

Does a Model Forget Differently When the Data Is Its Own? RL's Retention Advantage and Model Collapse Are Claims About the Same Loop, and No Study Has Measured Both

Mahendrakar, Pranay

SONYTECH

ORCID: 0009-0003-7224-029X

Life of Research

2026-09-25

Pranay Mahendrakar

Abstract

Two literatures make claims about what happens when a language model trains on data that resembles its own distribution. One asks whether reinforcement learning forgets a model's prior capabilities less than supervised fine-tuning does, and answers yes: on-policy RL is implicitly biased toward solutions that stay close, in KL divergence, to the base policy, while supervised fine-tuning can converge to distributions arbitrarily far away (Shenfeld, Pari and Agrawal, 2025). The other asks whether a model trained recursively on its own generated outputs degrades across generations, and answers yes as well: absent a steady supply of fresh real data, the tails of the training distribution disappear, a failure named model collapse (Shumailov et al., 2023). A self-training loop -- reinforcement learning with verifiable rewards applied to a policy that generates its own training data and is scored by a verifier it does not otherwise consult -- is an instance of both objects at once, and no study surveyed here measures both retention and collapse on the same system. This paper lays out what each literature actually measures, its reference point (a held-out prior task versus the original data distribution) and its timescale (one fine-tuning run versus many training generations); argues these are not obviously the same failure mode despite a shared vocabulary of KL divergence, entropy and distributional narrowing; and assembles a small set of 2025-2026 results -- on correct-set turnover inside RLVR itself, on RLVR's failure to expand a model's pass-at-large-k ceiling beyond its own base model, and on verifier reliability degrading in exactly the low-resource regime where synthetic data accumulates fastest -- that make the reconciliation harder than either literature admits when read on its own. The paper does not resolve which failure mode dominates a given self-training loop. It specifies the joint measurement nobody has run, and states plainly why each literature's current instruments cannot answer the other literature's question.

1. Introduction

Two claims about self-generated training data circulate in adjacent parts of the machine learning literature, and they point in opposite directions.

The first claim comes from work comparing reinforcement learning to supervised fine-tuning as ways of teaching a language model a new task. Shenfeld, Pari and Agrawal (2025) report that RL-tuned models retain prior capabilities substantially better than SFT-tuned models that reach the same new-task accuracy, and that the mechanism is structural rather than incidental: on-policy RL is implicitly biased toward the KL-minimal solution among all solutions that solve the new task, because it can only reinforce behavior the policy already assigns some probability to, while SFT can be pulled toward any distribution the training data happens to specify, however far from the base model that distribution sits.

They call this "RL's Razor." The claim is specifically about forgetting relative to a fixed prior policy, measured within a single fine-tuning run.

The second claim comes from work asking what happens when a generative model is trained, repeatedly, on data produced by an earlier version of itself or a closely related model, rather than on data collected from the world. Shumailov et al. (2023) name this model collapse and report that, absent a persistent supply of fresh real data, the tails of the original data distribution disappear across generations of this recursive process, a result they demonstrate in variational autoencoders, Gaussian mixture models and language models alike. Dohmatob et al. (2024) report that the effect is not a large-fraction problem: as little as one percent synthetic data in a training corpus is sufficient to trigger measurable collapse, and adding more total data does not rescue the outcome. The claim is about distributional coverage relative to the original real-data distribution, measured across many generations of a recursive training process.

A reader who takes both claims at face value has a puzzle on their hands whenever the two literatures' objects turn out to be the same system. Reinforcement learning with verifiable rewards (RLVR) -- the training recipe behind DeepSeek-R1 (DeepSeek-AI et al., 2025) and the dominant post-training paradigm for reasoning models through 2025 and 2026 -- has the policy sample its own candidate solutions, score them with a verifier (a unit test, a symbolic math checker, an exact-match grader), and update on the resulting reward signal. When that loop is run for more than one round, with each round's policy generating the training data (or the verified subset of it) the next round trains on -- the pattern followed explicitly by STaR (Zelikman et al., 2022), ReST-EM (Singh et al., 2023), Self-Rewarding Language Models (Yuan et al., 2024) and, in its most extreme form, Absolute Zero (Zhao et al., 2025), which trains with zero externally supplied data at all -- the system is simultaneously an instance of "RL fine-tuning on a new task," the object RL's Razor describes, and an instance of "recursive training on self-generated data," the object model collapse describes. It is not one or the other by construction; the categories were built for different questions and this system sits in the intersection whether or not anyone measures it there.

This paper does not run that measurement -- no experiments were performed for this paper, in keeping with its genre and the evidentiary standard that follows from it. What it does is narrower and, this paper argues, still useful: it states precisely what each literature's central claim does and does not cover (Sections 2 and 4); assembles the counter-evidence inside each literature that complicates its own headline claim, most of it published in 2025 and 2026 and none of it, as far as this sweep found, cited against the other literature's claim (Sections 3 and 5); lays out five concrete self-training systems and marks, for each, which of the two failure modes it has actually been checked for (Section 6); separates the retention question and the collapse question along three axes -- reference point, timescale and what is lost -- on which the two literatures' instruments are not interchangeable (Section 7); and proposes a candidate mechanism, distributional narrowing toward the policy's own existing modes, that could in principle produce both a short-horizon retention advantage and a long-horizon collapse liability from the same underlying dynamic, while being explicit that no cited study tests this mechanism directly (Section 8). Section 9 examines the one component -- the verifier -- that both literatures treat as capable of holding the loop open, and reports evidence that verifier reliability is not a fixed property of a self-training loop but a variable one. Section 10 gives the decision structure the paper's argument rests on as pseudocode, and Section 11 specifies the joint measurement this paper's central claim implies is missing. Section 12 states the strongest case against this paper's own framing before Section 13 closes with limitations.

2. What "Retention" Means in the RL-vs-SFT Literature

Shenfeld, Pari and Agrawal (2025) construct a scoped, mechanistic claim, and the scoping matters for

everything that follows. The comparison is between reinforcement learning and supervised fine-tuning, both applied to a base policy learning one new task, evaluated on a fixed suite of prior tasks the fine-tuning run never touches. Their central quantity is forward KL divergence between the fine-tuned policy and the base policy, restricted to the distribution induced on the new task; they report that this quantity predicts the degree of forgetting on prior tasks with R-squared of 0.96 in a controlled ParityMNIST setting and 0.71 across a set of language-model fine-tuning experiments -- a strong but not perfect fit; an R-squared of 0.71 leaves roughly three-tenths of the variance in forgetting unexplained by KL divergence alone, though the paper's own text was not accessed during this sweep to confirm whether it discusses that residual directly, and this paper's phrasing of it should be read as this paper's own gloss on the reported number, not as a quoted claim. On a representation-level probe -- similarity of Wikipedia-paragraph embeddings to the base model's embeddings, measured by CKNN -- RL-tuned models retain 0.94 similarity to the base model versus 0.56 for SFT-tuned models reaching comparable new-task accuracy, a gap the paper attributes to the same mechanism: on-policy RL can only reinforce outputs the base policy already assigns non-negligible probability to, so the set of policies reachable by RL from a given base model is implicitly constrained to a KL-ball around it, whereas SFT's training signal comes from a fixed dataset with no such constraint built in.

The mechanism the authors propose, RL's Razor, is specifically a claim about single-task fine-tuning against a fixed base model. It says nothing, on its own terms, about what happens if the "new task" is itself the product of many earlier rounds of the same policy's own outputs, because that scenario was not part of the experimental design. Sparsity Curse (Wu et al., 2026) offers a complementary mechanistic account at the parameter level: RLVR updates are reported to be sparse and off-principal in parameter space relative to SFT updates on the same task, which is consistent with -- though not identical to -- the KL-minimality story, since a sparse update to a small subset of parameters is one concrete way a policy could stay close to its base model in output distribution while still solving a new task.

Two features of this literature's evidentiary base are worth naming before the counter-evidence in Section 3. First, "prior knowledge" in this literature is operationalized as performance on a held-out task suite the model was never trained on during the episode being studied, or as representational similarity to a snapshot of the model taken before that episode began. The reference point is always a specific earlier version of the same model. Second, every experiment surveyed in this section studies a single fine-tuning episode: base model, one new task, one comparison of RL against SFT. None of the cited experiments run the policy through multiple further rounds of self-generated training and re-measure retention at each round against the original base model. The claim that RL forgets less is, as stated, a claim about the first step of a process that self-training loops iterate many times, and the paper's own evidence does not speak to the second step, the third, or the hundredth.

3. The Counter-Evidence: RLVR Forgets Too, Under the Conditions That Matter

If RL's Razor were the whole story, RLVR pipelines would not need retention-specific machinery, and yet a cluster of 2025-2026 papers builds exactly that machinery because the pipelines they study forget measurably. Most of this cluster trains with Group Relative Policy Optimization (GRPO), the RL algorithm introduced by Shao et al. (2024) for DeepSeekMath and subsequently the default optimizer for open RLVR pipelines generally, including the ones this section discusses, so its name recurs through the rest of this paper without further citation. Qin et al. (2026) frame the forgetting problem directly as correct-set turnover: previously solved problems become unsolvable again as RLVR training proceeds, a phenomenon distinct from the SFT-versus-RL comparison because it occurs within a pure RLVR run with no SFT baseline in the picture at all. Their measurement on Qwen3-VL-8B-Instruct, over the first

200 training steps under standard GRPO, finds approximately 190 previously-mastered items regress to unsolved; a retention-aware review mechanism the paper introduces reduces this to approximately 155 over the same window, narrowing but not eliminating the effect. The paper's repair-window principle -- that the cost of restoring a regressed item grows sharply with review delay -- implies that standard RLVR pipelines, which do not track which items have regressed, are systematically forgoing a low-cost window in which the loss is cheap to reverse.

Liu et al. (2026), studying continual multimodal post-training, state the tension explicitly: reinforcement fine-tuning "is widely believed to inherently resist catastrophic forgetting," and their own measurements show that belief holding only conditionally -- "under pronounced task distributional shifts... forgetting across representative RFT algorithms escalates sharply." Their diagnosis attributes this to the limits of RFT's implicit reward-variance regularization, which the paper finds insufficient to suppress what it calls uncontrolled optimization risk once the shift between the old and new task distributions grows large enough. Their proposed fix, explicit risk-aware policy scaling and batch stratification, reduces final forgetting by 79.8 percent relative to an RLOO backbone on the public MLLM-CL benchmark -- a large relative improvement that is, by the same token, an admission of how much the unmodified RFT baseline forgets when the task shift is pronounced. RL's implicit KL-minimal bias, in other words, is a real effect at the scale and shift magnitude Shenfeld et al. (2025) study, and an insufficient one once the shift magnitude grows, exactly the regime a many-generation self-training loop is liable to enter as its own outputs drift from the policy's original training distribution.

A second thread ties RLVR forgetting to diversity rather than to task-level accuracy. Li et al. (2025) report that RLVR training commonly degrades multi-attempt performance (Pass@k) even as single-attempt accuracy (Pass@1) improves, and attribute this to the divergence term in the RL objective: reverse-KL objectives, the paper argues, actively narrow the policy and so accelerate the loss of alternative correct solution paths, while omitting a divergence term entirely removes any safeguard against the same drift. Their proposed alternative, using a mass-covering forward-KL or JS-divergence term as an explicit rehearsal mechanism against the initial policy, is reported to resolve the Pass@k degradation in their math and SQL-generation experiments. Zeng et al. (2025) report a related dynamic under the name RLoop: standard RLVR optimization is described as driving policy over-specialization and "catastrophic forgetting of diverse solutions generated during training," with the useful diversity generated mid-training discarded by the time training ends rather than retained. Cui et al. (2025) supply a mechanistic account for why this narrowing happens at all: policy entropy collapses sharply early in RL training across the runs they examine, tracked by an empirical relationship of the form $R = -a * e^H + b$ between entropy H and downstream performance R , with performance saturating precisely as entropy is depleted. None of these three papers frame their result as being about catastrophic forgetting of prior, off-task knowledge in Shenfeld et al.'s sense -- they are about forgetting alternative correct solutions to the task being trained on -- but the mechanism they describe, a policy narrowing toward a small set of high-reward modes at the expense of everything else it could produce, is the same kind of narrowing RL's Razor credits with the retention advantage, applied to a different object (solution diversity within a task, rather than performance on tasks outside it).

A related but distinct forgetting channel appears whenever RLVR is chained after SFT rather than run alone, which is the common industrial recipe rather than the exception: Liu et al. (2025), building ViSurf for vision-language models, and Yuan et al. (2026), analyzing SFT and RLVR through the lens of task vectors, both report catastrophic forgetting specifically at the SFT-to-RLVR seam, when a model already fine-tuned on human data is then handed to an RLVR stage. This is a different case from Section 2's single-stage RL-versus-SFT comparison -- here RL is not being compared against SFT on the same starting point, it is being applied on top of an SFT stage's own drift -- and it is a further reminder that "RL forgets less" was measured against a clean base model, not against a model that has already

accumulated an SFT stage's distributional shift, a composition that self-training loops built on already-instruction-tuned checkpoints will typically contain. A parallel diversity-side result from the adjacent RLHF literature, not RLVR specifically, reinforces the general pattern: Kirk et al. (2023) report that RLHF fine-tuning narrows output diversity relative to the base model even as it improves task-specific reward, across the generalisation and diversity metrics their paper tracks, which is the RLHF analogue of the Pass@k and entropy results Section 8 discusses for RLVR.

Read together, this section's papers do not contradict Shenfeld et al. (2025) on the specific comparison that paper makes: nothing here disputes that RL forgets less than SFT does, holding the new task and its shift from the base model's distribution fixed. What they establish is that "RL forgets less than SFT" is not the same claim as "RL does not forget," and that the conditions under which the gap between those two claims opens up -- large distributional shift, extended training, and diversity measured within the training task rather than accuracy measured outside it -- are conditions a self-training loop is positioned to reach simply by running for more than one generation.

4. What "Collapse" Means in the Model-Collapse Literature

The model-collapse literature's central claim is about a different axis entirely: not whether a model retains performance on tasks it is not being trained on, but whether the population of outputs a model can still produce continues to cover the distribution the model was originally trained to approximate. Shumailov et al. (2023) formalize this with theoretical models -- variational autoencoders, Gaussian mixture models -- and demonstrate the same qualitative pattern in language models: when each generation of a model is trained on samples drawn from the previous generation, with no persistent supply of fresh, real data, the tails of the original distribution are progressively lost, and the process is described as irreversible once the training data has been fully replaced by model-generated content across enough generations.

Alemohammad et al. (2023) generalize the mechanism and name it Model Autophagy Disorder (MAD), studying three families of "autophagous" training loops that differ in how much fixed or fresh real data remains available at each generation, and in whether the loop biases sampled data toward quality or toward diversity. Their central conclusion is a structural trade-off rather than a single failure mode: without enough fresh real data at each generation, a self-consuming loop is described as "doomed" to progressively lose either quality (measured as precision) or diversity (measured as recall) -- the loop can be biased to preserve one at the expense of the other, but a loop with insufficient fresh data cannot preserve both indefinitely. Dohmatob et al. (2024) sharpen the severity claim with a scaling-law analysis: as little as one percent synthetic data in an otherwise real training corpus is sufficient to trigger measurable collapse, and the paper's key negative result is that adding more total training data does not rescue the outcome once contamination has occurred -- the pathology is set by the contaminated fraction, not washed out by scale.

As with Section 2, two scoping features matter for the rest of this paper. The reference point in this literature is the original real-data distribution as a whole -- specifically its tails, the rare or unusual examples that a finite sample under-represents -- not a snapshot of any particular earlier model. And the timescale is generational: the pattern is defined across repeated rounds of training-on-the-previous-round's output, not within a single fine-tuning episode. A single round of fine-tuning a model on its own generations, evaluated once, is not by itself an instance of the phenomenon these papers name model collapse; it is the necessary ingredient the phenomenon accumulates across many repetitions of.

5. The Counter-Evidence: Collapse Is Conditional, Not Categorical

Model collapse's headline framing -- that recursive training on self-generated data leads to inevitable degradation -- is itself qualified within the same literature, in ways relevant to whether a verifier-gated

self-training loop is protected from the failure mode by construction. Gerstgrasser et al. (2024) report a result with a title that states its own conclusion: model collapse, under their analysis, is not inevitable. The paper's key distinction is between replacing each generation's training data with the new generation's synthetic output (the setup that produces Shumailov et al.'s collapse) and accumulating synthetic data alongside the original real data rather than discarding it. Under accumulation, the paper reports, test error has a finite upper bound independent of the number of training iterations, across language models, diffusion models and variational autoencoders in their experiments; under replacement, test error grows with the number of iterations without such a bound. The practical implication the paper draws is that whether collapse occurs is a property of the data-management policy governing a self-training loop, not an unconditional property of training on self-generated data as such.

The mitigation literature this qualification belongs to is larger than the two papers above, and it is worth naming its range briefly because the diversity of proposed fixes is itself evidence that no single mitigation is yet treated as settled. Zhu et al. (2024) trace collapse in synthesized text data to over-concentration of the token distribution and propose token-level editing of synthetic data as a mitigation; Drayson et al. (2025) propose filtering machine-generated text out of future training corpora via detection classifiers, a mitigation aimed at the web-scale pretraining version of the same problem rather than at a single lab's self-training loop; Cai et al. (2025) and Wei and Zhang (2025) both target the image-generation instance of self-consuming loops, via latent-space filtering and adversarially curated data selection respectively; and Khelifa et al. (2026) extend the theoretical error-propagation analysis specifically to diffusion models under mixed synthetic-and-fresh-data pipelines. Xie and Hu (2026) survey this mitigation literature directly and, notably for this paper's premise, organize it entirely within the model-collapse framing -- their review does not cross into the RL-forgetting literature of Sections 2-3 at any point, which is itself a small piece of evidence, alongside Tao et al.'s (2024) self-evolution survey noted in Section 12, that the reconciliation this paper argues is missing has not been attempted by a survey on either side.

A second qualification concerns whether a correction or verification step inside the loop can substitute for fresh real data. Gillman et al. (2024) study self-correcting self-consuming loops directly and report that training instability or collapse can be stabilized under conditions their paper specifies, using a self-correction mechanism rather than an injection of fresh real data -- evidence that a sufficiently reliable in-loop check, not only an external data supply, can hold a self-consuming loop open. Fu et al. (2024) supply a theoretical account of the same family of loops, deriving bounds on how far the learned distribution can drift (in total variation distance) from the true distribution under recursive training on mixtures of real and synthetic data, formalizing the intuition that the rate of drift depends on the mixture ratio and not only on whether any real data is present at all.

This second qualification, however, is not unconditional either, and the paper that complicates it most directly for the argument here is Qiao et al. (2026), who study what happens when the selection or verification mechanism relied upon to keep a self-consuming loop safe is itself imperfect. Their finding is that in low-resource verification regimes -- where a verifier observes only a small, fragmented or biased slice of the target manifold, precisely the situation a narrow, task-specific verifier such as a unit test or an exact-match grader is in relative to the full space of a model's possible outputs -- data selection intended as a remedy for collapse instead becomes a vector for it: the verifier's own blind spots get selected for, compounding across generations rather than being corrected by them. This is a direct complication of the premise, common across the self-training systems in Section 6, that a reliable external verifier is what distinguishes a safe self-training loop (an RLVR pipeline scored by a unit test or a symbolic checker) from the unsafe generative-modeling loops Shumailov et al. and Alemohammad et al. study (an image or text model trained on its own unfiltered output with no correctness check at all).

Section 9 returns to this point directly, because it is the single most load-bearing assumption connecting the two literatures' opposite prognoses.

6. Same Loop, Two Vocabularies: Five Systems, Neither Probe Run on Any of Them

The claims in Sections 2 through 5 are abstract until they are checked against systems that actually exist. Five are surveyed here, chosen because each is a documented, named self-training recipe rather than a hypothetical, and because together they span the design space from lightly RL-flavored filtered self-training to full multi-round RL on a model's own generations with no external data at all.

STaR (Zelikman et al., 2022) bootstraps a chain-of-thought reasoner by generating rationales, keeping only those that lead to a correct final answer, and fine-tuning on the kept set -- repeated over multiple rounds. It is filtered self-training rather than RL in the policy-gradient sense, but the loop structure (generate, verify, train on the verified subset, repeat) is the one Section 8 argues carries the narrowing mechanism.

ReST-EM (Singh et al., 2023) formalizes a similar expectation- maximization-style self-training loop for math and code problems with verifiable answers, explicitly comparing against continued human-data fine-tuning and reporting that self-training on model-generated, verifier-filtered solutions can outperform training on additional human data at fixed model scale.

SPIN (Chen et al., 2024) frames self-play as an adversarial game between successive iterations of the same model, using no additional human-annotated data beyond an initial SFT dataset, and is explicitly positioned as a way to keep improving a language model "without the need for acquiring additional human-annotated data" -- the recursive, external-data-free property Section 4's papers treat as a precondition for collapse.

Self-Rewarding Language Models (Yuan et al., 2024) remove the external verifier that STaR, ReST-EM and typical RLVR pipelines rely on entirely: the same model generates candidate responses and judges them via LLM-as-a-judge prompting, then trains on the resulting preference pairs. This is the system in the survey most exposed to the verifier-reliability concern of Section 9, because its judge is not an independent check but the same weights, under the same training pressure, that produced the outputs being judged -- and Wataoka et al. (2024) document self-preference bias in LLM-as-a-judge setups generally: a judging model measurably favors outputs in its own style over comparably good outputs in a different style, a bias with an obvious compounding channel in a loop where the judge is retrained each round on data its own earlier preferences selected. A closely related family, iterative preference optimization over the model's own generated reasoning chains, is represented by Pang et al. (2024), whose Iterative Reasoning Preference Optimization trains repeated rounds of DPO on competing self-generated chain-of-thought candidates, selecting the winning versus losing reasoning path by whether it reaches the correct final answer. This occupies a middle position between STaR's hard filter (keep the correct rationale, discard the rest) and Self-Rewarding's fully learned judge (no external correctness signal at all): the correctness check is external and verifiable, as in STaR, but the training signal is a preference pair rather than a single accepted example, which changes what regresses when a round goes wrong without changing whether the loop's underlying object is self-generated data.

Absolute Zero (Zhao et al., 2025) is the limiting case: the policy proposes its own tasks, solves them, and is scored entirely by a code executor, with zero externally supplied task or answer data at any point. The paper reports overall state-of-the-art performance on coding and math-reasoning benchmarks relative to other zero-setting models that do rely on tens of thousands of human-curated in-domain examples -- a strong result for the premise that a sufficiently reliable verifier (here, a code executor, arguably as close to Qiao et al.'s "low-resource, fragmented verifier" concern as a system in this survey

gets, since a code executor checks correctness of self-proposed tasks against self-proposed criteria, not against an external ground truth) can sustain a fully closed self-training loop.

Table 1 (Section 10) consolidates the measured values behind these claims and the counter-claims of Sections 3 and 5. What is worth stating plainly here, ahead of that table, is a negative result about the survey itself rather than about any one paper in it: none of the five systems above is evaluated, in the source cited for it, on both a fixed prior-task retention suite of the kind Shenfeld et al. (2025) use and a distributional-coverage or diversity metric against the original training data of the kind Shumailov et al. (2023) or Alemohammad et al. (2023) use, at the same set of checkpoints across the same multi-generation run. STaR, ReST-EM and SPIN report task-accuracy improvements across rounds. Absolute Zero reports benchmark scores after training. Self-Rewarding reports win-rate improvements on instruction-following benchmarks across iterations. None reports a held-out retention suite disjoint from the training task, evaluated against the original base model, at every generation; none reports a tail-coverage or diversity metric against the pre-training data distribution at every generation. The absence is not a flaw specific to any one paper -- each measures what its own research question requires -- but the absence is total across the five systems surveyed, which is the empirical basis for this paper's central claim that the joint measurement has not been run.

7. Three Axes on Which the Two Claims Are Not Interchangeable

Before proposing any way the two literatures' claims might connect, it is worth stating precisely why they might not, because the shared vocabulary of KL divergence, entropy and "narrowing" invites a conflation the evidence does not license on its own.

Reference point. RL's Razor and its RLVR-forgetting counter-evidence (Section 3) measure distance from a specific earlier policy -- the base model, or the policy checkpoint before the episode being studied began. Model collapse and its counter-evidence (Sections 4-5) measure coverage of a population-level data distribution, specifically its tails, which is not the same object as any single model's parameters or output distribution at a point in time. A policy can stay close to its own immediately-prior checkpoint at every step (low per-step forgetting by Shenfeld et al.'s metric) while the cumulative drift over many steps still carries it far from the tails of the original data distribution (collapse by Shumailov et al.'s metric), in the same way a random walk with small steps can still travel arbitrarily far from its origin. Nothing in Section 2's evidence bounds the cumulative drift; it bounds only the per-episode drift relative to the immediately preceding checkpoint.

Timescale. Every experiment in Sections 2 and 3 is conducted within a single fine-tuning run or a training process on the order of hundreds of steps (Qin et al.'s 200-step window is representative). Every experiment in Sections 4 and 5 is conducted across multiple full training generations -- new model, new dataset, repeat -- often five to ten generations in the cited experiments. A self-training loop of the kind surveyed in Section 6 is usually described, and evaluated, on the shorter timescale (rounds of ReST-EM, iterations of SPIN, generations of Absolute Zero training within a single paper's compute budget), but nothing in those papers' own framing rules out the possibility that the boundary between "still inside RL's Razor's regime" and "now inside model collapse's regime" is crossed somewhere within the rounds a single paper already runs, just not measured with the right instrument to notice.

What is lost. RL's Razor and its counter-evidence track whether a model can still do something it could do before -- a capability, operationalized as accuracy on a held-out task or similarity to a prior representation. Model collapse and its counter-evidence track whether a population of possible outputs still spans a distribution -- diversity and tail coverage, operationalized as precision/recall trade-offs (Alemohammad et al., 2023), total variation distance (Fu et al., 2024), or test error bounds (Gerstgrasser et al., 2024). A model can lose diversity (narrow toward a small number of high-reward modes) without losing any measured capability, if the capability suite being checked happens to be

covered by the modes the model retains; conversely a model could in principle retain broad output diversity while still losing a specific capability the diversity metric does not happen to probe. These are different failure modes that could correlate strongly in practice -- and Section 8 lays out a candidate reason they might -- but correlation is not established by any cited study, because no cited study measures both on the same system.

Taken together, these three axes are the precise content of this paper's central claim: "RL forgets less" and "self-generated data collapses" are not competing answers to the same question. They are answers to two different questions, asked at two different reference points and two different timescales, about two different kinds of loss. A self-training loop is the system where both questions can be asked simultaneously, and Section 6 establishes that, as far as this sweep found, nobody has asked them simultaneously about the same run.

8. A Candidate Unifying Mechanism: Narrowing Toward Existing Modes

The three axes in Section 7 argue the two claims are not automatically the same claim. They do not argue the two claims are unrelated, and this section lays out the strongest candidate mechanism this sweep found for connecting them -- stated explicitly as a hypothesis this paper is not in a position to test, not as a finding.

The mechanism Shenfeld et al. (2025) credit with RL's retention advantage is that on-policy RL can only reinforce outputs the current policy already assigns meaningful probability to; it cannot manufacture probability mass on outputs the policy would not otherwise produce, the way SFT's gradient on a fixed target distribution can. This is, description for description, a statement that RL training narrows a policy toward its own pre-existing high-probability modes rather than introducing new ones. Cui et al.'s (2025) entropy-collapse account describes the same narrowing from a different angle and at a different grain: policy entropy drops sharply early in RL training as the policy concentrates probability mass on a shrinking set of high-reward completions, with downstream task performance saturating as that concentration completes. Yue et al. (2025) supply the sharpest test of what this narrowing does and does not buy: using pass-at-large-k as a coverage metric, they report that RLVR-trained models outperform their own base models at $k=1$ but that base models overtake RLVR-trained models at large k , across the model families, RL algorithms and math, coding and visual-reasoning benchmarks their paper covers, with coverage and perplexity analyses indicating that the reasoning patterns credited to RLVR training originate from and remain bounded by what the base model could already produce. Their conclusion is explicit and, for this paper's purposes, load-bearing: current RLVR training does not elicit reasoning patterns the base model did not already have latent capacity for; it reweights the base model's own distribution toward the patterns that pay off under the reward signal. Shao et al. (2025) provide a complementary result from the reward side of the same mechanism: RLVR training under GRPO with a spurious, randomly assigned reward improves Qwen2.5-Math-7B's MATH-500 accuracy by 21.4 percentage points, nearly matching the 29.1-point gain from a ground-truth reward on the same benchmark and model -- evidence that a sizeable share of what RLVR training accomplishes on this benchmark is attributable to sharpening the policy toward completions it already favored, rather than to information carried specifically by the correctness signal. Yan et al. (2026) follow this thread mechanistically, reporting that spurious-reward RLVR training activates what they term memorization shortcuts and produces a measurable "Perplexity Paradox," in which answer-token perplexity drops while prompt-side coherence degrades -- a signature, on their reading, of the model bypassing reasoning in favor of reproducing memorized patterns rather than of the reward signal teaching anything new. Chen et al. (2025) locate the same narrowing dynamic at the level of the RLVR training objective itself, arguing that clipping and entropy-minimization mechanisms common to RLVR algorithms suppress exploration in a way that interacts with, rather than being independent of, the spurious-reward effect

Shao et al. (2025) and Yan et al. (2026) document. None of these three papers claims the narrowing they observe is identical to the collapse mechanism of Section 4; the claim made here is narrower and is the paper's own synthesis: the direction of the effect -- probability mass concentrating on a shrinking set of modes already favored before training began -- is the same direction in both literatures, whatever the eventual difference in magnitude or reversibility turns out to be.

The idea that training a model on its own outputs changes the effective objective being optimized, rather than only changing the data, is older than the RLVR or model-collapse literatures and has a clean theoretical precedent worth naming. Mobahi et al. (2020) study self-distillation -- retraining a model on its own predictions, repeated for a small number of rounds -- in a Hilbert-space regression setting, and report that the procedure provably amplifies an implicit regularization already present in the original training objective, progressively sparsifying the fitted function in the basis that regularization penalizes, rather than adding new information at each round. That result predates both literatures surveyed in this paper by several years and was not framed as being about forgetting or about collapse, but it is a formal instance of exactly the claim Section 8 needs: retraining a model on its own output is not information-neutral, it systematically concentrates the model toward whatever its existing objective already favored, in a regression setting simple enough to prove.

If RLVR training is best described, per this cluster of results, as narrowing a policy toward a subset of modes it already had non-negligible probability on, then the same narrowing plausibly explains two things at once, at two different timescales, which is the connection this paper's title asks about. At the timescale of one fine-tuning episode, narrowing toward existing modes is close in output distribution to the base model by construction -- which is exactly the low-KL, high-retention outcome Shenfeld et al. (2025) measure and credit to RL. At the timescale of many self-training generations, the same narrowing, applied recursively to a policy's own outputs rather than to a fixed base model, is a plausible description of what a self-consuming loop's "quality" side of Alemohammad et al.'s (2023) precision/recall trade-off looks like from the inside: each generation reinforces the modes the previous generation already favored, at the expense of the modes it did not, which is a mechanism for progressively narrowing the tails of the output distribution rather than an accident that only shows up when real data is entirely removed.

This mechanism, if it holds, would predict that a self-training loop's retention advantage over SFT and its eventual susceptibility to collapse-like diversity loss are not opposing outcomes to be traded off, but the same narrowing dynamic observed at different generation counts -- retention-favorable early, collapse-favorable late, with no cited study establishing where the crossover sits or whether it is sharp or gradual. That prediction is stated here as exactly that: a prediction, assembled from results that were not designed to test it and that this paper has not tested either. Two pieces of evidence complicate it in the direction of caution rather than confirmation. First, Gerstgrasser et al.'s (2024) accumulation result shows that the presence of a fixed reservoir of real data changes the qualitative outcome from unbounded to bounded drift, which the narrowing-toward-existing-modes mechanism on its own does not predict unless the mechanism is understood as operating only on the synthetic share of the training mixture -- a refinement the cited evidence does not directly test either. Second, RLVR's verifier is not the same object as "more real data" in Gerstgrasser et al.'s sense; it filters the policy's own outputs rather than supplying independent samples from the original distribution, which is precisely the distinction Section 9 argues matters for whether a verifier can play the role Gerstgrasser et al.'s fresh real data plays.

9. Verifiers Are Not Uniformly Reliable, and That Cuts Both Ways

The premise that makes RLVR self-training loops seem categorically safer than the unfiltered self-consuming image and text loops Section 4 describes is that RLVR has a verifier -- a unit test, a symbolic

math checker, a code executor -- standing between the policy's raw output and what gets trained on, filtering out the errors that would otherwise accumulate. This section states what the cited evidence actually supports about that premise, which is narrower than the premise as usually stated.

Where the premise holds up best is where the verifier checks a formally specified, narrow property with no ambiguity: Absolute Zero's code executor (Zhao et al., 2025) checks whether proposed code executes and produces the claimed output, a check with no room for the verifier itself to be wrong given a correctly specified task. Where the evidence complicates the premise is everywhere the verifier's coverage of the space it is meant to police is itself incomplete or biased. Qiao et al. (2026) give this the sharpest formal treatment among the papers surveyed here: in what they term low-resource verification regimes, where a verifier observes only a small, fragmented or biased slice of the target manifold, a selection mechanism intended to prevent collapse is reported to instead precipitate it, because the verifier's blind spots are systematically selected for rather than corrected across generations. A narrow, task-specific RLVR verifier -- a grader for one class of math problem, a test suite covering one slice of a coding domain -- is a plausible instance of exactly this regime relative to the full space of outputs a language model can produce, even though it is not the regime Qiao et al.'s own experiments were run in, so the connection drawn here is an application of their result's shape to a case they did not test directly, offered as a hypothesis and not confirmed by their paper.

Self-Rewarding Language Models (Yuan et al., 2024) present a second and distinct verifier-reliability concern, because their "verifier" is not an external check at all but the policy's own LLM-as-a-judge evaluation of its own outputs. Wataoka, Takahashi and Ri (2024) document self-preference bias in LLM-as-a-judge setups generally -- a judging model measurably favoring outputs stylistically closer to its own than to a comparably good alternative -- and in a loop where the judge is periodically retrained on data its own prior judgments selected, this is a channel by which judge error compounds rather than washes out across rounds, a dynamic with no analogue in a code-executor-verified loop and no analogue in Gerstgrasser et al.'s (2024) fresh-real-data setup, since fresh real data has no mechanism to inherit bias from the model being trained.

Skalse et al. (2022) supply the general vocabulary for why an imperfect verifier is a liability rather than a neutral simplification: a proxy objective is "hackable" whenever it is possible to increase the proxy's expected value while decreasing the true objective's expected value, and the paper's formal treatment implies that any verifier short of a perfect, exhaustive check of the true objective is hackable in principle, a question of degree rather than kind. Gao et al. (2022) and Rafailov et al. (2024) document this degree empirically in the adjacent setting of RLHF reward models, reporting that optimizing against a learned proxy reward improves measured proxy performance while true, gold-standard performance degrades past a certain point of optimization -- Goodhart's law operating on a reward signal that, like an RLVR verifier, stands between raw model output and what gets reinforced. RLVR verifiers of the code-executor or symbolic-checker variety are typically closer to ground truth than a learned RLHF reward model is, which is presumably part of why RLVR's reputation for resisting reward hacking is stronger than RLHF's -- but "closer to ground truth" is not the same claim as "not hackable," and Qiao et al.'s (2026) low-resource-regime result is direct evidence that even a correctness-checking verifier can be an incomplete one relative to the full space a self-training loop's policy is free to explore.

None of this establishes that any specific named RLVR self-training system in Section 6 is, in fact, accumulating verifier-blind-spot bias across generations; it establishes that the assumption doing the load-bearing work in "RLVR is safe from collapse because it has a verifier" is an empirical claim about that specific verifier's coverage, not a property that follows automatically from a verifier's presence, and that the one paper in this survey built to test the assumption directly (Qiao et al., 2026) finds it fails in the regime a narrow, task-specific verifier is arguably already in.

10. The Decision Structure Nobody Runs

Table 1 consolidates the measured values this paper's argument rests on, one row per cited result. Algorithm 1 is this paper's own synthesis, not an empirical result and not reproduced from any cited source: it states the generic decision structure a self-training loop follows, and marks where a retention probe (Section 2's instrument) and a coverage probe (Section 4's instrument) would need to be inserted for a single run to answer both literatures' questions at once.

Table 1. Measured values behind this paper's argument, one row per cited result.

#	Result	Measured value	Source
1	Representational similarity to base model (Wikipedia paragraph embeddings, CKNNA), RL vs. SFT, comparable new-task accuracy	RL 0.94; SFT 0.56	Shenfeld, Pari and Agrawal, 2025
2	KL divergence from base policy predicts degree of forgetting (variance explained)	R-squared = 0.96 (ParityMNIST); R-squared = 0.71 (language-model tasks)	Shenfeld, Pari and Agrawal, 2025
3	RLVR gain on MATH-500, Qwen2.5-Math-7B, GRPO, spurious (random) vs. ground-truth reward	+ 21.4 pp (random reward); + 29.1 pp (ground-truth reward)	Shao et al., 2025
4	Correct-set turnover: previously-solved items regressing to unsolved, first 200 training steps, Qwen3-VL-8B-Instruct, standard GRPO	approximately 190 items regressed	Qin et al., 2026
5	Same setup, with a retention-aware review mechanism added	approximately 155 items regressed	Qin et al., 2026
6	Final forgetting under continual reinforcement fine-tuning, MLLM-CL benchmark, explicit risk-aware method vs. RLOO backbone	79.8 percent relative reduction	Liu et al., 2026
7	Fraction of synthetic data in an otherwise-real training corpus sufficient to trigger measurable model collapse	as low as 1 percent	Dohmatob et al., 2024
8	Test error across recursive training generations: data replaced each generation vs. accumulated alongside original real data	unbounded growth (replace); finite upper bound (accumulate)	Gerstgrasser et al., 2024
9	Pass-at-large-k, RLVR-trained model vs. its own base model, math/code/visual-reasoning benchmarks	base model exceeds RLVR-trained model at large k despite trailing at k = 1	Yue et al., 2025
10	Relationship between policy entropy and downstream RL training performance	$R = -a * e^H + b$ (performance saturates as entropy collapses)	Cui et al., 2025
11	Quality/diversity trade-off in self-consuming image-model loops without sufficient fresh real data per generation	at least one of precision or recall provably decreases	Alemohammad et al., 2023

12 Self-training with zero externally supplied task or answer data, code-executor verifier, coding and math benchmarks	reported overall SOTA vs. zero-setting models trained on tens of thousands of human-curated examples	Zhao et al., 2025
--	--	-------------------

Algorithm 1. Generic self-training / RLVR loop, with the two probes neither literature runs jointly. This is the author's own decomposition of the loop structure common to Sections 2-6, offered to make the argument's structure checkable, not a result from any cited paper.

```

Input:  base policy pi_0
        seed task/prompt distribution D_0
        verifier V (unit test, symbolic checker, code executor,
                    LLM-as-judge, or none)
        prior-capability probe suite P, disjoint from D
        reference sample R_0 of the original, pre-loop data distribution
Output: policy pi_N after N generations

for g in 1..N:
    # 1. Generate: sample candidate solutions from the current policy
    C_g <- sample(pi_(g-1), D_(g-1))

    # 2. Score: accept, rank, or reward the candidates via V
    #   STaR / ReST-EM:      correctness filter, keep-if-correct
    #   SPIN:                adversarial self-play score
    #   Self-Rewarding:      policy's own LLM-as-judge score
    #   Absolute Zero:       code-executor pass/fail
    S_g <- V(C_g)

    # 3. Update: train on the scored self-generated data
    #   RL (GRPO / PPO-style) or filtered SFT on the accepted subset
    pi_g <- update(pi_(g-1), C_g, S_g)

    # 4. Advance the data pool for the next generation
    #   REPLACE (D_g = C_g) vs ACCUMULATE (D_g = D_(g-1) union C_g)
    #   is exactly Gerstgrasser et al.'s (2024) unbounded-vs-bounded
    #   distinction (Table 1, row 8), and it is a design choice most
    #   named systems in Section 6 do not report making deliberately.
    D_g <- select_or_accumulate(D_(g-1), C_g)

    # --- PROBE A (Section 2's instrument): retention ---
    # Compares pi_g to the ORIGINAL BASE MODEL pi_0 on tasks OUTSIDE D.
    # This is what "RL forgets less than SFT" is a claim about.
    retention_g <- eval(pi_g, P) vs eval(pi_0, P)

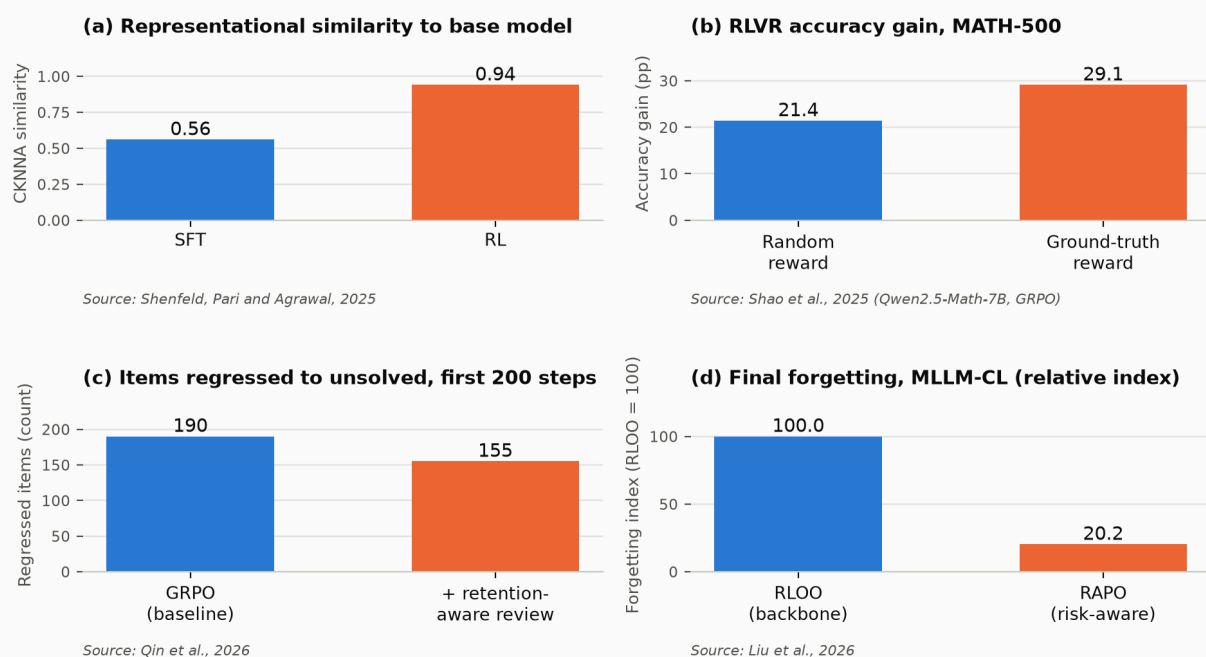
    # --- PROBE B (Section 4's instrument): coverage / collapse ---
    # Compares the POPULATION of pi_g's outputs to the tails of the
    # ORIGINAL DATA DISTRIBUTION R_0, not to any single prior model.
    # This is what "self-generated data collapses" is a claim about.
    coverage_g <- diversity_and_tail_coverage(pi_g, R_0)

    # No system surveyed in Section 6 logs both retention_g and
    # coverage_g at every g against the same run. This paper's claim
    # is exactly that this gap, not either probe's result alone, is
    # what is missing from the published record as of this sweep.

```

The table and the algorithm are meant to be read together. Every row in Table 1 is a measurement of either Probe A (rows 1, 2, 4, 5, 6, 9, 10) or Probe B (rows 7, 8, 11), or a system-level result relevant to whether the loop's data-management policy (row 8) or verifier (row 12, and Section 9) is positioned to keep either probe's outcome favorable. No row in the table is a joint measurement of both probes on the same run at the same checkpoints, because, as far as this sweep found, no published study reports one.

Figure 1. Four independent measurements of the same pattern: staying closer to a reference point (a prior model, a ground-truth signal, an unregressed item set) tracks with less measured loss, across four unrelated metrics and systems



11. What Would Settle This

The measurement this paper's argument implies is missing has a specific shape, distinct from simply "running more RLVR experiments." It requires, on a single self-training system run for enough generations to plausibly enter the regime Section 5's counter-evidence describes:

First, a fixed, held-out capability probe suite disjoint from the training task, of the kind Shenfeld et al. (2025) use, evaluated against the original base model at every generation, not only at the end of training -- so that a retention curve, not a single before/after comparison, is available.

Second, a fixed reference sample of the pre-loop training distribution, with a diversity or tail-coverage metric evaluated against it at every generation, of the kind Alemohammad et al. (2023) or Fu et al. (2024) use -- so that a coverage curve is available on the same timescale as the retention curve, from the same run, not from a separately reported experiment on a different system.

Third, an explicit record of the loop's data-management policy (replace versus accumulate, in Gerstgrasser et al.'s 2024 terms) and of the verifier's coverage properties, ideally measured directly rather than assumed, in the spirit of Qiao et al.'s (2026) low-resource-regime diagnostic -- so that if retention and coverage diverge, the divergence can be attributed to a specific, checkable property of the

loop rather than left as an unexplained difference between systems.

Fourth, enough generations for the run to plausibly cross whatever threshold separates the short-horizon regime Section 2's evidence covers from the long-horizon regime Section 4's evidence covers -- a threshold this paper does not know the location of and flags as itself an open empirical question, not a known constant that a study need only exceed.

No paper surveyed here runs all four. STaR, ReST-EM and Absolute Zero report multiple generations without a fixed retention probe or a diversity-against-original-data metric. Shenfeld et al. (2025) and the RLVR-forgetting papers in Section 3 report a retention probe without running enough generations, or without the diversity metric, to speak to collapse. Alemohammad et al. (2023) and Gerstgrasser et al. (2024) report a diversity or coverage metric across many generations without a held-out-capability retention probe. What is missing is not another instance of any one of these measurements; it is one paper running all of them on one system.

12. The Strongest Case Against This Paper's Own Framing

The most direct objection to this paper's premise is that the puzzle it poses may be a category error rather than an open question: if RL's Razor's mechanism (narrowing toward the base policy's own high-probability modes) is fundamentally protective of the base model's original distribution, then a self-training loop that keeps reapplying RLVR to its own outputs might simply never leave the KL-ball around the original base model that RL's Razor describes -- in which case there is no long-horizon collapse regime for a pure RLVR self-training loop to enter, and the puzzle this paper poses would dissolve rather than resolve. Nothing cited here refutes this version of events outright. Yue et al.'s (2025) pass-at-large-k result is consistent with it: if RLVR training never expands a model's coverage beyond what the base model could already produce, one reading is that it also never contracts that coverage below some floor set by the base model's own diversity, which would be a reassuring rather than an alarming interpretation of the same finding this paper cites as counter-evidence in Section 8.

Two considerations keep this paper from adopting that reassuring reading as the default. First, Yue et al.'s (2025) own result shows RLVR-trained models losing pass-at-large-k coverage relative to their base model at large k, which is itself a measured narrowing of the base model's original diversity, not merely a failure to expand it -- the paper's finding is compatible with "RLVR neither expands nor is protected from contracting" rather than with "RLVR is protected from contracting." Second, and more directly, the pass-at-large-k evidence is drawn from single-generation RLVR studies, the same short-horizon limitation Section 7 identifies in the retention literature generally; it says nothing about whether repeated re-application of RLVR to a policy's own prior-round outputs, rather than to a fixed base model, compounds this narrowing across rounds the way Alemohammad et al.'s (2023) and Shumailov et al.'s (2023) recursive setups do. The reassuring reading and this paper's central claim are both consistent with the cited evidence; adjudicating between them is exactly the joint, multi-generation measurement Section 11 specifies and that, per Section 6's survey, has not been run.

A second, narrower objection concerns scope: this paper's survey of named self-training systems (Section 6) is five systems, not an exhaustive census of the 2022-2026 self-improvement literature, and the survey of self-evolution approaches by Tao et al. (2024) documents a considerably larger space of methods this paper does not individually examine. It is possible some system outside this paper's five does report both a retention probe and a coverage probe jointly; this paper's claim is scoped to what this sweep found, and is stated as a claim about the published record as surveyed here, not as a claim of logical impossibility.

13. Limitations

This paper reports no experiments and measured no number itself; every quantitative claim in Table 1 and Figure 1 is drawn from the abstract or a directly stated headline result of the cited source, and this paper did not independently verify any of those results by re-running the underlying experiments. Where a cited paper's abstract did not state a number precisely enough to quote (for instance, Shumailov et al.'s 2023 qualitative description of tail loss, or several of the self-consuming-loop stabilization papers in Section 5), this paper reports the qualitative claim rather than inventing a figure, and Table 1 excludes rows for which no quotable number was found during the sweep.

The central mechanism proposed in Section 8 -- that a single narrowing dynamic could explain both a short-horizon retention advantage and a long-horizon collapse liability -- is explicitly labeled a hypothesis assembled from results not designed to test it, and this paper has not tested it either. It should not be read as an established finding, and a future study running the joint measurement specified in Section 11 could falsify it cleanly: if retention and coverage curves from the same run do not move together, the shared-mechanism story fails and the two literatures' claims would need a different reconciliation, or none.

The literature sweep behind this paper was conducted through the arXiv API over a bounded session and is not a systematic or exhaustive review; Section 12 already notes that the survey by Tao et al. (2024) documents a considerably larger space of self-evolution methods than the five systems examined directly in Section 6, and it is possible relevant results, particularly recent preprints not yet indexed or not surfaced by the search terms used, were missed. The RLVR-forgetting cluster in Section 3 and the model-collapse-related search results are both weighted toward 2025 and 2026 preprints, several without peer review at the time of this sweep; this paper cites them as the current state of an active research question, not as settled results, and several of their own headline claims (particularly the precise magnitude of forgetting reduction methods in Sections 3 and 5) are specific to the benchmarks and model families those papers study and may not generalize to other settings.

Per the queue constraint this paper was drafted under, RLHF-specific forgetting (as opposed to RLVR and self-training specifically) is treated as background only and is not surveyed with the same depth as Sections 2 through 6; Kirkpatrick et al. (2016) and De Lange et al. (2019) are cited for general continual-learning vocabulary and are not evaluated against the self-training-specific claims this paper makes. Finally, this paper's own title question -- does a model forget differently when the data is its own -- is answered here only in the sense of specifying what a difference would look like and how to measure it; it is not answered in the sense of reporting which regime, retention-dominant or collapse-dominant, a real self-training loop of the kind Section 6 surveys is actually in at any given generation count, because that question is exactly the one this paper argues nobody has measured.

14. Conclusion

RL's Razor and model collapse are both well-evidenced claims within their own scope, and neither is disputed here on the terms it was originally stated in. What this paper disputes is the inference, available to a reader who encounters both literatures separately, that they describe competing or independent phenomena. They describe the same class of system -- a policy trained repeatedly on data close to its own distribution -- at different reference points, different timescales, and different units of loss, using overlapping vocabulary (KL divergence, entropy, narrowing) that makes the two claims look more comparable than the underlying measurements license. Section 6's survey of five named self-training systems finds the joint measurement -- retention against a fixed prior-task suite and coverage against the original data distribution, on the same run, at the same checkpoints -- absent from all five. Section 8's candidate mechanism, distributional narrowing toward a policy's own existing modes, offers one account of how a short-horizon retention advantage and a long-horizon collapse liability could be two readings of a single dynamic, but it is offered as a hypothesis the cited evidence does not test, not

as this paper's finding. Until the measurement in Section 11 is run, the honest answer to this paper's title question is that it is not known whether a model forgets differently when the data is its own, or merely later.

References

- Shenfeld, I., Pari, J. & Agrawal, P. (2025). RL's Razor: Why Online Reinforcement Learning Forgets Less. arXiv:2509.04259. doi:10.48550/arXiv.2509.04259. [PREPRINT]
- Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot, N. & Anderson, R. (2023). The Curse of Recursion: Training on Generated Data Makes Models Forget. arXiv:2305.17493. doi:10.48550/arXiv.2305.17493. [PREPRINT]
- Alemohammad, S., Casco-Rodriguez, J., Luzi, L., Humayun, A. I., Babaei, H., LeJeune, D. et al. (2023). Self-Consuming Generative Models Go MAD. arXiv:2307.01850. doi:10.48550/arXiv.2307.01850. [PREPRINT]
- Dohmatob, E., Feng, Y., Subramonian, A. & Kempe, J. (2024). Strong Model Collapse. arXiv:2410.04840. doi:10.48550/arXiv.2410.04840. [PREPRINT]
- Zhu, X., Cheng, D., Li, H., Zhang, K., Hua, E., Lv, X. et al. (2024). How to Synthesize Text Data without Model Collapse? arXiv:2412.14689. doi:10.48550/arXiv.2412.14689. [PREPRINT]
- Qiao, X., Du, X., Liu, W., Zhang, J., Mai, P. & Zhang, M. (2026). When Sample Selection Bias Precipitates Model Collapse. arXiv:2606.13732. doi:10.48550/arXiv.2606.13732. [PREPRINT]
- Khelifa, N. B., Turner, R. E. & Venkataramanan, R. (2026). Quantifying Error Propagation and Model Collapse in Diffusion Models. arXiv:2602.16601. doi:10.48550/arXiv.2602.16601. [PREPRINT]
- Xie, X. & Hu, B. (2026). Reviewing Model Collapse and Countermeasures. arXiv:2608.21366. doi:10.48550/arXiv.2608.21366. [PREPRINT]
- Drayson, G., Yilmaz, E. & Lamos, V. (2025). Machine-generated text detection prevents language model collapse. arXiv:2502.15654. doi:10.48550/arXiv.2502.15654. [PREPRINT]
- Gerstgrasser, M., Schaeffer, R., Dey, A., Rafailov, R., Sleight, H., Hughes, J. et al. (2024). Is Model Collapse Inevitable? Breaking the Curse of Recursion by Accumulating Real and Synthetic Data. arXiv:2404.01413. doi:10.48550/arXiv.2404.01413. [PREPRINT]
- Gillman, N., Freeman, M., Aggarwal, D., Hsu, C.-H., Luo, C. & Tian, Y. (2024). Self-Correcting Self-Consuming Loops for Generative Model Training. arXiv:2402.07087. doi:10.48550/arXiv.2402.07087. [PREPRINT]
- Fu, S., Zhang, S., Wang, Y., Tian, X. & Tao, D. (2024). Towards Theoretical Understandings of Self-Consuming Generative Models. arXiv:2402.11778. doi:10.48550/arXiv.2402.11778. [PREPRINT]
- Cai, Z., Wang, Y., Liu, Y. & Zhang, X. (2025). Stabilizing Self-Consuming Diffusion Models with Latent Space Filtering. arXiv:2511.12742. doi:10.48550/arXiv.2511.12742. [PREPRINT]
- Wei, X. & Zhang, X. (2025). Self-Consuming Generative Models with Adversarially Curated Data. arXiv:2505.09768. doi:10.48550/arXiv.2505.09768. [PREPRINT]
- Zelikman, E., Wu, Y., Mu, J. & Goodman, N. D. (2022). STaR: Bootstrapping Reasoning With Reasoning. arXiv:2203.14465. doi:10.48550/arXiv.2203.14465. [PREPRINT]
- Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Garcia, X. et al. (2023). Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. arXiv:2312.06585. doi:10.48550/arXiv.2312.06585. [PREPRINT]
- Chen, Z., Deng, Y., Yuan, H., Ji, K. & Gu, Q. (2024). Self-Play Fine-Tuning Converts Weak Language Models to Strong Language Models. arXiv:2401.01335. doi:10.48550/arXiv.2401.01335. [PREPRINT]
- Yuan, W., Pang, R. Y., Cho, K., Li, X., Sukhbaatar, S., Xu, J. et al. (2024). Self-Rewarding Language Models. arXiv:2401.10020. doi:10.48550/arXiv.2401.10020. [PREPRINT]
- Zhao, A., Wu, Y., Yue, Y., Wu, T., Xu, Q., Yue, Y. et al. (2025). Absolute Zero: Reinforced Self-play Reasoning with Zero Data. arXiv:2505.03335. doi:10.48550/arXiv.2505.03335. [PREPRINT]
- Shao, R., Li, S. S., Xin, R., Geng, S., Wang, Y., Oh, S. et al. (2025). Spurious Rewards: Rethinking Training Signals in RLVR. arXiv:2506.10947. doi:10.48550/arXiv.2506.10947. [PREPRINT]
- Yan, L., Li, R., Chen, G., Li, Q., Geng, J. & Li, W. (2026). Spurious Rewards Paradox: Mechanistically Understanding

How RLVR Activates Memorization Shortcuts in LLMs. arXiv:2601.11061. doi:10.48550/arXiv.2601.11061. [PREPRINT]

Chen, P., Li, X., Li, Z., Yin, W., Chen, X. & Lin, T. (2025). Exploration vs Exploitation: Rethinking RLVR through Clipping, Entropy, and Spurious Reward. arXiv:2512.16912. doi:10.48550/arXiv.2512.16912. [PREPRINT]

Qin, C., Yang, C., Si, Q., Gu, N., Fu, P. & Lin, Z. (2026). Learning to Solve, Forgetting to Retain: Correct-Set Turnover in RLVR. arXiv:2606.03087. doi:10.48550/arXiv.2606.03087. [PREPRINT]

Liu, Y., Chen, J., Zhang, Q., Jin, T., Zhu, M. & Xi, M. (2026). Taming the Implicit: Dual-Channel Risk-Aware Reinforcement Fine-Tuning for Continual Multimodal Post-Training. arXiv:2608.03660. doi:10.48550/arXiv.2608.03660. [PREPRINT]

Li, L., Zhou, Z., Hao, J., Liu, J. K., Miao, Y. & Pang, W. (2025). The Choice of Divergence: A Neglected Key to Mitigating Diversity Collapse in Reinforcement Learning with Verifiable Reward. arXiv:2509.07430. doi:10.48550/arXiv.2509.07430. [PREPRINT]

Zeng, Z., Liu, J., Yin, Z., Zhang, G., Huang, W. & Qiu, X. (2025). RLoop: A Self-Improving Framework for Reinforcement Learning with Iterative Policy Initialization. arXiv:2511.04285. doi:10.48550/arXiv.2511.04285. [PREPRINT]

Yuan, C., Xiao, C., Rong, Y., Cheng, H. & Huang, L.-K. (2026). Decouple before Integration: Test-time Synthesis of SFT and RLVR Task Vectors. arXiv:2605.00610. doi:10.48550/arXiv.2605.00610. [PREPRINT]

Liu, Y., Chen, L., Liu, J., Zhu, M., Zhong, Z. & Yu, B. (2025). ViSurf: Visual Supervised-and-Reinforcement Fine-Tuning for Large Vision-and-Language Models. arXiv:2510.10606. doi:10.48550/arXiv.2510.10606. [PREPRINT]

Wu, C., Li, Z., Bu, J., Liu, J. & Wang, H. (2026). Sparsity Curse: Understanding RLVR Model Parameter Space from Model Merging. arXiv:2606.18521. doi:10.48550/arXiv.2606.18521. [PREPRINT]

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A. et al. (2016). Overcoming catastrophic forgetting in neural networks. arXiv:1612.00796. doi:10.48550/arXiv.1612.00796. [PREPRINT]

Kirk, R., Mediratta, I., Nalmpantis, C., Luketina, J., Hambro, E., Grefenstette, E. et al. (2023). Understanding the Effects of RLHF on LLM Generalisation and Diversity. arXiv:2310.06452. doi:10.48550/arXiv.2310.06452. [PREPRINT]

Cui, G., Zhang, Y., Chen, J., Yuan, L., Wang, Z., Zuo, Y. et al. (2025). The Entropy Mechanism of Reinforcement Learning for Reasoning Language Models. arXiv:2505.22617. doi:10.48550/arXiv.2505.22617. [PREPRINT]

Skalse, J., Howe, N. H. R., Krashennikov, D. & Krueger, D. (2022). Defining and Characterizing Reward Hacking. arXiv:2209.13085. doi:10.48550/arXiv.2209.13085. [PREPRINT]

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Yue, Y. et al. (2025). Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? arXiv:2504.13837. doi:10.48550/arXiv.2504.13837. [PREPRINT]

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Wang, P. et al. (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948. doi:10.48550/arXiv.2501.12948. [PREPRINT]

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X. et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300. doi:10.48550/arXiv.2402.03300. [PREPRINT]

Gao, L., Schulman, J. & Hilton, J. (2022). Scaling Laws for Reward Model Overoptimization. arXiv:2210.10760. doi:10.48550/arXiv.2210.10760. [PREPRINT]

Rafailov, R., Chittpu, Y., Park, R., Sikchi, H., Hejna, J. & Knox, B. et al. (2024). Scaling Laws for Reward Model Overoptimization in Direct Alignment Algorithms. arXiv:2406.02900. doi:10.48550/arXiv.2406.02900. [PREPRINT]

Tao, Z., Lin, T.-E., Chen, X., Li, H., Wu, Y. & Li, Y. (2024). A Survey on Self-Evolution of Large Language Models. arXiv:2404.14387. doi:10.48550/arXiv.2404.14387. [PREPRINT]

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X. & Leonardis, A. (2019). A continual learning survey: Defying forgetting in classification tasks. arXiv:1909.08383. doi:10.48550/arXiv.1909.08383. [PREPRINT]

Wataoka, K., Takahashi, T. & Ri, R. (2024). Self-Preference Bias in LLM-as-a-Judge. arXiv:2410.21819. doi:10.48550/arXiv.2410.21819. [PREPRINT]

Mobahi, H., Farajtabar, M. & Bartlett, P. L. (2020). Self-Distillation Amplifies Regularization in Hilbert Space. arXiv:2002.05715. doi:10.48550/arXiv.2002.05715. [PREPRINT]

Pang, R. Y., Yuan, W., Cho, K., He, H., Sukhbaatar, S. & Weston, J. (2024). Iterative Reasoning Preference Optimization. arXiv:2404.19733. doi:10.48550/arXiv.2404.19733. [PREPRINT]

(c) 2026 Pranay Mahendrakar. Licensed under CC BY 4.0.

The literature search, drafting and citation verification for this paper were carried out with AI assistance under the author's direction. Every citation was machine-verified against its live arXiv Atom API record during drafting (title and author list checked against the record returned), and every quantitative claim in this paper is taken from the abstract or a directly quoted headline result of the source credited with it. No experiment was run and no number in this paper was measured or recomputed by its author; Table 1 re-presents numbers published by the cited papers, each named on its row, and Figure 1 plots four of those numbers directly with no transformation beyond unit labeling. Algorithm 1 is original conceptual synthesis by the author, not an empirical result and not reproduced from any single cited source; it is presented as such.