RL's Razor: Why On-Policy Reinforcement Learning Forgets Less

Anonymous Author(s)
Affiliation
Address
email

Abstract

We compare fine-tuning models with supervised fine-tuning (SFT) and reinforcement learning (RL) and find that, even at matched new-task accuracy, RL consistently forgets less. We investigate the cause and show that the degree of forgetting is not determined by the training algorithm itself, but by the distributional shift, namely the KL divergence between the fine-tuned and base policy when evaluated on the new task distribution. RL's advantage arises because on-policy updates bias optimization toward KL-minimal solutions among the many that solve a task, whereas SFT can converge to distributions arbitrarily far from the base model. We validate this across experiments with large language models and controlled toy settings, as well as provide theory on why on-policy RL updates lead to a smaller KL change. We term this principle *RL's Razor*: among all ways to solve a new task, RL prefers those closest in KL to the original model.

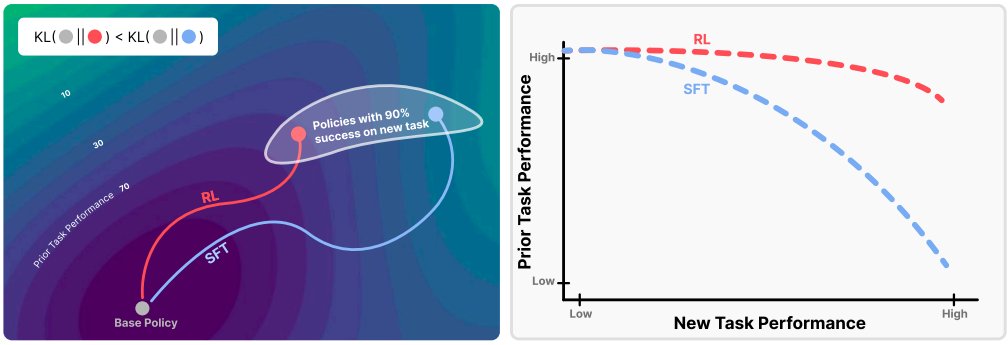

Figure 1: **RL prefers KL-minimal solutions.** Left: RL converges to policies close in KL to the base model. Right: This reduces forgetting at matched new-task accuracy compared to SFT.

1 Introduction

Foundation models have rapidly become the backbone of modern AI, powering applications in language, vision, robotics, and beyond. Despite their remarkable capabilities, today's models are largely *static* once deployed: they excel at tasks learned during pre-training or post-training, but are not designed to self-improve and continually acquire new capabilities. We imagine a future where deployed models are long-lived *agents* assisting humans in the long-term and continuously adapting to new needs. As such, models must improve and adapt to new data, environments, and objectives [1, 2, 3, 4, 5, 6].

A central challenge to this vision is *catastrophic forgetting*—the tendency for models to lose previously acquired capabilities when trained on new tasks [7, 8, 9, 10]. Although scaling model size and pre-training data improves robustness [11, 10, 12], catastrophic forgetting remains a persistent obstacle, undermining the promise of continual improvement [13, 14, 6]. To enable foundation models to serve as long-term agents, we need to develop post-training methods that allow models to acquire new skills without erasing old ones.

To further this goal, we analyze the performance of two widely used post-training schemes of supervised fine-tuning (SFT) and reinforcement learning (RL). Our experiments reveal a surprising finding: even when SFT and RL achieve the same performance on the new task, we observe that **SFT often achieves new-task gains by erasing prior knowledge, while RL better preserves old skills**. Figure 1 (right) illustrates this tradeoff: although both methods can reach high performance on the new task, RL maintains substantially higher performance on prior tasks compared to SFT.

This striking empirical gap raises the question: what underlying mechanism allows RL to improve on new tasks, but unlike SFT, not disturb the model's prior knowledge?

Previous approaches to catastrophic forgetting targeted specific factors such as constraining weight updates [9, 15], preserving learned features [16, 17], or regularizing shift in output distribution [18, 19]. While these methods can reduce forgetting, they focus on its effects rather than its underlying cause. Consequently, it remains unclear what truly governs forgetting or why different training algorithms behave so differently. In search of this governing principle, we ablated many such variables and uncovered an empirical forgetting law: **When fine-tuning a model, π, on a new task τ, the degree of forgetting is accurately predicted by $\mathbb{E}_{x\sim\tau}\left[\mathbf{KL}(\pi_0||\pi)\right]$**, the Kullback–Leibler (KL) divergence between the finetuned and the base policy computed on the new task distribution τ. Although the underlying reason for this phenomenon remains unclear, its consistency across settings suggests it captures a fundamental property of forgetting.

This law also clarifies the surprising difference between SFT and RL. Our analysis reveals a simple but powerful principle we call ***RL's Razor*: among the many high-reward solutions for a new task, on-policy methods such as RL are inherently biased toward solutions that remain closer to the original policy in KL divergence**. Figure 1 (left) highlights this effect: among the many policies that reach a high success rate on the new task, RL is biased toward KL-minimal solutions, while SFT can converge to distant ones. This bias arises directly from RL's *on-policy training*: by sampling from the model's own distribution at every step, RL constrains learning to outputs already given non-negligible probability by the base model. To improve reward, these samples are reweighted and used to update the model, which gradually shifts the policy rather than pulling it toward an arbitrary distribution. Thus, when multiple equally good solutions exist for a new task, RL tends to find solutions close to the original policy, while SFT can converge to solutions much farther away, depending on the provided labels. Theoretical analysis in a simplified setting confirms this view, showing that policy gradient methods converge to KL-minimal solutions even without explicit regularization.

Finally, to validate the KL hypothesis, we construct an "oracle SFT" distribution that provably minimizes KL divergence while achieving perfect accuracy. Training on this oracle distribution produces even less forgetting than RL itself. This demonstrates that RL's advantage does not stem from being inherently different, but from its implicit KL minimization. Whenever training is biased toward KL-minimal solutions, forgetting is reduced.

Our main contributions are:

- We show that RL fine-tuning forgets less than SFT, even when both reach the same performance on new tasks.
- We identify KL divergence to the base policy, measured on the new task distribution, as a strong predictor of catastrophic forgetting across objectives and hyperparameters.
- We provide empirical and theoretical evidence that the on-policy nature of policy gradient methods leads to smaller KL shifts and explains RL's advantage.

Together, these findings suggest a new perspective on post-training: to achieve continual adaptation without forgetting, algorithms should explicitly aim to minimize KL divergence from the base model. This principle opens the door to designing future training methods that combine RL's ability to preserve prior knowledge with the efficiency of SFT, enabling foundation models that can truly *learn for life*.

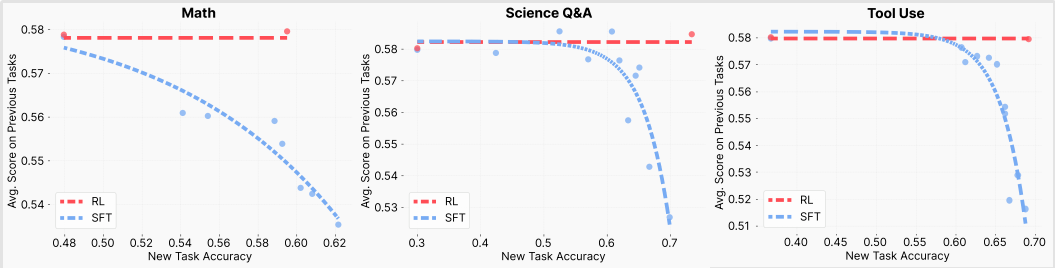

Figure 2: **Pareto frontiers of RL and SFT.** Each point represents a fine-tuned model. We sweep hyperparameters and plot only those on the Pareto-frontier. RL improves new-task performance while preserving prior knowledge, whereas SFT increases performance at the expense of forgetting.

2 Related work

Foundation Models and Post-training In modern deep learning, large-scale models trained on broad, diverse datasets (usually termed Foundation models) serve as general-purpose backbones across domains such as language, vision, robotics, and multimodal reasoning [20, 21, 22, 23, 24]. Pre-training often relies on self-supervised or weakly supervised objectives, producing models with broad domain knowledge and some zero-shot capabilities [25, 26]. However, raw pre-trained models may not directly meet the requirements of specific applications or align with domain-specific constraints. Post-training methods address this gap by adapting foundation models to downstream tasks through supervised fine-tuning on curated datasets [27, 28, 29, 30], reinforcement learning from human or automated feedback [31, 32, 33, 34], and other techniques [35].

Catastrophic Forgetting. While fine-tuning primarily aims to improve performance on a new specific task, preserving the model's pre-existing general capabilities is equally critical. Unfortunately, fine-tuning often leads to catastrophic forgetting—a phenomenon where learning new information significantly deteriorates previously acquired knowledge [7, 8, 9]. Many works have sought to reduce forgetting by constraining updates, for example through weight penalties, feature preservation, or output matching [36]. These approaches highlight important factors, but what remains missing is a unifying principle that predicts forgetting across different algorithms and settings.

In contrast, our work does not propose a new method but instead identifies a simple law: the KL divergence between the fine-tuned and base policy, measured on the new task distribution, reliably predicts the degree of forgetting. This explains the success of some popular forgetting mitigation methods like Elastic Weight Update [9], which can be seen as approximation of KL minimization [37]. Interestingly, practitioners have also observed that KL regularization, originally introduced in RL fine-tuning of LLMs to stabilize optimization or prevent reward hacking [19, 38], helps reduce catastrophic forgetting [32].

SFT versus RL. Most prior comparisons between SFT and RL have focused on performance on the new task being learned. A seminal result by [39] showed that in sequential decision making, on-policy learning can achieve stronger performance even when the learning signal is identical. Building on this, recent studies have found that RL fine-tuned models often exhibit superior generalization beyond the training distribution [40, 41, 42] and transfer more effectively to related tasks [43] compared to SFT trained models. However, none of this work has examined their relative susceptibility to catastrophic forgetting, which is the focus of our study.

Concurrently, [44] report that RL forgets less than SFT, but ascribe RL's advantage to negative examples and ignore sampling-distribution effects. Section 5 shows that this assumption is inconsistent with our results.

3 Reinforcement Learning Forgets Less than SFT

In this section, we compare the degree of catastrophic forgetting induced by SFT and RL. The comparison is carried out by training large language models on new tasks and then measuring how much their ability to perform previously acquired tasks is degraded.

3.1 Performance Trade-offs

Experimental Setup. For each new task, we trained two models on the same set of prompts. The first model was trained with SFT, using annotations either from the original dataset or generated with GPT-4o. The second model was trained with RL, specifically using GRPO [45]. The base model in all cases was Qwen 2.5 3B-Instruct [46]. Evaluation was performed along two axes:

- New tasks: We measured performance on the held-out test set of the newly introduced task to assess the performance gain from the training.
- Previous tasks: We measured performance on a diverse set of unrelated benchmarks. A drop in these benchmarks was taken as a measure of catastrophic forgetting.

To obtain a reliable comparison, we trained dozens of models for each method under a variety of hyperparameter choices. Importantly, all RL experiments were done *without explicit KL regularization*. We then plotted only the models lying on the Pareto frontier, i.e., points such that no model can improve its performance on the new task without incurring greater degradation on previous tasks. The resulting lines show the best trade-off between learning and forgetting achievable by each training method. For more details, see Appendix B.

Tasks and Datasets. We repeated this experiment across three distinct domains:

- *Math reasoning*: math questions from the Open-Reasoner-Zero dataset [47], annotated with GPT-4o [48] responses filtered for correctness.
- *Science Q&A*: Chemistry L-3 subset of SciKnowEval [49], also annotated with GPT-4o.
- *Tool use*: ToolAlpaca dataset [50], using available annotations.

For the evaluation of catastrophic forgetting, we used established benchmarks: Hellaswag [51], TruthfulQA [52], MMLU [53], IFEval [54], Winogrande [55], and HumanEval [56]. These serve as proxies for diverse prior abilities that the model should ideally retain.

Results. Figure 2 illustrates the learning-forgetting trade-offs for all tasks. Across all of them, RL training produces nearly horizontal Pareto frontiers as gains on the new task are achieved without loss on previous tasks. In contrast, SFT exhibits a steep downward slope—as new task accuracy rises, average performance on prior tasks consistently degrades. The contrast is particularly sharp in *Math*, where even slight improvement in performance results in drops in prior ability. In *Science Q&A* and *Tool Use*, SFT is able to achieve some performance without forgetting, but extracting all possible gains from the dataset results in severe performance degradation.

Takeaway 1

While SFT boosts new-task performance by sacrificing prior knowledge, RL achieves comparable improvements with substantially less forgetting.

3.2 Representation Preservation

While benchmark performance provides an external view of forgetting, it can also be sensitive to superficial factors, such as formatting mismatches with the previous tasks. To probe whether the training altered the model's internal capacity more fundamentally, we analyzed changes to the model's representations.

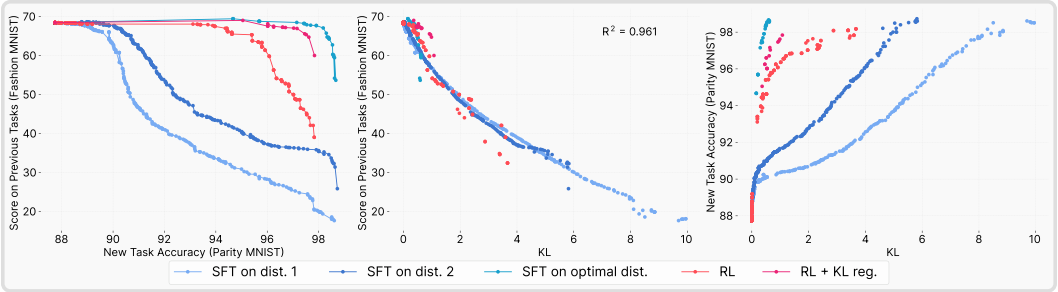

Figure 4: **KL divergence predicts catastrophic forgetting.** (Left) Learning-Forgetting Trade-offs. SFT outperform RL only when an oracle distribution is used as a source of annotation. (Middle) Forgetting aligns to a single curve when plotted against KL divergence, showing KL as a strong predictor across methods. (Right) RL improves new-task accuracy with much smaller KL shifts than SFT, highlighting the conservativeness of on-policy updates.

Experimental Setup. To study how representations change between models, we compare their embeddings on a shared dataset. Directly comparing raw embedding values is not meaningful, since these can vary arbitrarily during training. Instead, it is common to compare the relative geometry of the embeddings—that is, how different inputs relate to each other. This geometry can be summarized by a kernel (similarity) matrix, which encodes pairwise relationships among inputs. Centered Kernel Alignment (CKA) [57] is a standard measure for comparing such kernels, providing a way to quantify representational similarity between models.

For this analysis, we constructed kernels from random Wikipedia paragraphs, ensuring that the comparison probes representations of content unrelated to the new training tasks. We then measured the similarity of the kernel between the base model and its fine-tuned variant using a version of CKA called CKNNA ([58], see Appendix B.3 for more details). Specifically, we compare models that achieved similar final accuracy on the new task.

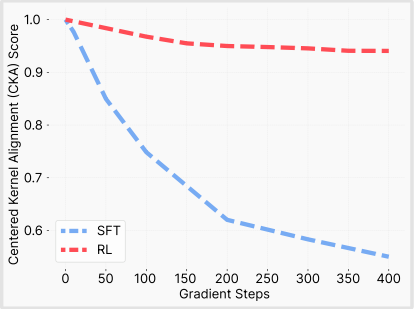

Figure 3: **CKA similarity to the base model during training.** Although SFT and RL achieve comparable task performance, SFT models diverge substantially in their representations, whereas RL models remain more closely aligned with the base model.

Results. Figure 3.2 shows that RL-trained models maintain almost perfect representational alignment with the base model, even after learning the new task. By contrast, SFT-trained models diverge substantially, indicating that the training reshapes their internal space in ways that distort previously encoded knowledge. Together with the benchmark results, this suggests that RL is able to integrate new abilities without disturbing the broader conceptual structure, while SFT incurs representational shifts that manifest as catastrophic forgetting.

4 Smaller KL divergences lead to less forgetting

A central question raised by our initial results is why RL fine-tuning tends to forget less than SFT. To address this, we sought a confounding variable that could robustly explain the behavior of both methods. We systematically tested several candidates, including weight change under different norms, sparsity, and gradient rank. None of these explained the observed differences (see Section 6). What ultimately emerged is that the *KL divergence between the trained model and the base model on the new task distribution* is an excellent predictor of catastrophic forgetting.

To test this hypothesis in a setting where we could run large numbers of experiments and push RL until it starts to forget, we developed a toy problem we call ParityMNIST. The task is derived from MNIST [59], but instead of digit classification, the label is defined by parity (even vs. odd). An image of an even digit is correctly classified if the model predicts *any* label of an even number. Therefore, multiple different labelings are equally correct: any distribution that consistently maps images of

even digits to the labels {0,2,4,6,8} and odd digits to the others achieves optimal accuracy. This mirrors the realistic setting where many different output distributions are equally valid.

We pretrained a 3-layer MLP jointly on a small subset of ParityMNIST and FashionMNIST [60], then fine-tuned only on ParityMNIST while measuring forgetting on FashionMNIST. To parallel the main setup:

- In the **SFT** setting, the model was trained on labels sampled from a single arbitrary distribution out of the many possible correct ones.
- In the **RL** setting, the reward was correctness with respect to parity, leaving the model free to converge to any valid distribution.

For more details, see Appendix B.2. This design allowed us to replicate the phenomenon where RL reached high accuracy on the new task with substantially slower degradation of prior knowledge, while SFT exhibited a steeper trade-off (Figure 4, left). Importantly, *reproducing the effect in this simple MLP setting shows that it is not specific to transformers or language modeling, but a more general property of fine-tuning deep generative models*.

KL as Predictor. When we plot forgetting directly against KL divergence from the base model on the ParityMNIST distribution, results across both RL and SFT collapse onto the same curve (Figure 4, middle). This shows that KL, not the training algorithm itself, predicts forgetting. A second-order polynomial fit achieves $R^2 = 0.961$ in this toy setting. We repeated the experiment with two different arbitrary SFT label distributions. Although their Pareto frontiers differed, plotting forgetting against KL again yielded the same curve, reinforcing the conclusion that KL governs forgetting irrespective of the training method or label choice. The same correlation is observed in our LLM experiments with $R^2 = 0.61$. Although the statistic is lower, the residuals are mean-zero and not explained by predictors, consistent with random noise possibly due to approximated measures of KL and task accuracy. In addition, we find that at the optimization level, update steps that change the model more in terms of KL also yield gradients that are more directionally similar to the catastrophic forgetting gradient, see Figure 8.

Optimal SFT Distribution. Finally, the simplicity of ParityMNIST allowed us to analytically compute the optimal SFT distribution, that is, the distribution with minimal KL divergence to the base model among all that achieve 100% accuracy, see B.2 for details. Training SFT directly on this distribution outperformed RL, showing that the key factor is the KL distance from the base model, with RL being naturally biased toward such solutions. In addition, we also trained the SFT model on data generated by a RL trained model to and find that it can reach the same level of performance and forgetting as the RL model, see Figure 7.

Takeaway 2

Across both SFT and RL, the amount of catastrophic forgetting is determined by how far the model moves from its base distribution on the new task, as measured by KL divergence.

5 On-policy methods leads to smaller KL divergence

Having established that the KL divergence between the trained model and its base distribution on the new task predicts catastrophic forgetting, we now ask: why are RL fine-tuned models able to achieve strong task performance while moving less in KL than SFT models?

5.1 Experimental Evidence

To answer this, we examine the differences in training objectives. For discrete output spaces, SFT minimizes the cross-entropy loss:

$$\mathcal{L}_{\text{SFT}}(\pi) = -\mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\beta}[\log \pi(y|x)]$$

where annotations are sampled from an external distribution π_β, usually provided by humans. For RL, we consider the standard policy gradient estimator:

$$\mathcal{L}_{\text{RL}}(\pi) = -\mathbb{E}_{x\sim\mathcal{D}, y\sim\pi}\left[A(x,y)\log \pi(y|x)\right]$$

Figure 5: **Comparison of algorithm classes.** (Left) The four quadrants illustrate algorithm types, defined by whether they are on- or off-policy and whether they incorporate negative gradients. (Middle) On-policy methods retain prior knowledge more effectively. (Right) Both GRPO and 1-0 Reinforce achieve higher new-task accuracy while incurring smaller KL shifts from the base model, showing that on-policy methods consistently induce more conservative KL updates.

where $A(x, y)$ is an Advantage function, which is the reward of y normalize in respects to other rewards to the same x. In practice, the expectation is taken over trajectories (or outputs) sampled from the current policy, and the policy gradient trick [61] ensures gradients are taken only through the log-probability term, not through the sampling distribution inside the expectation. Two features distinguish this from SFT:

1. Sampling Distribution. While in RL the training was done on outputs drawn from the model's own distribution, in SFT, they come from fixed external annotations.
2. Negative Examples. While sampling from π, some of the responses will not be ideal. These are usually assigned a negative coefficient $A(x, y)$. This pushes probability mass away from poor outputs, a mechanism absent in SFT.

Our hypothesis is that one of these two differences is what causes RL's slower forgetting. To examine our hypothesis, we perform experiments with four different objectives:

- *GRPO*. An on-policy objective that utilizes negative examples. Here, $A(x, y)$ is the normalized reward.
- *1–0 Reinforce*. An on-policy algorithm that does not use negative examples. Here, $A(x, y) = 1$ for correct responses and 0 for incorrect ones. This is equivalent to sampling from the model and performing SFT on correct answers only.
- *SFT*. An offline objective that does not use negative examples.
- *SimPO*. An offline objective that utilizes negative examples. We create negative examples by sampling incorrect responses from an external model, and use the SFT data for positive examples. The SimPO [62] loss compares correct and incorrect outputs via a logistic term:

$$\mathcal{L}_{\text{SIMPO}}(\pi) = -\mathbb{E}_{x\sim\mathcal{D}, y_w\sim\pi_{\beta+}, y_l\sim\pi_{\beta-}} \left[\log\sigma\left(\log\pi(y_w|x) - \log\pi(y_l|x) - 1\right)\right] \quad (1)$$

  where $\pi_{\beta+}$ and $\pi_{\beta-}$ denote distributions for correct and incorrect responses, respectively. We used SimPO rather than naïve likelihood/negative likelihood because the latter was unstable to train.

We compared the four objectives on the Science Q&A task, measuring their learning–forgetting trade-offs as in Section 4. The results, shown in Figure 5, reveal that 1–0 Reinforce behaves similarly to GRPO, while SimPO resembles SFT. Thus, the critical factor is not the presence of negative gradients but the use of on-policy data. Plotting KL divergence confirms this conclusion: on-policy methods (GRPO and 1–0 Reinforce) reach the same task performance with significantly smaller KL divergence from the base model than offline methods (SFT and SimPO).

5.2 Theoretical Perspective

Policy gradient methods can be understood as a form of conservative projection. At each step, the policy samples outputs it already finds likely, then re-weights those samples according to reward, shifting probability mass toward higher-reward outcomes while suppressing lower-reward ones. Crucially, because updates are defined relative to the model's own distribution, they nudge the policy toward a nearby re-weighted distribution, rather than pulling it toward a potentially distant external distribution (as in SFT). This explains why policy gradient methods tend to remain close to the base model in KL divergence.

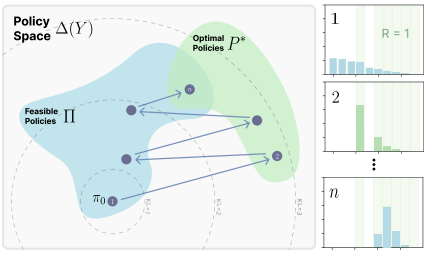

Figure 6: **KL-minimal path to optimality.** Alternating I-projection into the set of optimal policies and M-projection into Π carries π_0 into P^* while preferring the closest solution in KL.

This perspective can be formalized by observing that, in the binary-reward case, the re-weighted distribution targeted by policy gradient is exactly the minimum-KL projection of the current policy onto the set of optimal ones.

Lemma 5.1. *Let p be a distribution over a finite set Y, and let $R : Y \to \{0, 1\}$ be a reward function. Rejection sampling from p with acceptance condition $R(y) = 1$ yields a distribution q_{RS}. This distribution can be equivalently characterized as the solution to:*

$$q_{RS} =_q D_{KL}(q||p) \quad s.t \quad \mathbb{E}_{y\sim q}[R(y)] = 1$$

Building on this, we show that policy gradient converges to the KL-minimal optimal policy within the representable family.

Theorem 5.2. *Let Y be a finite set and let $\Pi \subseteq \Delta(Y)$ be a convex family of feasible policies (e.g., an exponential family). Let $R : Y \to \{0, 1\}$ be a binary reward function and $P^* = \{q : \mathbb{E}_q[R] = 1\}$ the set of optimal policies. Then, under suitable regularity conditions, solving the reinforcement learning objective with policy gradient converges to*

$$\pi^\dagger = \arg \min_{\pi \in P^* \cap \Pi} D_{\mathrm{KL}}(\pi \,\|\, \pi_0),$$

where π_0 is the initialization. In other words, policy gradient selects, among all optimal representable policies, the one closest in KL-divergence to the starting policy.

A detailed version with proofs is provided in Appendix A.

Takeaway 3

On-policy training explains why RL maintains smaller KL divergence than SFT. Sampling from the model's own distribution keeps it close to the base model, while SFT pushes it toward arbitrary external distributions.

6 Incorrect Hypothesis

Science advances not only through identifying the right explanations, but also by systematically ruling out incorrect ones. To this end, we tested a broad set of candidate variables as potential predictors of catastrophic forgetting. These variables fell into four categories:

- **Weight-level changes.** A natural hypothesis is that forgetting is tied to how much the parameters themselves move. We measured parameter changes under L_1, Fisher-weighted L_2, and spectral norm metrics. The Fisher matrix was computed on the basis of the model parameters, with expectation over inputs from the previous task. These metrics correlated only weakly with forgetting: large parameter shifts could occur without forgetting, and conversely, forgetting sometimes occurred despite small parameter movement.
- **Sparsity and rank of updates.** Motivated by [63], who argue that RL updates are sparse while SFT weight updates are dense, we explicitly tested this hypothesis. In our setting, however, we found that the reason for the observed sparse updates where the use of `bfloat16` for model training. Since `bfloat16` has a limited mantissa, small parameter updates (such as those produced by RL) can fail to cross the representational threshold, effectively causing no update at all. Performing the

same training with `float32` resulted in models with identical performance but without any sparsity in their weight updates. Checking the rank of the weight changes, we found that all algorithms lead to full rank weight updates.

- **Representation-level changes.** Following the CKNNAA results from subsection 3.2, we examined hidden activation shifts (L1 and L2 distances) as proxies for changes in internal representations. These variables show some correlation, but the curves were distinct between training objectives.
- **Distributional distances.** We considered multiple measures of output distribution change, all measured over inputs from the new task τ: Forward KL ($\mathbb{E}_{x\sim\tau}\left[\mathrm{KL}(\pi_0||\pi)\right]$), Reverse KL ($\mathbb{E}_{x\sim\tau}\left[\mathrm{KL}(\pi||\pi_0)\right]$), Total Variation, and L_2 distance between distributions. While reverse KL showed a good signal, and TV moderately correlated with forgetting, none approached the predictive power of forward KL.

Table 1 summarizes these results. Across all candidates, forward KL divergence between the fine-tuned and base model on the new task distribution emerges as the only consistent and high-fidelity predictor of catastrophic forgetting.

Variable	R^2 (2nd deg. polynomial)
KL, forward	$\mathbf{0.96 \pm 0.01}$
KL, reverse	0.93 ± 0.01
TV	0.80 ± 0.01
Distribution change, L2	0.56 ± 0.02
Weight change, L1	0.34 ± 0.02
Weight change, Fisher Weighted L2	0.58 ± 0.02
Weight change, spectral norm	0.58 ± 0.02
Sparsity of weight change	N/A
Rank of weight change	N/A
Activation change, L1	0.52 ± 0.02
Activation change, L2	0.55 ± 0.02

Table 1: Predictive power of alternative variables compared to forward KL. None approaches the explanatory strength of forward KL divergence.

7 Discussion and Conclusion

Our study reveals that catastrophic forgetting is governed not by the choice of training algorithm, but by the KL divergence from the base policy on the new task distribution. This explains why RL forgets less than SFT, as on-policy training naturally biases updates toward KL-minimal solutions, preserving prior knowledge while acquiring new skills.

However, we still lack a mechanistic account of why larger KL shifts on the new task disrupt prior knowledge—whether through representational interference, implicit capacity limits, or other dynamics. Moreover, while we demonstrate the KL–forgetting link across moderate-scale LLMs and toy models, its behavior at frontier scales and in more diverse generative domains remains unknown. Addressing these gaps will be essential for grounding the principle and extending it to real-world deployment.

Taken together, our results motivate a new design axis for post-training research: algorithms should be judged not only by how well they optimize new tasks, but also by how conservatively they move in KL relative to the base model. Embracing this principle may allow us to build agents that not only learn new skills, but also truly learn for life.

References

[1] Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

[2] Alan Dao and Thinh Le. Rezero: Enhancing llm search ability by trying one-more-time. *arXiv preprint arXiv:2504.11001*, 2025.

[3] Mohammad Mahdi Moradi, Hossam Amer, Sudhir Mudur, Weiwei Zhang, Yang Liu, and Walid Ahmed. Continuous self-improvement of large language models by test-time training with verifier-driven sample selection. *ArXiv*, abs/2505.19475, 2025.

[4] Zhongyang Li, Ziyue Li, and Tianyi Zhou. C3po: Critical-layer, core-expert, collaborative pathway optimization for test-time expert re-mixing. *ArXiv*, abs/2504.07964, 2025.

[5] Toby Simonds and Akira Yoshiyama. Ladder: Self-improving llms through recursive problem decomposition. *arXiv preprint arXiv:2503.00735*, 2025.

[6] Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. *ArXiv*, abs/2506.10943, 2025.

[7] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[8] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

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

[10] 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.

[11] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International conference on learning representations*, 2021.

[12] Andrea Cossu, Antonio Carta, Lucia Passaro, Vincenzo Lomonaco, Tinne Tuytelaars, and Davide Bacciu. Continual pre-training mitigates forgetting in language and vision. *Neural Networks*, 179:106492, 2024.

[13] Rishi Bommasani. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[14] Haiyang Guo, Fanhu Zeng, Fei Zhu, Jiayi Wang, Xukai Wang, Jingang Zhou, Hongbo Zhao, Wenzhuo Liu, Shijie Ma, Da-Han Wang, et al. A comprehensive survey on continual learning in generative models. *arXiv preprint arXiv:2506.13045*, 2025.

[15] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[16] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1320–1328, 2017.

[17] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839, 2019.

[18] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

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

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[21] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

[23] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Hao-Shu Fang, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023.

[24] Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, Jianfeng Gao, et al. Multimodal foundation models: From specialists to general-purpose assistants. *Foundations and Trends® in Computer Graphics and Vision*, 16(1-2):1–214, 2024.

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

[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[27] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[28] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

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

[30] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

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

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

[33] 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.

[34] Simon Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Peter Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971, 2024.

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

[36] 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*, 46(8):5362–5383, 2024.

[37] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018.

[38] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

[39] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[40] Seungwook Han, Jyothish Pari, Samuel J Gershman, and Pulkit Agrawal. General reasoning requires learning to reason from the get-go. *arXiv preprint arXiv:2502.19402*, 2025.

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

[42] Tianle Li, Jihai Zhang, Yongming Rao, and Yu Cheng. Unveiling the compositional ability gap in vision-language reasoning model. *arXiv preprint arXiv:2505.19406*, 2025.

[43] Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. *arXiv preprint arXiv:2507.00432*, 2025.

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

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

[46] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[47] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

[48] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[49] Kehua Feng, Keyan Ding, Weijie Wang, Xiang Zhuang, Zeyuan Wang, Ming Qin, Yu Zhao, Jianhua Yao, Qiang Zhang, and Huajun Chen. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*, 2024.

[50] Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

[51] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[52] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

[53] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[54] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[55] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[56] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[57] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR, 2019.

[58] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

[59] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[60] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[61] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[62] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

[63] Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. *arXiv preprint arXiv:2505.11711*, 2025.

[64] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[65] Imre Csiszár. Information geometry and alternating minimization procedures. *Statistics and Decisions, Dedewicz*, 1:205–237, 1984.

[66] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2000.

[67] CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

[68] Asela Gunawardana, William Byrne, and Michael I Jordan. Convergence theorems for generalized alternating minimization procedures. *Journal of machine learning research*, 6(12), 2005.

[69] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

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

A Theory

Lemma A.1 (Rejection sampling as an I-projection)**.** *Let p be a distribution over a finite set Y, and let $R : Y \to \{0, 1\}$ be a reward function. Rejection sampling from p with acceptance condition $R(y) = 1$ yields a distribution q_{RS}. This distribution can be equivalently characterized as the solution to:*

$$q_{RS} =_q D_{KL}(q||p) \quad s.t \quad \mathbb{E}_{y\sim q}[R(y)] = 1$$

Equivalently, q_{RS} is the I-projection of p onto the set $\{q : \mathbb{E}_q[R] = 1\}$

Proof. Let $S = \{y \in Y : R(y) = 1\}$. Rejection sampling produces the conditional distribution

$$q_{\mathrm{RS}}(y) = \begin{cases} \frac{p(y)}{p(S)} & y \in S, \\ 0 & y \notin S, \end{cases}$$

where $p(S) = \sum_{y\in S} p(y)$ and we assume $P(S) > 0$.

Now consider the optimization problem. The constraint $\mathbb{E}_q[R] = 1$ means

$$\sum_{y\in Y} q(y)R(y) = \sum_{y\in S} q(y) = 1$$

so q must put all of its mass on S. Thus the feasible set is exactly all distributions supported on S.

For any q supported on S, we can write $p(y) = p(S)\, p(y|S)$ for $y \in S$, and then

$$\begin{aligned} D_{\mathrm{KL}}(q\|p) &= \sum_{y\in S} q(y) \log \frac{q(y)}{p(y)} = \sum_{y\in S} q(y) \log \frac{q(y)}{p(y \mid S)} - \log p(S) \sum_{y\in S} q(y) \\ &= D_{\mathrm{KL}}\big(q\|p(\cdot \mid S)\big) - \log p(S) \end{aligned}$$

where we used $\sum_{y\in S} q(y) = 1$ in the last step. The second term is constant in q, so minimizing $D_{\mathrm{KL}}(q||p)$ is the same as minimizing $D_{\mathrm{KL}}(q||p(\cdot|S))$. By strict convexity of $D_{\mathrm{KL}}(\cdot\|\cdot)$ in its first argument, the unique minimizer is $q = p(\cdot \mid S) = q_{\mathrm{RS}}$. □

Lemma A.2 (Policy gradient as an M-projection)**.** *Let Y be a finite set and let $\Pi \subseteq \Delta(Y)$ be a set of admissible policies (distributions over Y). Consider the single-step reinforcement learning objective*

$$\max_{\pi} \mathbb{E}_{y\sim\pi}[R(y)]$$

where $R : Y \to \mathbb{R}_{\geq 0}$ is a reward function. By the policy gradient theorem, this objective is equivalently optimized by

$$\max_{\pi} \mathbb{E}_{y\sim\bar{\pi}}\big[R(y) \log \pi(y)\big]$$

where $\bar{\pi}$ indicates that gradients are not propagated through the sampling distribution. Define the distribution

$$q(y) = \frac{\pi(y)R(y)}{Z}, \qquad Z = \sum_{y\in Y} \pi(y)R(y)$$

Then taking a policy gradient step is equivalent to taking a gradient step on the following objective:

$$\min_{\pi} -\mathbb{E}_{y\sim q}[\log \pi(y)]$$

In other words, optimizing the RL objective using policy gradient is equivalent to finding the M-projection of q onto the set of feasible policies π using gradient descent.

Proof. Expanding the policy gradient objective gives

$$\mathbb{E}_{y\sim\bar{\pi}}[R(y)\log\pi(y)] = \sum_{y\in Y}\pi(y)R(y)\log\pi(y)$$

Let $Z = \sum_{y\in Y}\pi(y)R(y)$. Define $q(y) = \pi(y)R(y)/Z$. Then the above becomes

$$\sum_{y\in Y}\pi(y)R(y)\log\pi(y) = Z\sum_{y\in Y}q(y)\log\pi(y) = Z\,\mathbb{E}_{y\sim q}[\log\pi(y)]$$

Since Z does not depend on π in the gradient computation (it is treated as a constant in the $\bar{\pi}$ sense), maximizing the original objective is equivalent to maximizing $\mathbb{E}_{y\sim q}[\log\pi(y)]$.

Finally, recall that the M-projection of a distribution q onto a set of distributions Π is given by

$$\min_{\pi\in\Pi}\mathrm{KL}(q\|\pi) = \mathbb{E}_q[\log\frac{q}{\pi}] \;=\mathbb{E}_q[\log q] \;-\; \mathbb{E}_q[\log\pi]$$

since $\mathbb{E}_q[\log q]$ does not depend on π, the maximizer of $\mathbb{E}_{\bar{\pi}}[R\log\pi]$ over Π coincides with $\arg\min_{\pi\in\Pi}\mathrm{KL}(q\|\pi)$. Thus, the policy gradient update corresponds to the M-projection of q onto the policy class. □

Theorem A.3 (RL with binary reward as an EM algorithm)**.** *Let Y be a finite set and let $\Pi\subseteq\Delta(Y)$ be a set of feasible policies. Let $R: Y\to\{0,1\}$ be a binary reward function and P^* the set of all optimal policies $P^* = \{q : \mathbb{E}_q[R] = 1\}$. Then, solving the Single-step reinforcement learning objective using policy gradients is equivalent to performing the following optimization procedure:*

$$q_t = \arg\min_{q\in P^*}\mathrm{KL}(q\|\pi_t), \qquad \pi_{t+1} = \arg\min_{\pi\in\Pi}\mathrm{KL}(q_t\|\pi)$$

This procedure is also known as EM with information projection.

Proof. Sampling $y\sim\pi$ and accepting iff $R(y) = 1$ is exactly rejection sampling onto the event $S = \{y\in Y : R(y) = 1\}$. The resulting distribution is $\pi(\cdot|S)$. By Lemma A.1 with $p\leftarrow\pi$, this $\pi(\cdot|S)$ solves

$$\min_q D_{\mathrm{KL}}(q\|\pi) \quad \text{s.t.} \quad \mathbb{E}_q[R] = 1$$

establishing the I-projection. Applying Lemma A.2 on the RL objective gives us the M-projection. □

Proposition A.4 (Convergence to minimum KL solution)**.** *Under the setting appear in theorem A.3 and assume Π is an e-flat (exponential-family) model with full support, the optimal set P^* is nonempty and realizable (i.e., $\Pi\cap P^*\neq\varnothing$). Then:*

(1) *If the M-projection is exact at every step, then (π_t) converges to*

$$\pi^\dagger = \arg\min_{\pi\in P^*\cap\Pi} D_{\mathrm{KL}}(\pi\,\|\,\pi_0)$$

(2) *If the M-projection is inexact but, for some errors $\varepsilon_t\geq 0$, it holds that*

$$D_{\mathrm{KL}}(q_t\|\pi_{t+1}) \;\leq\; \min_{\pi\in\Pi} D_{\mathrm{KL}}(q_t\|\pi) \;+\; \varepsilon_t \quad \textit{with} \quad \sum_{t=0}^{\infty}\varepsilon_t < \infty$$

then π_t also converges to the same limit $\pi^\dagger$.

Proof. The I-step is always an exact I-projection (Lemma A.1). In the case of an exact M-step, the iterative process is EM with information projections. The e-/m-flat geometry yields the Pythagorean identities implying convergence to $\pi^\dagger$ [64, 65, 66]. When the M-step only ensures a (near-)minimization up to summable errors, the iteration is GEM: monotone improvement and convergence follow from the GEM theory of [67] together with generalized alternating minimization for Bregman divergences [68], which, under the same e-/m-flat assumptions, selects the same minimum-KL limit $\pi^\dagger$. □

Practical considerations. Our theoretical equivalence should be interpreted with the following caveats:

- Beyond REINFORCE. In practice, many policy gradient algorithms such as GRPO and PPO replace the raw reward $R(y)$ with an advantage estimate $A(y)$. Since this substitution is a control variate technique, it leaves the expected gradient direction unchanged while reducing its variance. Thus, our projection-based interpretation continues to hold.
- The optimal policy set P^* defined by the linear constraint $\mathbb{E}_q[R] = 1$ is an m-flat family, but the representable policy set Π induced by a neural network parametrization is not in general e-flat. This may prevent exact convergence to the minimum-KL solution described above. Nevertheless, our theorem provides a principled explanation for the bias observed in practical RL algorithms.

B Training and Evaluation Details

B.1 LLM experiments

Unless otherwise stated, all reinforcement learning experiments were conducted using GRPO [45].

For the *Math* reasoning task, the training set provided final answers but lacked reasoning chains required for SFT training. To obtain these, we queried DeepSeek R1 [33], sampling up to 16 responses per prompt and retaining a single response that matched the correct final answer. This yielded valid annotations for 96% of the dataset. For the *Science Q&A* task, we applied the same procedure with GPT-4o, obtaining correct annotations for the entire dataset.

To construct the learning–forgetting trade-off curves (e.g., Figure 2), we followed the protocol below:

1. Hyperparameter sweep. We trained multiple models under a broad sweep of hyperparameters (see Table 2).
2. New-task evaluation. For *Math* and *Science Q&A*, accuracy was measured by comparing the model's final answer to the ground truth, ignoring intermediate reasoning chains. For Tool Use, we extracted API calls from the output and matched them against ground-truth calls via regular expressions.
3. Previous-task evaluation. We assessed performance on unrelated benchmarks as described in Section 3.1, using the `Language Model Evaluation Harness` [69].
4. Pareto filtering. From the trained models, we retained only those lying within 2 accuracy points of the Pareto frontier.
5. Curve fitting. An exponential function was fit to the filtered points to produce the trade-off curves.

B.2 MNIST Experiments

All MNIST experiments were conducted using a 3-layer MLP with input dimension 785, hidden layers of sizes 512 and 256, and output dimension 10. The input consisted of a flattened 28×28 image concatenated with a binary indicator: $+1$ for ParityMNIST and -1 for FashionMNIST.

Pretraining. We pretrained the network jointly on ParityMNIST and FashionMNIST using small subsets of the original datasets (500 images from each). For ParityMNIST, the label was chosen uniformly at random among all digit labels with the correct parity.

Fine-tuning methods. We evaluated five fine-tuning strategies:

1. **GRPO**.
2. **GRPO + KL regularization** with coefficient 0.1.
3. **SFT 1**: all even digits mapped to label 0, all odd digits to label 1.
4. **SFT 2**: even digits randomly mapped to $\{0, 4\}$, odd digits to $\{1, 5\}$.
5. **SFT with oracle distribution**: annotations drawn from the minimum-KL distribution consistent with task correctness.

Hyperparameter	SFT / SIMPO	RL
Base Model	Qwen2.5 3B-Instruct	Qwen2.5 3B-Instruct
Learning Rate	{1e-5, 3e-5, 5e-5, 7e-5, 9e-5}	{1e-5, 2e-5, 3e-5, 4e-5, 5e-5}
Optimizer	adamw	adamw
LR Scheduler	{constant w. warmup, cosine w. warmup}	constant w. warmup
Warmup steps	50	50
Epochs	{1,2}	1
Batch Size	{16,32,64,128}	See Below
Max Grad Norm	1	1
bfloat16	True	True
Weight Decay	0	0
GRPO-only hyperparameters		
KL reg.		0
Group Size		64
Prompts per generation		8
num iterations (μ)		{1,2}
Loss type		Dr. GRPO [70]

Table 2: Hyperparameters used for the LLM experiments. Curly braces {} indicate a sweep over the specified values. Additional parameters such as weight decay and max gradient norm were manually ablated; since they showed no significant effect on results, they were not included in the final sweep.]

Oracle distribution. Motivated by the KL–forgetting connection, we define the oracle distribution as the one that achieves perfect task accuracy while remaining closest (in KL divergence) to the pretraining distribution π_0. Concretely, for an input image x we compute $\pi_0(\cdot|x) \in \mathbb{R}^{10}$ and the binary indicator vector $R \in \{0,1\}^{10}$ encoding which labels are correct given the digit's parity. The oracle distribution q^* is the solution to:

$$q^* = \arg\min_q D_{\mathrm{KL}}(\pi_0 \| q) \quad \text{s.t.} \quad q^\top R = 1.$$

Since KL is convex and the constraint is linear, we can calculate a closed-form solution for every image. We then sample from q^* to produce SFT annotations.

Hyperparameter sweep. For each method we trained models across a sweep of 15 learning rates logarithmically spaced between $3e-6$ and $1e-3$, using either a constant-with-warmup or cosine-with-warmup scheduler, and training for 1 or 2 epochs. Including mid-training checkpoints, this produced approximately 500 runs per method.

B.3 Centered Kernel Alignmen

Centered Kernel Alignment (CKA) [57] Given representations $X, Y \in \mathbb{R}^{n\times d}$, define kernels $K = XX^\top$, $L = YY^\top$. Let $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ be the centering matrix. The centered kernels are

$$\bar{K} = HKH, \quad \bar{L} = HLH.$$

CKA is then computed as

$$\mathrm{CKA}(K, L) \;=\; \frac{\langle \bar{K}, \bar{L}\rangle_F}{\|\bar{K}\|_F \, \|\bar{L}\|_F},$$

where $\langle A, B\rangle_F = \mathrm{tr}(A^\top B)$.

CKA with k-NN Alignment (CKNNA) [58] Let $\alpha(i,j) \in \{0,1\}$ indicate whether i, j are mutual k-nearest neighbors in both X and Y. Define the masked inner product

$$\langle A, B\rangle_\alpha = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha(i,j)\, A_{ij} B_{ij}.$$

CKNNA is then given by

$$\mathrm{CKNNA}(K, L) \;=\; \frac{\langle \bar{K}, \bar{L}\rangle_\alpha}{\sqrt{\langle \bar{K}, \bar{K}\rangle_\alpha \, \langle \bar{L}, \bar{L}\rangle_\alpha}}.$$

When $\alpha(i, j) = 1$ for all $i \neq j$, CKNNA reduces to standard CKA.

C Additional Results

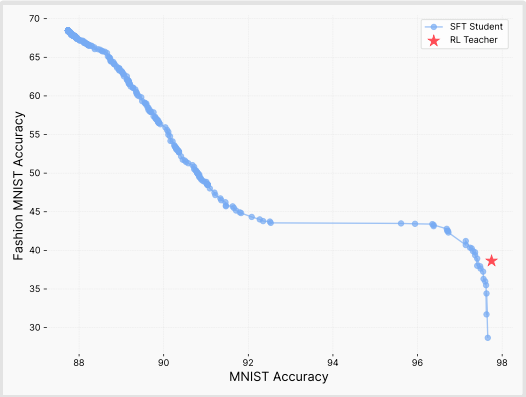

Figure 7: **SFT distillation from an RL teacher.** Accuracy trade-off between the new task (MNIST) and the prior task (FashionMNIST). Sweeping student hyperparameters shows that SFT can match the teacher within noise on both tasks. This suggests that what matters is not the optimization path, but the distribution of the final model.

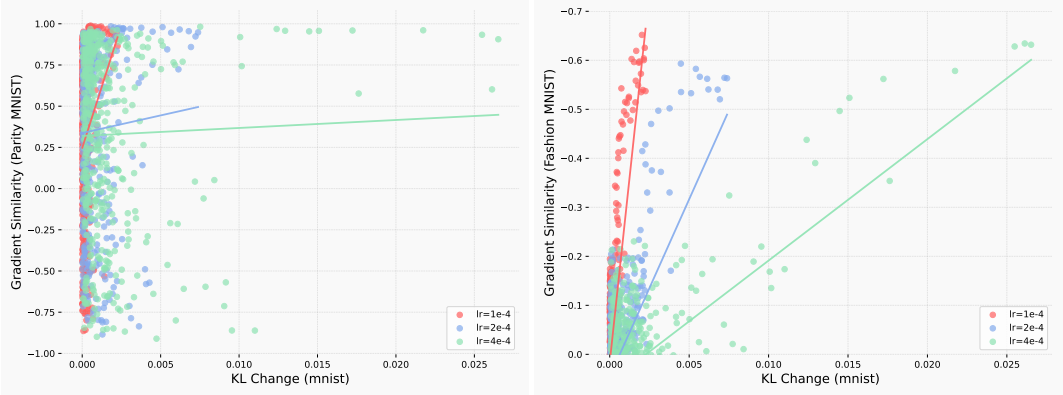

Figure 8: **Gradient similarity versus KL change.** (Left) On the new training task (ParityMNIST), gradient cosine similarity and KL change per step remain anti-correlated. (Right) On the prior task (FashionMNIST), the gradient similarity is more correlated with the KL change per step on the training task (ParityMNIST). Together, these plots show that taking a larger step on the current task induces gradients that are more similar in direction to the