2 hours and 24 minutes**.

We are to find how many **minutes** the walk takes her when she walks at $ s + \frac{1}{2} $ km/h,
↪  including the $ t $ minutes spent at the coffee shop.

---

### Step 1: Convert all times into minutes for consistency

Let\u2019s define:\n- Total time in minutes for first scenario: $ 4 \times 60 = 240 $ minutes\n- Total
↪  time in minutes for second scenario: $ 2 \times 60 + 24 = 144 $ minutes

Let:\n- $ t $ be the number of minutes spent at the coffee shop (same in both scenarios)\n- Then the
↪  **walking time** in each case is:\n  - First scenario: $ 240 - t $ minutes\n  - Second scenario: $ 144
↪  - t $ minutes

Since she walks 9 km in each case, we can write equations based on speed and time:

$$\n\text{Speed} = \frac{\text{Distance}}{\text{Time}} \Rightarrow \text{Time} =
↪  \frac{\text{Distance}}{\text{Speed}}\n$$

So for the first scenario:

$$\n\frac{9}{s} = \frac{240 - t}{60}\n\quad \Rightarrow \quad 540 = s(240 - t) \tag{1}\n$$

For the second scenario:

$$\n\frac{9}{s + 2} = \frac{144 - t}{60}\n\quad \Rightarrow \quad 540 = (s + 2)(144 - t) \tag{2}\n$$

---

### Step 2: Solve the system of equations

From (1):  \n$$\ns(240 - t) = 540\n$$

From (2):  \n$$\n(s + 2)(144 - t) = 540\n$$
```

```
Set them equal:

$$\ns(240 - t) = (s + 2)(144 - t)\n$$

Expand both sides:

Left: $ 240s - st $

Right: $ s(144 - t) + 2(144 - t) = 144s - st + 288 - 2t $

Subtract $ 144s - st $ from both sides:

$$\n96s = 288 - 2t\n\quad \Rightarrow \quad 48s = 144 - t\n\quad \Rightarrow \quad t = 144 - 48s
↪  \tag{3}\n$$

Now plug (3) into equation (1):

$$\ns(240 - (144 - 48s)) = 540\n\quad \Rightarrow \quad s(96 + 48s) = 540\n\quad \Rightarrow \quad 48s^2 +
↪  96s - 540 = 0\n$$

Divide by 12:

$$\n4s^2 + 8s - 45 = 0\n$$

Use quadratic formula:

$$\ns = \frac{-8 \pm \sqrt{64 + 720}}{8} = \frac{-8 \pm \sqrt{784}}{8} = \frac{-8 \pm 28}{8}\n$$

Take positive root:

$$\ns = \frac{20}{8} = 2.5 \text{ km/h}\n$$

Now compute $ t $ using (3):

$$\nt = 144 - 48 \cdot 2.5 = 144 - 120 = 24 \text{ minutes}\n$$

---

### Step 3: Compute total time at new speed

New speed: $ s + \frac{1}{2} = 2.5 + 0.5 = 3 $ km/h

Walking time for 9 km:

$$\n\frac{9}{3} = 3 \text{ hours} = 180 \text{ minutes}\n$$

Add coffee shop time $ t = 24 $ minutes:

$$\n\text{Total time} = 180 + 24 = \boxed{204} text{ minutes}\n$$

---

### Final Answer

$$\n\boxed{204}\n$$
```

TOOLUSE Qwen3-4B-Thinking-2507 Generation

```
Okay, the user is asking for a random picture of an axolotl. Let me check the available tools.

First, there's the Axolotl tool with two main functions: getRandomAxolotlImage and searchAxolotlImages.
↪  Since the user wants a random picture, the getRandomAxolotlImage tool seems perfect here. The search
↪  tool requires parameters like color, gender, size, which the user didn't specify, so that's not
↪  needed.

The getRandomAxolotlImage has no parameters, so the Action Input should be an empty JSON object. I need to
↪  make sure the response is in the correct JSON format as per the documentation. The output should be an
↪  object with url, source, and description.

I should call getRandomAxolotlImage with no parameters. Let me structure the Action correctly. The user
↪  didn't mention any specific criteria, so no need for the search tool. Just use the random one.
</think>

Thought: The user is asking for a random picture of an axolotl. The appropriate tool for this is
↪  getRandomAxolotlImage, which requires no parameters.

Action: getRandomAxolotlImage
Action Input: {}
```