CONTINUAL LEARNING WITH SUCCESS CONDITIONED SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with binary verifiable rewards provides only one bit of feedback per rollout, making credit assignment across thousands of tokens difficult. We argue that the success-conditioned policy provides a natural dense learning target for both single-task and continual learning by restricting the reference model to its own successful responses, which is the least disruptive way to succeed. Computing this target, however, requires a token-level value function that many verifiable-reward methods avoid learning. Prompting a frozen model with one of its own verified successes offers a full-vocabulary surrogate without an explicit critic. Yet this surrogate conditions on a particular successful trajectory rather than on success itself, introducing trajectory-specific nuisance information. We propose Success-Conditioned Self-Distillation (SCSD), which addresses this mismatch by searching for a textual transformation of a successful rollout that brings the prompted policy closer to the success-conditioned target. The resulting teacher is then distilled into a standalone student policy through on-policy distillation on the student's own rollouts. Across multiple tasks, SCSD achieves state-of-the-art performance in both single-task and continual learning settings.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a central tool for post-training large language models (LLMs) on reasoning, coding, and agentic tasks (Shao et al., 2024; Guo et al., 2025). Beyond improving a single capability, a deployed model is expected to keep acquiring new skills, which makes continual post-training, i.e., learning a stream of tasks without losing the earlier ones, an increasingly important setting (Parisi et al., 2019; Wang et al., 2024). However, the learning signal of RLVR is extremely sparse: a rollout of several thousand tokens is scored by a single bit $R(x, y) \in \{0, 1\}$, and policy-gradient methods such as GRPO (Shao et al., 2024) must spread that bit across every token.

To obtain a denser signal, on-policy self-distillation re-prompts the model with privileged information and uses its own next-token distribution as the teacher (Hübotter et al., 2026; Shenfeld et al., 2026; Ye et al., 2026). With binary rewards, the most readily available privileged information is one of the model's own verified successes. Existing methods therefore face a trade-off between signal density and stability. Sparse on-policy RL only reweights behavior the model already produces, so it moves the policy little and forgets less, but learns slowly per update (Wang et al., 2026). Dense self-distillation learns faster, but its teacher suppresses uncertainty and induces large drift, which in continual post-training accumulates into severe forgetting (Kim et al., 2026; Wang et al., 2026). A critical research question therefore emerges:

Can we obtain dense supervision as self-distillation while inheriting small drift as on-policy RL?

In this paper, we make a key finding that the drift of self-distillation is not intrinsic to dense supervision, but comes from what the teacher is conditioned on. We view every teacher as a point in a *drift–success plane* (Figure 1): its success probability against its divergence from the reference model π_{ref}, which we call its drift. The plane has a hard lower boundary. Any policy that succeeds with probability u on a prompt of pass rate p_x pays a KL of at least $d_{\text{KL}}(u\|p_x)$ from π_{ref}, and the boundary is attained at $u = 1$ by $\pi_{\mathcal{S}}$, the reference model restricted to its own successful responses (Section 3.1). The hindsight-prompted teacher of prior self-distillation conditions on *one specific*

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.

success $y^\star$ rather than on *success itself*, i.e., the whole success set, so it also transports trajectory-specific nuisance, such as one phrasing, one reasoning route, and one formatting choice. It reaches high success but far to the right of the boundary, and the gap, which we call the excess drift, is the part of its drift that success does not require.

We propose Success-Conditioned Self-Distillation (SCSD), which treats the teacher's context as the optimization variable. SCSD searches in text space for a transformation g of the verified success whose prompted teacher still succeeds while staying as close as possible to π_{ref}, i.e., a minimum-drift context. The searched teacher is frozen and distilled into the student on its own rollouts, so the student never sees the success at inference. SCSD adds no critic and no separately trained teacher, and reduces to SDPO (Hübotter et al., 2026) when g is the identity.

We evaluate SCSD with Qwen3-8B on math, science, and tool use, both on single tasks and on the sequence MATH → SCIENCE → TOOLUSE. At matched gradient updates, SCSD is the only method that reliably improves science, it avoids the collapse SDPO suffers on math, and it moves the policy 2.8–3.3× less than SDPO on these two tasks. In the continual setting, SCSD has the largest on-task gain at every stage and retains more than SDPO on two of the three tasks.

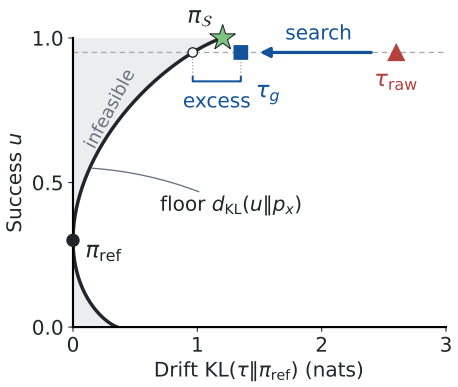

Figure 1: **The drift–success plane** (illustrative). SCSD moves the teacher from the raw success toward the floor which retains success while having less drift.

2 Preliminaries

Problem setting. We cast post-training with a verifiable reward as a token-level Markov decision process. A prompt $x \sim \mathcal{D}$ is the initial state, the state at step t is the prefix $s_t = (x, y_{<t})$, the action is the next token y_t, and a verifier returns a binary outcome $R(x, y) \in \{0, 1\}$ at the end of the response. We write π_{ref} for the reference model, i.e., the weights at the start of the current round of training, and π_θ for the policy being trained. For each prompt, let $\mathcal{S}_x = \{y : R(x, y) = 1\}$ be its success set and $p_x = \pi_{\text{ref}}(\mathcal{S}_x \mid x)$ the pass rate of π_{ref}. The goal is to maximize $J(\pi) = \mathbb{E}_{x\sim\mathcal{D}}[\pi(\mathcal{S}_x \mid x)]$.

Continual post-training. The model is trained on tasks $\mathcal{T}_1, \ldots, \mathcal{T}_K$, each with prompt distribution $\mathcal{D}_k$ and verifier R_k, and only $\mathcal{D}_k$ is available at stage k. Stage k initializes from π_{k-1}, so its reference model is $\pi_{\text{ref}} = \pi_{k-1}$. We report the on-task gain $J_k(\pi_k) - J_k(\pi_{k-1})$, which measures plasticity, and the change $J_j(\pi_K) - J_j(\pi_0)$ on every task after the full sequence, which measures stability.

On-policy self-distillation. Given a privileged context c, the teacher is the same model conditioned on c, $\tau_c(\cdot \mid s) = \pi_{\text{ref}}(\cdot \mid s, c)$, and the student minimizes $\mathbb{E}_{s\sim d_{\pi_\theta}}[D(\tau_c(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s))]$ on its own prefixes for some divergence D. SDPO (Hübotter et al., 2026) takes c to be a successful rollout $y^\star$ of the same prompt, together with environment feedback when available. This keeps the on-policy state coverage of RL while replacing one bit per response with a full distribution per token. Compared with GRPO, whose single advantage per response makes credit assignment coarse, it learns faster; however, the prompted teacher can be much sharper than π_{ref}, and dense self-distillation drifts further and forgets more than RL (Kim et al., 2026; Wang et al., 2026). The question we study is what the teacher's context c should contain.

3 Success-Conditioned Self-Distillation

In this section, we introduce SCSD. We first characterize the cheapest way for any teacher to succeed, which gives a calibrated scale for the drift a teacher introduces. We then show that the raw hindsight teacher is over-conditioned under this scale, cast the choice of its context as a minimum-drift search, and distill the searched teacher on-policy. Figure 2 gives an overview.

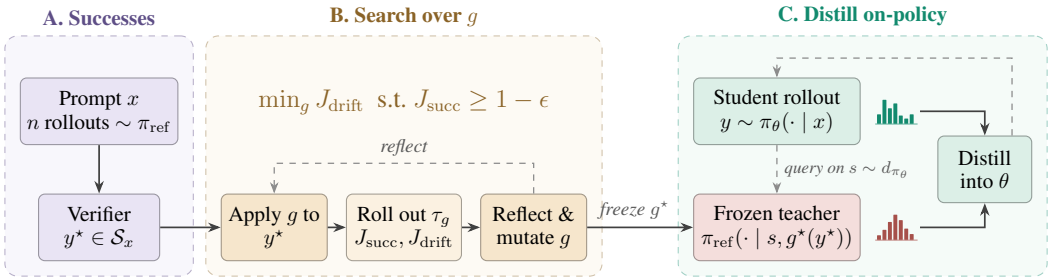

Figure 2: **Overview of SCSD.** (A) Verified successes are collected from the reference model π_{ref}. (B) A text-space search looks for a transformation g of a success whose prompted teacher τ_g keeps the success rate J_{succ} while minimizing its drift J_{drift}, both measured with π_{ref} alone. (C) The winning teacher is frozen and distilled into the student on the student's own rollouts. SCSD reduces to SDPO when g is the identity.

3.1 The Drift Floor of Success

For a prompt with $p_x > 0$, let $\pi_{\mathcal{S}}$ be the reference model restricted to its success set,

$$\pi_{\mathcal{S}}(y \mid x) = \frac{\pi_{\text{ref}}(y \mid x)}{p_x} \mathbf{1}\big[y \in \mathcal{S}_x\big]. \tag{1}$$

The following two results show that $\pi_{\mathcal{S}}$ is the least disruptive way to obtain success, and that this gives a natural scale for any teacher we might build.

Proposition 1 (Minimum-change successful policy)**.** *Among all policies supported on $\mathcal{S}_x$, $\pi_{\mathcal{S}}$ is the unique minimizer of $D_{\text{KL}}(\pi \| \pi_{\text{ref}})$, and $D_{\text{KL}}(\pi_{\mathcal{S}} \| \pi_{\text{ref}}) = \log(1/p_x)$.*

Proposition 2 (Drift floor)**.** *For any policy π with success probability $u = \pi(\mathcal{S}_x \mid x)$,*

$$D_{\text{KL}}(\pi \| \pi_{\text{ref}}) \;\geq\; d_{\text{KL}}(u \| p_x), \qquad d_{\text{KL}}(u \| v) = u \log \frac{u}{v} + (1-u) \log \frac{1-u}{1-v}. \tag{2}$$

Proposition 1 formally says that $\pi_{\mathcal{S}}$ is the KL projection of π_{ref} onto the policies that always succeed, and Proposition 2 extends this to any success level. Both results follow from the data-processing inequality applied to the outcome map $y \mapsto R(x, y)$, and setting $u = 1$ in Proposition 2 recovers the value in Proposition 1 (Appendix A). Intuitively, the floor separates the divergence a teacher must pay to succeed from the divergence it pays for anything else. We therefore define the *excess drift*

$$\Delta(\pi) = D_{\text{KL}}(\pi \| \pi_{\text{ref}}) - d_{\text{KL}}\big(u_\pi \| p_x\big) \;\geq\; 0, \tag{3}$$

which measures how much behavior change a teacher introduces that success does not explain. Note that $\pi_{\mathcal{S}}$ is dense, since each of its next-token conditionals is defined, but computing those conditionals requires the success probability of every continuation under π_{ref}, i.e., a token-level value function under a sparse binary reward. This is exactly the learned critic that verifier-based training aims to avoid, and SCSD aims to find solutions to avoid estimating it.

3.2 Hindsight-Prompted Teacher: Conditioning on One Success, Not on Success Itself

Suppose we sample n rollouts from π_{ref} and the verifier marks at least one as successful, giving $y^\star \in \mathcal{S}_x$. Placing $y^\star$ in context gives the raw hindsight teacher

$$\tau_{\text{raw}}(a \mid s) = \pi_{\text{ref}}(a \mid s, y^\star), \tag{4}$$

which costs one frozen-model pass and no additional parameters. However, τ_{raw} conditions on *one* element of $\mathcal{S}_x$, not on the *whole* success set $\mathcal{S}_x$ that defines $\pi_{\mathcal{S}}$. Conceptually, decompose a success as $y^\star = (z_{\text{success}}, z_{\text{nuisance}})$, where z_{success} explains *why* the rollout worked, such as the strategy or the decisive check, and z_{nuisance} is everything specific to this instance, such as wording, redundant deliberation, and formatting habits. Conditioning on all of $y^\star$ transports both, so the

teacher becomes confident about content that carries no information about success. In the latent-strategy view of in-context learning (Xie et al., 2022), a single trajectory concentrates the teacher's posterior on the one strategy it realizes, whereas success alone retains every strategy that works. This is over-conditioning rather than under-conditioning: τ_{raw} is typically *sharper* than necessary, and in the terms of Eq. (3) it succeeds with high probability at a large excess drift. This matches the empirical record: rich privileged context suppresses the model's uncertainty (Kim et al., 2026) and denser self-distillation drifts and forgets more (Wang et al., 2026).

3.3 Minimum-Drift Context Search

A transformation g maps the selected success to a shorter, more transferable description and induces the teacher

$$\tau_g(a \mid s) := \pi_{\text{ref}}(a \mid s, g(y_x^\star)) \,. \tag{5}$$

We score it by its pass rate and by its divergence from the unprompted model,

$$J_{\text{succ}}(g) = \mathbb{E}_x\big[\tau_g(\mathcal{S}_x \mid x)\big], \qquad J_{\text{drift}}(g) = \mathbb{E}_x[D_{\text{KL}}(\tau_g \| \pi_{\text{ref}})] \,, \tag{6}$$

and, following Proposition 1, search for

$$g^\star = \arg\min_g J_{\text{drift}}(g) \quad \text{s.t.} \quad J_{\text{succ}}(g) \geq 1-\epsilon, \quad \mathcal{L}(g) = J_{\text{drift}}(g) + \lambda\big[(1-\epsilon) - J_{\text{succ}}(g)\big]_+^2, \tag{7}$$

where $[\cdot]_+ = \max(0, \cdot)$ and we optimize the penalized form $\mathcal{L}$ with λ large enough that the success term dominates whenever the constraint is violated. When g is the identity, $\tau_g = \tau_{\text{raw}}$ and SCSD reduces to SDPO; when g maps every success to the empty string, $\tau_g = \pi_{\text{ref}}$ and there is nothing to distill.

Why not simply maximize reward? A reward-maximizing context is free to over-specify: in the limit it can dictate the answer or pin the model to the surface form of $y^\star$, achieving $J_{\text{succ}} \approx 1$ far from π_{ref}. By minimizing J_{drift} subject to success, we ask instead for *the least behavior change that still guarantees success*, which is the defining property of $\pi_{\mathcal{S}}$. The constraint keeps the problem non-trivial, since the empty context attains $J_{\text{drift}} = 0$ but only the base pass rate.

Measuring the drift and searching in text space. Both terms are measurable with the frozen model alone: for $y \sim \tau_g$, $\widehat{D_{\text{KL}}}(\tau_g \| \pi_{\text{ref}}) = \sum_t [\log \pi_{\text{ref}}(y_t \mid s_t, g(y_x^\star)) - \log \pi_{\text{ref}}(y_t \mid s_t)]$, and the verifier estimates J_{succ} on the same rollouts. Because g is text, we optimize it with reflective evolutionary search in the style of GEPA (Agrawal et al., 2026) rather than by gradient descent. Starting from a generic coarsening instruction g_0 (Figure 4), each search cycle proceeds as follows:

1.	**Apply**	rewrite the verified successes $y^\star$ of a minibatch of prompts with g;
2.	**Verify**	roll out the induced teacher τ_g and verify it, which gives $\widehat{J}_{\text{succ}}$;
3.	**Score**	score the same rollouts with the estimator above, which gives $\widehat{J}_{\text{drift}}$;
4.	**Reflect**	explain in natural language why a candidate fails and where its drift is high;
5.	**Mutate**	propose a new g from the reflection, and accept or reject it by $\mathcal{L}$.

3.4 Distilling the Searched Teacher On-Policy

After search, we freeze the teacher $\tau_\star(a \mid s) = \pi_{\text{ref}}(a \mid s, g^\star(y_x^\star))$. The student generates bare-prompt rollouts and minimizes

$$\mathcal{L}_{\text{distill}}(\theta) = \mathbb{E}_{x,\, s \sim d_{\pi_\theta}}\big[D\big(\tau_\star(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s)\big)\big]. \tag{8}$$

Only the teacher sees $y_x^\star$; student generation and inference use x alone, and querying the teacher on student prefixes keeps the supervision on the states the student visits (Agarwal et al., 2024; Ross et al., 2011).

Practical consideration. Rather than searching once, we interleave search and distillation, re-running the search every 10 updates on the retained rollouts, and require each candidate to match the pass rate of the unabridged $y^\star$ within $\epsilon = 0.05$. For D we use the same token-level divergence as SDPO, a top-100 generalized Jensen–Shannon divergence with mixing weight 0.5, so that SCSD and SDPO differ only in the teacher's context. Appendix C.7 lists all search settings, including three places where the implementation deviates from the objective as stated here.

4 Related Work

On-policy self-distillation. A recent line of work obtains dense supervision by conditioning the model on privileged information and distilling the result back into the unconditioned model on its own rollouts. The privileged information can be demonstrations (Shenfeld et al., 2026), successful rollouts and environment feedback (Hübotter et al., 2026), reference answers (Yang et al., 2026), task context (Ye et al., 2026), or agentic trajectories (Lu et al., 2026), and the transfer follows on-policy distillation (Agarwal et al., 2024; Ross et al., 2011). Recent analyses show that such teachers can suppress uncertainty and degrade out-of-distribution reasoning (Kim et al., 2026), and that denser self-distillation drifts further and forgets more than RL in continual post-training (Wang et al., 2026). Concurrent work studies which improvement objective self-distillation implicitly optimizes (Anonymous, 2026); we instead ask what the teacher's context should contain. Fast-slow training also optimizes a textual context with reflective search, but uses it as fast weights at inference (Tiwari et al., 2026). Compared with these methods, SCSD makes the privileged context itself the search variable, chooses it by a minimum-drift criterion, and uses it only inside the teacher.

Continual learning. Continual learning studies how to learn a sequence of tasks without catastrophic forgetting (McCloskey & Cohen, 1989; Goodfellow et al., 2013; Parisi et al., 2019; Wang et al., 2024). Classic methods fall into three families (van de Ven & Tolias, 2019): regularization, which penalizes changes to important parameters (Kirkpatrick et al., 2017) or distills the previous model's outputs (Li & Hoiem, 2017; Rebuffi et al., 2017); replay, which constrains the update with a small memory (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a;b); and architectural expansion (Rusu et al., 2016), including learnable prompts on a frozen backbone (Wang et al., 2022b;a; Razdaibiedina et al., 2023). Parameter regularization alone is insufficient in class-incremental settings (Lesort et al., 2019), and long sequences also erode the ability to learn at all (Berariu et al., 2021; Dohare et al., 2024). For LLMs, sequential fine-tuning causes forgetting (Luo et al., 2023) that grows predictably with the amount of fine-tuning (Kalajdzievski, 2024), degrades aligned models (Li & Lee, 2024), and skews the model's implicit task inference (Kotha et al., 2024). Remedies include pseudo-replay (Sun et al., 2020; Wang et al., 2021), recall-and-learn regularization (Chen et al., 2020), continual knowledge learning and pre-training (Jang et al., 2022; Jin et al., 2022; Ke et al., 2023), orthogonal or mixture-of-expert low-rank adapters (Wang et al., 2023a; Dou et al., 2024), and benchmarks such as TRACE (Wang et al., 2023b); instruction-tuned models can themselves be strong continual learners (Scialom et al., 2022), while training on self-generated data can erase distribution tails (Shumailov et al., 2023). For RL-based post-training, KL-regularized fine-tuning has been read as distribution matching that avoids forgetting (Korbak et al., 2022b;a), COPR regularizes toward earlier optimal policies (Zhang et al., 2023), and on-policy RL forgets less than supervised fine-tuning and dense self-distillation (Wang et al., 2026). SCSD adds no replay, regularizer, or new parameters; it reduces forgetting by making the dense target itself minimal in divergence from the model being updated.

Text-based optimization. A growing line of work treats natural-language text as the optimization variable and searches it with an LLM in the loop. Prompt optimization methods select instructions by proposal and search (Zhou et al., 2023), let an LLM act as the optimizer over past candidates and their scores (Yang et al., 2024), follow textual "gradients" with beam search (Pryzant et al., 2023), or evolve prompts with evolutionary operators (Guo et al., 2024; Fernando et al., 2023). TextGrad extends textual feedback to backpropagation through compound systems (Yuksekgonul et al., 2024), DSPy compiles such pipelines end to end (Khattab et al., 2023), and GEPA evolves prompts with natural-language reflection (Agrawal et al., 2026). Verbal feedback can even replace weight updates altogether (Shinn et al., 2023). A separate line, context distillation, moves the effect of a prompt into the weights (Askell et al., 2021; Snell et al., 2022; Ye et al., 2026). Compared with these methods, which optimize the text for the task reward and keep it at inference, SCSD optimizes the teacher's context for minimum drift subject to success. In this sense, the optimized text never reaches the student; it only shapes the teacher that is distilled into the weights.

5 Experiments

The goal of our experiments is to answer the following questions: (1) Does SCSD improve the policy over on-policy RL and self-distillation when training a single task? (2) Does coarsening the

Table 1: **Single-task results** on Qwen3-8B. *Left*: in-distribution accuracy as mean@8 and pass@8. *Right*: divergence from the base model in KL per token (k_3 estimate on 100 fixed training prompts per domain). SCSD is the only method that improves science and stays at or above the base model on in-distribution pass@8, while moving the policy 2.8–3.3× less than SDPO on science and math.

	Method	Science	Math	Tool use
mean@8	Base model	0.263	0.579	0.603
	GRPO	0.293	0.538	0.564
	SDPO	0.390	0.379	0.577
	SCSD	**0.493**	**0.558**	0.586
pass@8	Base model	0.600	0.700	0.618
	GRPO	0.560	0.733	0.603
	SDPO	0.500	0.633	0.632
	SCSD	**0.620**	**0.767**	**0.647**

	KL from base		
Method	Sci.	Math	Tool
GRPO	**0.002**	**0.001**	**0.002**
SDPO	0.309	0.129	0.317
SCSD	0.111	0.039	0.268

teacher's context reduce how far the policy moves? (3) Does this lead to a better trade-off between learning and forgetting in continual post-training?

Setup. We use Qwen3-8B with thinking mode enabled and consider three tasks with verifiable rewards: **Math** trains on DAPO prompts (Yu et al., 2025) and is evaluated on AIME24 (30 prompts), **Science** uses the biology split of SciKnowEval (50 prompts), and **Tool use** uses ToolAlpaca (68 prompts). We compare SCSD with two representative baselines: (i) GRPO (Shao et al., 2024), the standard on-policy RL method with sparse rewards, and (ii) SDPO (Hübotter et al., 2026), which distills a teacher conditioned on the raw success and is the special case of SCSD with g the identity. To isolate the effect of the teacher's context, all methods share the batch size, rollout count, clipping, and per-task length budget, and both distillation methods freeze the teacher at the weights that begin each stage, i.e., π_{ref}.

Evaluation. We re-score every checkpoint, including the base model, under one decoding configuration (8 rollouts, temperature 0.6, top-p 0.95). We report **mean accuracy** over 8 rollouts, **pass**@8, and the **KL divergence from the base model** on a fixed probe set, which measures how far training moves the policy. Because the evaluation sets are small, we report 95% paired-bootstrap confidence intervals over prompts for every difference we interpret.

5.1 Single-Task Post-Training

SCSD is the only method that reliably improves science. As shown in Table 1, SCSD raises science accuracy from 0.263 to 0.493, a gain of $+0.230$ (95% CI $[+0.08, +0.38]$), and outperforms GRPO by $+0.200$ ($[+0.04, +0.36]$). SDPO also improves science ($+0.128$), but its gain is not resolvable on 50 prompts ($[-0.03, +0.28]$), and neither is the lead of SCSD over SDPO ($+0.102$, $[-0.05, +0.25]$).

SCSD avoids the math collapse of SDPO. Perhaps surprisingly, SDPO lowers math accuracy by 0.200 ($[-0.31, -0.10]$) after training on math itself, whereas SCSD stays at the base level (-0.021, $[-0.08, +0.04]$), so coarsening the context is worth $+0.179$ ($[+0.08, +0.28]$). This is consistent with the over-conditioning view of Section 3.2: a competition-math success carries the most trajectory-specific detail of the three tasks, and conditioning the teacher on all of it is worse than not training at all.

SCSD moves the policy less and never loses coverage. The drift follows GRPO $\ll$ SCSD $<$ SDPO on every task. Compared with the raw teacher of SDPO, coarsening the context reduces the drift by $3.3\times$ on math and $2.8\times$ on science, while GRPO barely moves the policy (a per-token KL of at most 0.002), which is consistent with its small accuracy change. The smaller drift also shows up in coverage: SDPO falls below the base model on pass@8 on math and science, even though its science mean rises. In other words, SDPO becomes more reliable on problems it already solves and loses those it solves only occasionally. In contrast, SCSD stays at or above the base model on pass@8 for every task. Appendix B.1 further reports out-of-distribution accuracy.

Table 2: **The continual-learning matrix.** Every stage checkpoint scored on every task (mean@8). The underlined entry of each row is the task that stage trained. Δ_{stage} is the divergence from the weights the stage inherited and Δ_{base} from the untrained model, both in KL per token on a fixed probe. All methods take equal gradient updates. SCSD has the largest on-task gain at every stage and ends the sequence closer to the base model than SDPO.

		Accuracy (mean@8)			Divergence	
Method	Stage	Math	Science	Tool use	Δ_{stage}	Δ_{base}
Base model	—	0.521	0.293	0.572	—	0
GRPO	1 (math)	0.538	0.293	0.564	0.001	0.001
	2 (science)	0.496	0.320	0.577	0.003	0.006
	3 (tool use)	0.542	0.330	0.590	0.003	0.006
SDPO	1 (math)	0.408	0.345	0.160	0.129	0.129
	2 (science)	0.225	0.435	0.357	0.094	0.899
	3 (tool use)	0.229	**0.460**	0.467	0.017	0.970
SCSD	1 (math)	**0.546**	0.260	0.559	0.039	0.039
	2 (science)	0.325	0.378	0.482	0.065	0.313
	3 (tool use)	0.346	0.453	**0.610**	0.009	0.538

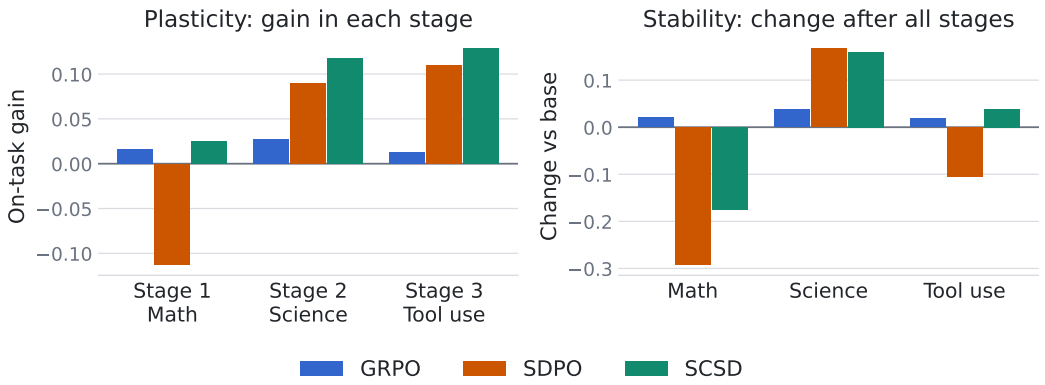

Figure 3: **The two halves of the continual-learning trade-off.** *Left*: the gain each stage makes on the task it trains (plasticity). *Right*: each task's accuracy after all stages relative to the untrained model (stability). SCSD has the largest on-task gain in every stage and ends above SDPO on math and tool use while matching it on science.

5.2 Continual Post-Training

We now train a single model through the sequence MATH → SCIENCE → TOOLUSE. Each stage initializes from the previous stage's weights and reuses the configuration of the corresponding single-task run, and every stage checkpoint is scored on *all three* tasks, which makes both learning and forgetting visible. Table 2 gives the full matrix, and Figure 3 decomposes it into plasticity, i.e., the on-task gain of each stage, and stability, i.e., the change of each task after the full sequence.

At matched updates, GRPO neither learns nor forgets much. GRPO ends the sequence within $+0.04$ of the base model on every task, with a per-stage KL of only 0.001–0.003 per token. In other words, GRPO does not forget because it barely changes the policy at all under this budget, not because it learns new tasks without forgetting old ones.

SDPO trades every other task for the one it is trained on. During the math stage, the tool-use accuracy of SDPO collapses from 0.572 to 0.160, i.e., training on math alone destroys most of an unrelated capability, and later stages recover it only to 0.467. Its math accuracy then falls to 0.225 during the science stage and never recovers. Only the task being trained improves, and SDPO overfits to get the highest science accuracy (0.460).

Seed instruction g_0 (shared by all runs)

```
Rewrite the correct solution below into the minimal set of transferable decisions and
checks that a solver would need in order to reach a correct answer on its own.  Keep only
what generalizes:  the approach taken, the key steps or facts it depends on, the quantities
that must be tracked, and the checks that confirm correctness.  State the final answer
once, at the end.  Remove the narration, the exploratory detours, the specific wording, and
any computation that can be redone from the steps you keep.  Write at most 10 short lines.
```

Searched g – Math

```
[...]  The solver already has a natural
way of writing a solution; every word you
transfer that is not strictly load-bearing
for correctness SUPPRESSES what it would
naturally write.  [...]
WHAT IS LOAD-BEARING (KEEP)
- The key facts, formulas, substitutions,
constraints, and the specific setup choices
the answer depends on [...]
CRITICAL: DISAMBIGUATING SETUP MUST SURVIVE
[...]  retain the specific relation the
solution uses (e.g.  the exact equation
form like "c^2 + h^2 = 22500").  [...]
WHAT CAUSES EXCESS (CUT)
- The solution's full chain of one-off
arithmetic, per-case number crunching, and
step-by-step bookkeeping.  [...]
- Abandoned approaches, trial-and-error,
and stated alternatives.  Give only the
final chosen path.
```

Searched g – Science

```
TASK CONTEXT: These are protein questions
of several related types:  comparing two
nearly identical protein sequences (labeled
A and B) to decide which mutant has higher
activity [...]
REWRITE RULES:
- Preserve the specific domain facts
the solution relies on:  substitution
position(s) and residue identities,
which mutated position matters, why a
substitution helps or hurts (charge/
polarity/hydrophobicity/size/steric/
active-site reasoning) [...]
- Do NOT verify or recompute.  Never
re-count residue positions, re-align
sequences, or re-examine which amino acid
sits at a stated position.  [...]
- Match the solution's own framing and word
choices so the rewrite stays close to it
and transfers minimal incidental detail.
```

Searched g – Science, stage 2 of the continual sequence (after training on math)

```
[...]  - PRESERVE the concrete domain-specific facts the solution relies on, since the
solver may not have them otherwise.  This includes:  identity mappings (e.g.  an ENSP/
Entrez/gene ID maps to a specific named protein), protein functions/roles, the specific
interaction or relationship type [...]
- Be concise but complete:  cutting essential facts lowers the solver's success; keeping
incidental wording adds excess.
```

Figure 4: **Seed and searched instructions of SCSD.** All searches start from the same seed g_0 (top). The searched g learns to keep the load-bearing part of a success and cut the rest, e.g., the decisive equations on math and the residue-level facts on science (middle), and it adapts to the model being taught (bottom). Instructions are quoted verbatim with elisions marked [...], and highlights are ours.

SCSD learns the most in every stage and forgets less than SDPO. SCSD achieves the largest on-task gain at all three stages ($+0.025$, $+0.117$, and $+0.129$, against -0.113, $+0.090$, and $+0.110$ for SDPO), including the last stage, where plasticity is hardest to maintain. At the same time, it ends the sequence above SDPO on math (-0.175 against -0.292 relative to the base model) and tool use ($+0.039$ against -0.105), and level on science ($+0.160$ against $+0.167$), while drifting only a little more than half as far from the base model (a KL of 0.538 against 0.970). This demonstrates that coarsening the teacher's context improves both sides of the plasticity–stability trade-off. Note that single-task results alone do not predict this, since the two distillation methods cannot be separated on single-task science.

6 Analysis and Limitations

6.1 What Does the Search Learn?

Because g is text, we can directly read what the search learns. Figure 4 shows the seed instruction g_0, from which every run starts, together with the instructions accepted at the end of three searches, quoted verbatim with elisions marked [...].

The searched g rediscovers the success–nuisance decomposition. Starting from a one-paragraph seed, every search converges to the same two-part structure: an explicit list of what is load-bearing and an explicit list of what to cut. This is exactly the decomposition $y^\star =$

$(z_{\text{success}}, z_{\text{nuisance}})$ of Section 3.2, which the search is never told about. It is induced by the objective alone: dropping z_{success} violates the success constraint, while every part of z_{nuisance} that is kept is charged by the drift term. More specifically, on math the searched g keeps the decisive setup exactly as the solution states it and cuts the arithmetic the solver can redo, which lowers the drift, i.e., the response-level KL from π_{ref}, from 736 to 540 at a success of 0.97–0.98. On science, it keeps substitution positions, residue identities, and the reason a substitution matters, which lowers the drift from 845 to 228 at a success of 0.99.

Why does the searched g lower the drift? Note that the math instruction states the drift objective in its own words, "every word you transfer that is not strictly load-bearing for correctness SUPPRESSES what it would naturally write". The science instruction goes one step further with two less obvious rules, "Do NOT verify or recompute" and "Match the solution's own framing and word choices". Intuitively, both keep the teacher close to π_{ref}. A rewriter that re-derives a step injects reasoning that π_{ref} did not produce, whereas the success $y^\star$ is itself a sample from π_{ref}, so its own wording is the cheapest way to carry its content. In this sense, the search learns not only *what* to transfer, but also *how* to phrase it at minimum drift.

Does the searched g adapt to the model it teaches? In the continual sequence, the science stage starts from the checkpoint after math, which scores 0.260 on science, below the base model's 0.293 (Table 2). The accepted instruction reflects this: it asks the rewriter to preserve domain facts "since the solver may not have them otherwise", such as which protein a gene identifier refers to. With it, the success on the search set rises from 0.73 under g_0 to 0.82–0.89 in the remaining cycles, while the drift stays below that of g_0 (203–349 against 411), and the instruction remains 2.7× shorter than the single-task one. This demonstrates that the search re-balances the decomposition to the reference model it is measured against: facts that the base model could recall become load-bearing once the model has moved.

6.2 Limitations

SCSD requires at least one verified success per prompt and is most useful at intermediate pass rates. Its drift estimate depends on a fixed decoding budget, and the selected exemplar can still shape the teacher after coarsening; we do not claim that τ_g equals $\pi_{\mathcal{S}}$. A low-drift teacher can also reveal information that a bare-prompt student cannot reconstruct, for which answer-masked transformations are the necessary control. Finally, the search adds rollouts for every candidate: an SCSD stage took 2.5–5× the wall-clock time of an SDPO stage.

7 Conclusion and Future Work

In this paper, we propose Success-Conditioned Self-Distillation (SCSD), which approximates the least disruptive successful policy by searching for a minimum-drift textual transformation of the model's own verified successes. The drift floor $d_{\text{KL}}(u\|p_x)$ calibrates this search and turns the excess drift into a measurable diagnostic of nuisance information. SCSD requires no critic and no separately trained teacher, and reduces to SDPO when the transformation is the identity. At matched gradient updates, it is the only method that reliably improves science, it avoids the math collapse of SDPO while moving the policy 2.8–3.3× less, and it has the largest on-task gain at every stage of a continual sequence. We believe that SCSD represents a concrete step toward dense post-training signals that change a model only as much as success requires. One promising direction is to scale the evaluation to longer task sequences and larger models, where the gap between minimal and unconstrained drift should matter most.

AI Use Statement

We used generative AI tools to launch experiments and polish the paper. All AI-assisted code and text were reviewed by the authors, and the reported numbers were checked against the experiment logs. We take responsibility for the full content of this work.

Reproducibility Statement

We describe the method in Section 3 and give every configuration needed to reproduce our results in Appendix C: the model, datasets, splits, and evaluation-set sizes (Appendix C.1); the continual-learning protocol (Appendix C.4); the search configuration (Appendix C.7); the decoding protocol and metrics (Appendix C.5). We will release the training and evaluation code.

References

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

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.19457.

Anonymous. Revisiting on-policy self-distillation from an RL perspective, 2026. Companion submission, under review; available as supplementary material.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Tudor Berariu, Wojciech Czarnecki, Soham De, Jorg Bornschein, Samuel Smith, Razvan Pascanu, and Claudia Clopath. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042*, 2021.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019a.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632:768–774, 2024.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

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

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. EvoPrompt: Connecting LLMs with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*, 2022.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*, 2022.

Damjan Kalajdzievski. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*, 2024.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. Continual pretraining of language models. In *International Conference on Learning Representations*, 2023.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

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, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. Controlling conditional language models without catastrophic forgetting. In *International Conference on Machine Learning*, 2022a.

Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. In *Advances in Neural Information Processing Systems*, 2022b.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. In *International Conference on Learning Representations*, 2024.

Timothée Lesort, Andrei Stoian, and David Filliat. Regularization shortcomings for continual learning. *arXiv preprint arXiv:1912.03049*, 2019.

Chen-An Li and Hung-Yi Lee. Examining forgetting in continual pre-training of aligned large language models. *arXiv preprint arXiv:2401.03129*, 2024.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Elsevier, 1989.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *International Conference on Learning Representations*, 2023.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

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

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

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

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

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. LAMOL: LAnguage MOdeling for lifelong language learning. In *International Conference on Learning Representations*, 2020.

Rishabh Tiwari, Kusha Sareen, Lakshya A Agrawal, Joseph E Gonzalez, Matei Zaharia, Kurt Keutzer, Inderjit S Dhillon, Rishabh Agarwal, and Devvrit Khatri. Learning, fast and slow: Towards llms that adapt continually. *arXiv preprint arXiv:2605.12484*, 2026.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Han Wang, Ruiliu Fu, Chengzhang Li, Xuejun Zhang, Jun Zhou, and Yonghong Yan. Reminding the incremental language model via data-free self-distillation. *arXiv preprint arXiv:2110.08745*, 2021.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Meng Wang, Haohan Zhao, Wenzhuo Liu, Lu Yang, Geng Liu, Haiyang Guo, Guo-Sen Xie, Gaofeng Meng, Hongbin Liu, and Fei Zhu. Denser $\neq$ better: Limits of on-policy self-distillation for continual post-training. *arXiv preprint arXiv:2607.01763*, 2026.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023a.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023b.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022b.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled RLVR. *arXiv preprint arXiv:2604.03128*, 2026.

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, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Han Zhang, Lin Gui, Yuanzhao Zhai, Hui Wang, Yu Lei, and Ruifeng Xu. COPR: Continual learning human preference through optimal policy regularization. *arXiv preprint arXiv:2310.15694*, 2023.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023.

A PROOFS

Proof of Proposition 1. Let π be supported on $\mathcal{S}_x$. On $\mathcal{S}_x$ we have $\pi_{\mathrm{ref}}(y \mid x) = p_x\, \pi_{\mathcal{S}}(y \mid x)$ by Eq. (1), so

$$D_{\mathrm{KL}}(\pi \| \pi_{\mathrm{ref}}) = \sum_{y \in \mathcal{S}_x} \pi(y \mid x) \log \frac{\pi(y \mid x)}{p_x\, \pi_{\mathcal{S}}(y \mid x)} = D_{\mathrm{KL}}(\pi \| \pi_{\mathcal{S}}) + \log \frac{1}{p_x}.$$

The first term is non-negative and vanishes only at $\pi = \pi_{\mathcal{S}}$, which proves both claims. □

Proof of Proposition 2. The outcome map $y \mapsto R(x, y)$ pushes $\pi(\cdot \mid x)$ forward to a Bernoulli distribution with mean u and $\pi_{\mathrm{ref}}(\cdot \mid x)$ to one with mean p_x. By the data-processing inequality, the divergence between two distributions cannot increase under a common deterministic map, so $D_{\mathrm{KL}}(\pi \| \pi_{\mathrm{ref}}) \geq D_{\mathrm{KL}}(\mathrm{Bern}(u) \| \mathrm{Bern}(p_x)) = d_{\mathrm{KL}}(u \| p_x)$. At $u = 1$ the right-hand side equals $\log(1/p_x)$, the value attained by $\pi_{\mathcal{S}}$. □

B ADDITIONAL RESULTS

B.1 OUT-OF-DISTRIBUTION ACCURACY

Table 3 reports the accuracy of the single-task checkpoints on a held-out suite from the same domain, under the same decoding as Table 1. All methods improve on Math500 after math training and degrade on GPQA after science training. Note that the GPQA loss grows with the science drift (-0.020, -0.108, and -0.119 for GRPO, SCSD, and SDPO), which suggests that the out-of-domain loss is tied to how far the policy moves.

Table 3: **Out-of-distribution accuracy** of the single-task checkpoints (Math500 after math training, GPQA-Diamond after science training).

	mean@8		pass@8	
Method	Math500	GPQA	Math500	GPQA
Base model	0.741	**0.534**	0.788	**0.731**
GRPO	0.744	0.514	0.802	0.670
SDPO	**0.768**	0.415	**0.872**	0.665
SCSD	0.762	0.426	0.814	0.609

B.2 CASE STUDIES OF THE SEARCHED CONTEXT

In this section, we provide additional case studies of the searched instruction g, complementing Figure 4. All quotations are verbatim from the search logs, with elisions marked [. . .] and highlights added by us.

How does the searched g evolve on math? The first accepted instruction on math is already a compact list of rules (Figure 5). It keeps the approach, the key steps, and the checks, removes narration and exploratory detours, and forbids the rewriter from adding reasoning of its own. The later instruction of Figure 4 keeps these rules but reorganizes them into an explicit "keep" list and "cut" list, and adds the requirement that the decisive setup survives exactly. Along the way, the drift falls from 736 to 540 while the success stays at 0.97–0.98, which demonstrates that the search improves the teacher by sharpening what counts as load-bearing rather than by transferring more of the success.

What does the searched g forbid on science? On science, the first and last accepted instructions open with the same sentence, and the search stops accepting candidates after three cycles at $J_{\mathrm{succ}} = 0.99$, while the drift falls from 845 to 228. Most of the 2453 added characters are prohibitions on the rewriter's own initiative (Figure 6). In other words, the search does not learn the task itself; instead, it closes off the ways in which the rewriter could inject reasoning of its own, which is exactly what the drift term penalizes. Note that the instruction also adds a short taxonomy of the task, which tells the rewriter which facts are load-bearing for each question type.

Searched g – Math, first accepted instruction

```
You are a REWRITER. You are given a math/physics/logic question and a VERIFIED CORRECT
SOLUTION to it.  Your job is to compress that solution into the minimal set of transferable
decisions and checks a solver needs to reach the correct answer on its own.  [...]
1.  Keep only what generalizes:  the overall approach, the key steps/facts it depends on,
the quantities that must be tracked, and the checks that confirm correctness.
2.  Remove narration, self-talk, hedging, exploratory detours, restatements of the problem,
and any arithmetic that can be re-derived from the steps you keep.  [...]
3.  Do not add reasoning, caveats, alternative interpretations, or assumptions that are not
in the given solution.  [...]
4.  State each retained step once, as a direct imperative or declarative line.  [...]
present only the final chosen path -- not the trial-and-error that led to it.
5.  State the final answer exactly once, at the very end, in the same form the solution
gives it (e.g., a boxed value).
```

Figure 5: **The first accepted instruction on math** ($J_{\text{succ}} = 0.98$, drift 736). The final instruction of the same search is shown in Figure 4.

Searched g – Science, rules added by the search

```
IMPORTANT: You are writing an instruction-following REWRITE of the solution.  You are NOT
solving the problem and NOT verifying it.  Do not attempt to independently recount sequence
positions, re-align sequences, re-check which residue is at which index [...]
TASK CONTEXT [...]:  These are protein questions of several related types:
- Comparing two nearly identical protein sequences (labeled A and B) to decide which mutant
has higher activity/fluorescence [...]
- Predicting a folding stability score for a single sequence from multiple-choice numeric
options.
- Predicting a fitness score for a mutant sequence given a wild-type reference and its
fitness [...]
- Do not introduce reasoning, positions, residues, or mechanisms the solution did not name.
- Be terse.  Prefer a short numbered or bulleted list.  Write at most 8 short lines.  Every
line must carry information the solution actually used [...]
```

Figure 6: **Rules added by the search on science** ($J_{\text{succ}} = 0.99$, drift $845 \rightarrow 228$).

What does the searched g keep on tool use? On tool use, the load-bearing part of a success is different: the answer is a tool call rather than a derivation. The final instruction from the base model reflects this (Figure 7). It requires the rewriter to keep the exact tool name and the full Action Input JSON, and to omit the reasoning that the solver writes fluently on its own, such as restating the request, listing parameters, and comparing candidate tools. With it, the drift falls from 210 at the first cycle to 104 at the last, while the success on the search set rises from 0.59 to 0.77. This shows that the same objective yields a different, task-appropriate split between what to keep and what to cut.

Searched g – Tool use, final instruction from the base model

```
[...]  But keep the notes as LIGHT as possible:  the solver has its own strong voice,
and every extra word, restated fact, or narrated inference you add distorts what it would
naturally write [...]
CRITICAL -- THE FORMAT ANCHOR [...]:  the exact tool name(s) chosen AND the full Action
Input JSON with realistic parameter values [...]  Reproduce parameter values EXACTLY as in
the solution (spelling, IDs, numbers, domains, casing, symbols).  [...]
CRITICAL -- AVOID EXCESS SUPPRESSION [...]:  The solver already writes fluent reasoning on
its own [...]  Measured high-suppression spans that you must OMIT:
- Opening narration ("the user is asking...", "let me check the tools available" [...]).
- Listing a tool's full parameter set [...]
- Comparisons ruling out other named tools, unless correctness genuinely hinges on a
non-obvious distinction.
CONTENT TO PRESERVE [...]:  Domain/derived facts the solver could NOT infer from the docs
alone AND that the solution uses [...]
```

Figure 7: **The final instruction on tool use** (J_{succ} $0.59 \rightarrow 0.77$, drift $210 \rightarrow 104$).

C Experimental Setup

In this section, we provide the full experimental configuration.

Table 4: Settings held identical across GRPO, SDPO, and SCSD. Sampling parameters for the *training* rollout are the model defaults; validation uses a separate, lower-temperature protocol (Appendix C.5).

Setting	Value
Prompts per update (question batch size)	32
Rollouts per prompt n	8
Training rollout sampling	temperature 1.0, top-p 1.0, top-k disabled
Advantage estimator	group-relative (GRPO), no critic
Advantage std. normalization	disabled
Policy-ratio clipping (low / high)	0.2 / 0.28
KL penalty in loss / in reward	disabled / disabled
Importance-sampling correction	token level, ratio threshold 2.0
Micro-batch size per GPU	1
Optimizer	AdamW, $\beta = (0.9, 0.999)$, weight decay 0.01, grad. clip 1.0
LR schedule	constant after 10 warmup updates

Table 5: Per-method settings. Dashes mark settings that do not apply. The learning-rate, minibatch, and environment-feedback rows are unmatched across methods; see the text. Both distillation arms run at $\rho = 0$.

Setting	GRPO	SDPO	SCSD
Learning rate	1×10^{-6}	1×10^{-5}	1×10^{-5}
Minibatch size (updates per batch)	8	32	32
Distillation loss	—	success-conditioned self-distillation	
Teacher regularization	—	EMA, rate $\rho = 0$ (frozen)	
Mixing weight α	—	0.5	0.5
Importance-weight clip	—	2.0	2.0
Top-k vocabulary bucketing	—	100	100
Environment feedback in teacher context	—	yes	no
Teacher context slot	—	raw $y_x^\star$	$g(y_x^\star)$
Text-space search over g	—	—	yes (Sec. C.7)
vLLM memory fraction	0.55	0.40	0.40

C.1 Model, Tasks, and Infrastructure

All runs use Qwen3-8B with thinking mode enabled, trained on a single node of eight H100-class GPUs. Note that we set thinking mode explicitly rather than inheriting it: Qwen3-8B is a hybrid model that reasons only when its chat template is invoked with the corresponding flag, and the flag is read both for the rollout prompt and for the teacher re-prompt, so an unset value would change both policies at once.

We train on three tasks. **Math** uses the DAPO training split, with AIME24 as the validation set. **Science** uses the biology split of SciKnowEval, a four-way multiple-choice benchmark. **Tool use** uses ToolAlpaca (4046 training and 68 test prompts). All methods train for 50 steps on each task.

C.2 Settings Shared by All Methods

Table 4 lists the settings shared by GRPO, SDPO, and SCSD, and Table 5 lists the method-specific ones.

Both distillation methods set the EMA teacher-update rate to $\rho = 0$, which keeps the teacher at the initial weights, i.e., the frozen reference model π_{ref} assumed in Section 3. Note that ρ is not a harmless hyperparameter: a small non-zero value substantially degrades the teacher's own pass rate during math training. We therefore report the setting that the analysis assumes and leave the sweep to future work.

Table 6: Per-task sequence-length budgets, from Eqs. (9)–(11). R is the response budget, Q the reserved question length, L the total model context, and TP the tensor-parallel degree. All values are below Qwen3-8B's 40960-token positional limit.

				GRPO		SDPO		SCSD	
Task	R	Q	Updates	P	L / TP	P	L / TP	P	L / TP
Math	14336	2048	50	2048	16896 / 1	2048	31744 / 2	16896	31232 / 2
Science	8192	2048	50	2048	10752 / 1	2048	19456 / 1	10752	18944 / 1
Tool use	8192	4096	50	4096	12800 / 1	4096	21504 / 1	12800	20992 / 1

C.3 Sequence-Length Budgets

The three methods need *different* context budgets for the same task, since they put different objects through the model. We write the budgets as explicit formulas rather than tuned constants, which keeps them valid as the response budget changes across tasks, and Table 6 gives the resulting values:

$$\text{GRPO:} \quad L = 512 + P + R, \tag{9}$$

$$\text{SDPO:} \quad L = 512 + P + (R + 512) + R, \tag{10}$$

$$\text{SCSD:} \quad P_{\text{scsd}} = Q + R + 512, \qquad L = P_{\text{scsd}} + R, \tag{11}$$

where R is the response budget, Q the number of tokens reserved for the question, P the prompt budget, and L the total model context. GRPO only runs a prompt and a response. SDPO additionally carries the raw success as *feedback* inside the teacher re-prompt, which is bounded separately from the prompt, so it keeps a small prompt budget and a large total context. In contrast, SCSD requires a large *prompt* budget: scoring a candidate g needs a reference pass that places the unabridged $y_x^\star$ in the prompt, and $y_x^\star$ is itself bounded by R.

The reserved question length Q is measured rather than guessed: the longest tokenized, chat-templated question is 1513 tokens on math, 498 on science, and 1520 on tool use. Note that a flat prompt budget for SCSD fails in both directions. If it is too small, the reference pass silently truncates $y_x^\star$, so every candidate g is scored against a corrupted reference; for example, a 2925-token question with a 14336-token response exceeds the 16k budget that suffices elsewhere. If it is too large, the run reserves key-value cache for context that is never used, which on science and tool use decides whether two-way tensor parallelism is needed. We therefore monitor a truncation counter on the reference pass, which is zero by construction under Eq. (11).

All three methods follow the same tensor-parallel rule: an 8B model with $L > 30000$ is split two ways, with the optimizer state offloaded to host memory. At one-way parallelism, every GPU holds a full ≈ 15 GiB copy of the parameters, and since the vLLM memory fraction covers both parameters and key-value cache, the cache runs out first. Note that neither choice affects the training semantics.

C.4 Continual-Learning Protocol

The sequence is MATH $\rightarrow$ SCIENCE $\rightarrow$ TOOLUSE. Stage 1 is the single-task run on math, and stage $k > 1$ initializes the policy from the checkpoint of stage $k - 1$, which is converted from its sharded FSDP state into a HuggingFace checkpoint before the run starts. Each stage reuses exactly the configuration of the corresponding single-task run, i.e., the same number of steps, batch size, learning rate, and prompt and response budgets, so a stage and its single-task counterpart differ only in their initial weights. Each stage saves a single checkpoint at its final update, containing the model weights but not the optimizer state; this is sufficient because every stage starts with a fresh optimizer.

Because a stage only validates the task it is training, we obtain the retention numbers from a separate pass, in which every stage checkpoint is scored post hoc on *all* tasks. This gives the $3 \times 3 \times 3$ matrix behind Table 2 and Figure 3.

C.5 Evaluation Infrastructure and Metrics

Decoding. Every checkpoint and the base model are scored with 8 rollouts per prompt at temperature 0.6, top-p 0.95, and top-k disabled, with the generation capped at the task's training response budget

Table 7: Reflective text-space search over g. The budget bounds candidate *scoring* rollouts per search, so with a search set of $|\mathcal{S}| = 64$ prompts it admits two accepted candidates per search.

Setting	Value
Search period (updates between searches)	10
Search set size $\lvert\mathcal{S}\rvert$	64 prompts
Scoring-rollout budget per search	200
Rollouts per prompt when scoring a candidate	4
Rollouts per prompt for the reference pass	4
Constraint slack ϵ in Eq. (7)	0.05
Penalty weight λ in Eq. (7)	10
Constraint form	per-prompt hinge
J_{drift} estimator	single sampled token log-ratio, Section 3.3
Reference for the constraint	unabridged $y_x^\star$, re-measured each search
Rewriter and reflection models	frozen instruction-tuned LMs, external to training

R (Table 6) for single-task checkpoints, and at 14336 tokens, the longest budget of any stage, for the continual-learning checkpoints. The score is the task verifier, identical to the training reward.

Out-of-distribution suites. Math is additionally scored on Math500 (500 prompts) and science on GPQA-Diamond (197 prompts).

C.6 Measuring Divergence from a Reference Policy

We report divergence as a token-mean k_3 estimate of $D_{\mathrm{KL}}(\pi\|\pi_{\mathrm{ref}})$ over response tokens, computed on rollouts that the evaluated checkpoint generates itself. Note that this quantity is therefore on-policy with respect to the checkpoint and not symmetric. The probe consists of 100 prompts from the checkpoint's own training domain with shuffling disabled, so every method is scored on identical inputs.

We use two references, which answer different questions. Against the **base model** (Tables 1 and 2), the divergence measures the total distance traveled since the start of training. Against the **previous stage**, it measures how far one stage moves the policy. Because the latter also changes the probe distribution across stages, we compare methods within a stage rather than stages against each other. To obtain the reference log-probabilities, which are only computed on the training path, these jobs run a single step at learning rate 0 with the divergence coefficient set to 0, so the weights are unchanged and only the logged divergence is read.

C.7 Search Configuration for SCSD

Table 7 gives the settings of the reflective search. The search is re-run every 10 updates on the rollouts retained from those updates, so candidate scoring adds no on-policy rollouts beyond the candidate generations themselves. Note that the retained rollouts are only used for measurement and never enter a gradient step.