PROJECTED DISTILLATION: ON LEARNING TO SOLVE HARD PROBLEMS FROM CLOSE, CORRECT TARGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy learning is central to post-training: training on a model's own generations limits distribution shift and helps preserve existing capabilities. On hard problems, however, on-policy RL provides no reward-based learning signal when all attempts fail. On-policy distillation provides next-token supervision from a teacher, but on hard problems, if a rollout goes wrong early, subsequent supervision is conditioned on prefixes that already contain the mistake. Correct behavior can instead come from a gold or teacher solution, but such off-policy targets can differ substantially from model's generations and, as we show, may erode existing capabilities. We therefore treat on-policyness as a continuum and propose projected distillation, which minimally edits the student's failed attempts into correct targets. The method uses contrastive updates to favor corrections over failed attempts and refreshes these corrections as the student evolves, enabling near-on-policy learning even when all student attempts fail. Theoretically, we prove that the student's learning regret grows with the edit size: it scales linearly with the horizon under minimal correction, as in DAgger, and quadratically under full-answer replacement, as in behavior cloning. Empirically, across two students (`Qwen3-8B` and `OLMo-2-7B`) and five domains spanning math, code, science, and tool use, projected distillation gives the highest area under the specialization–retention frontier (best on 8 of 10 model–task pairs). Our ablations highlight the importance of online corrections, contrastive loss, and, on-policyness: moving targets farther off-policy steadily worsens the specialization–retention trade-off, with retention falling from 95% to 75% when minimal corrections are replaced by student-agnostic gold solutions.

1 INTRODUCTION

Post-training adapts general-purpose language models to reason, follow instructions, and use tools. The dominant approaches are reinforcement learning from verifiable rewards (RLVR; Shao et al., 2024; Schulman et al., 2017), distillation from a stronger teacher (Agarwal et al., 2023), and supervised fine-tuning (Wei et al., 2022). Across these approaches, the training signal is most effective when it remains on-policy – that is, close to the model's own output distribution. Updates near this distribution can strengthen the target behavior while preserving capabilities acquired during pretraining (Agarwal et al., 2023; Yang et al., 2024). Post-training is particularly important for models with only a few billion parameters: although they are inexpensive to run and easy to deploy, they often lack the proficiency required for specialized tasks. We study how to teach such models to solve hard problems that they currently answer only rarely, if at all, without degrading their existing capabilities.

A fundamental challenge of on-policy RL is that on very hard problems the on-policy signal disappears. RLVR requires the model to succeed occasionally, so when every sampled attempt fails the reward is zero everywhere and nothing is reinforced. On-policy distillation (OPD; Agarwal et al., 2023) keeps the update on the model's own rollouts and corrects them token by token to be closer to a response of a stronger teacher. However, the student rollout of a difficult task contains many mistakes: feedback on the later tokens is computed on top of an already-wrong prefix, so it largely patches earlier errors instead of teaching the step at hand (Zhang et al., 2026a). Additionally, the later-token feedback in OPD is inefficient: as the student learns, the prefix on which later-token feedback was given and training budget was spent becomes less probable.

Alternatively, the training signal can come from a complete correct answer supplied from outside, either ground-truth data or a stronger teacher's solution. Unlike OPD, which corrects the student's

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

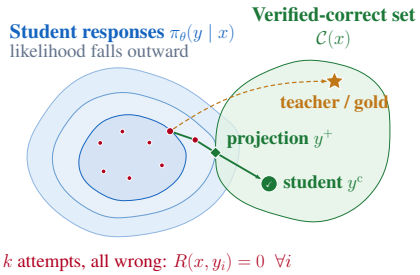


Figure 1: **Learning from the nearest correct answer.** Correct answers form a set $\mathcal{C}(x)$, but the student assigns them very different likelihoods. On a hard problem, its k on-policy attempts (●) are all wrong and earn no reward. A teacher or gold answer (★) supplies correctness, but may lie far from what the student writes. Projected distillation instead minimally edits a failed attempt into a newly computed projection (◆) and refreshes that target as the student changes. The green path illustrates repeated online rounds: successive wrong rollouts may move closer until the student produces a correct answer itself (✓). The resulting small edits keep supervision near-policy; writing m for edit size and H for the generation horizon, our analysis bounds their drift by a term proportional to mH.

own rollout token by token, this target replaces the rollout entirely: the whole training budget goes on producing a correct answer rather than on patching the one the student wrote. Such a target is drawn from a distribution other than the student's own, so it is off-policy. The further it lies from what the student would produce, the more the student specializes on the task while eroding its abilities elsewhere. This suggests keeping the target a complete correct answer but choosing the one closest to what the student already writes, so the supervision stays near-on-policy.

Several recent works pursue similar ideas. SPOT (Lin & Han, 2026), SODA (Chen et al., 2026), and SuperCorrect (Yang et al., 2025) fix the student's mistakes with a strong black-box teacher and learn from the corrections. What they leave unexamined is the on-policyness of the correction itself, how it changes with the size of the edit, and whether refreshing the corrections online against the evolving student matters. We focus on these aspects. We call the recipe of minimal, online corrections *projected distillation* (Fig. 1) and study it along five axes: how well it learns from close, correct solutions relative to other post-training approaches; why the edit must be minimal to keep the correction near-on-policy; why the corrections must be refreshed against the still-training student; which objective learns best from a correction; and how the projector itself can be improved. A regret analysis in the imitation-learning setting shows the importance of the edit size.

Our contributions. (1) We introduce *projected distillation*, a post-training recipe that learns from minimal, online corrections of the student's mistakes. It needs only the correction text, so any capable black-box model can serve as the projector. Across two students and five benchmarks it attains the best specialization–retention trade-off, and retains prior abilities where standard post-training forgets on continual-learning metrics (Sec. 4.1). **(2)** Holding correctness fixed, we show that the correction must be both minimal and online. Pushing it off-policy monotonically worsens the trade-off (Sec. 4.2), and freezing the pool instead of refreshing it makes it collapse as the student drifts (Sec. 4.3). We further give an efficient training objective, a diff-localized contrast with self-training on correct rollouts (Sec. 4.4), and a refinement procedure that improves the projections from a mid-capacity (32B) teacher without updating its parameters (Sec. 4.5). **(3)** We theoretically analyze projected distillation through imitation learning. The edit size interpolates between DAgger's linear-in-horizon regret and behavior cloning's quadratic regret (Sec. 5).

2 Related Work

Methods for improving a model on hard problems. Approaches differ in where the correct target comes from and how far that target lies from the model's own distribution; together these largely determine the specialization–retention trade-off. Reinforcement learning from verifiable rewards (RLVR; GRPO (Shao et al., 2024), PPO (Schulman et al., 2017)) draws the target from the model's own rollouts, but on the pass@k=0 set every rollout is wrong and no signal remains. On-policy distillation (GKD (Agarwal et al., 2023)) and self-distillation (Yang et al., 2024) keep the update on those rollouts while matching a teacher's token distribution, which requires teacher logits. Supervised fine-tuning on an external gold or teacher solution supplies correctness directly but is maximally off-policy, so fitting it forgets. Between these, there is scope to manufacture a correct target based on the student's attempt. This opens up the possibility of treating the correct solutions along a continuum from on-policy to off-policy with respect to the model's current generation style. Our work studies how far the edits may go, whether they need to be recomputed as the model changes, and what objective to use for effective learning.

Closest work. The literature presents approaches similar to ours: LeMa (An et al., 2023) mines corrections once, from an assortment of models rather than the student, with free-form rewrites and an objective that uses response likelihood maximization. SPOT (Lin & Han, 2026) gathers mistakes of the base model, rectifies the mistakes using minimal edits from an oracle, creates a preference dataset and learns with a reward-based binary cross-entropy objective. SODA (Chen et al., 2026) creates mistake–correction pairs without emphasizing minimal edits, performs an SFT warm-up, and then, with the SFT'ed policy fixed, learns from static preference pairs. SuperCorrect (Yang et al., 2025) distills thought templates from a teacher and applies collaborative DPO over the teacher's correction traces. We also work in the hard-task regime where the model fails. However, our paper focuses mainly on the regime of online learning from close, correct targets to teach the student how to effectively solve hard tasks. We perform a controlled study of the edit sizes used to provide corrections (Sec. 4.2), static vs. online-refreshed pools of corrections (Sec. 4.3) and effective learning objectives for black-box learning (Sec. 4.4). In addition, we theoretically analyze how the edit-size influences the learning regret, which distinguishes our work from the previous ones. Sec. B gives a more detailed discussion of related work, covering distillation approaches, preference objectives used in black-box learning, imitation learning and continual learning.

3 Projected Distillation

We study post-training on problems the model rarely solves, where its own samples earn no reward and correctness has to come from outside.

Setup. Given a *problem* x, a solution is a token sequence $y \in \mathcal{V}^{\leq H}$ drawn from a vocabulary $\mathcal{V}$ with a generation horizon H. The *student* produces y autoregressively via $\pi_{\theta_S}(y \mid x) = \prod_t \pi_{\theta_S}(y_t \mid x, y_{<t})$, where y_t denotes the t-th token, $y_{<t}$ its prefix, and π_{ref} the frozen initialization of the student. Correctness is judged by a *verifier* $R(x, y) \in \{0, 1\}$ (e.g., exact-match checking or unit tests), inducing the set of accepted solutions $\mathcal{C}(x) = \{y : R(x, y) = 1\}$. Our focus is on problems that lie beyond the student's current ability, i.e., pass@k=0 set, for which all k sampled attempts fail. Further, in Sec. 5, we formalize the generation process using a dynamic-token sequence MDP to provide theoretical insights.

Projection. The accepted set $\mathcal{C}(x)$ typically contains many solutions, but these solutions can differ substantially in likelihood under the student. Training toward a solution the student already considers likely requires less movement of its output distribution and therefore better preserves unrelated capabilities (Kirkpatrick et al., 2017; Kumar et al., 2022). We therefore choose, among the verifier-accepted solutions in $\mathcal{C}(x)$, the one with highest likelihood under the student.

Definition 1 (Projection)**.** For a student π_{θ_S} and a problem x, the *projection* is the correct answer the student is most likely to produce, $\text{proj}(x) \in \arg\max_{y' \in \mathcal{C}(x)} \log \pi_{\theta_S}(y' \mid x)$.

Searching $\mathcal{C}(x)$ is intractable, so we reach that element through the attempt itself. A wrong attempt is a sample from the student and is therefore already likely under it; a correction that preserves most of its tokens preserves most of that likelihood, while the verifier places the result in $\mathcal{C}(x)$. Minimal editing is thus a tractable surrogate for Definition 1, and Sec. 4.2 tests it, varying the edit size and measuring the likelihood shift $\Lambda = \log \pi_{\theta_S}(y^+ \mid x) - \log \pi_{\theta_S}(y^- \mid x)$ that each size demands. A *projector* receives the problem and the attempt in context and is prompted to make the smallest change that corrects the answer (examples in Sec. C; prompts in Sec. D.3).

Learning from projection. In each round we prompt the student on the hard problems and collect a batch of responses. Most of them are wrong. We write y^- for a wrong response and y^+ for its projection, and y^c for the responses the student solves on its own. We have access only to the text the projector returns, not to its logits, so we cannot match its token distribution. We learn from the pair (y^-, y^+) by preference instead. The projection edits only a small part of the attempt, so y^- and y^+ agree almost everywhere, and a contrast over their full length would spend most of its gradient on text that was already correct. We therefore restrict the contrast to the tokens the edit changed. A token-diff of y^- against y^+ marks a *mistake span* E^- and a *fix span* E^+; for instance, for $y^- = \texttt{ABCD}$ and $y^+ = \texttt{AXCZ}$, $E^- = \{\texttt{B}, \texttt{D}\}$ and $E^+ = \{\texttt{X}, \texttt{Z}\}$. We apply a DPO contrast (Rafailov et al., 2023) to those spans alone, using the frozen initialization π_{ref} as reference and β for its strength:

$$\mathcal{L}_{\text{Diff-Local-DPO}} = -\,\mathbb{E}_x \log \sigma\Big(\beta\big[\textstyle\sum_{t \in E^+} \log \frac{\pi_{\theta_S}(y_t^+ \mid x, y_{<t}^+)}{\pi_{\text{ref}}(y_t^+ \mid x, y_{<t}^+)} - \sum_{t \in E^-} \log \frac{\pi_{\theta_S}(y_t^- \mid x, y_{<t}^-)}{\pi_{\text{ref}}(y_t^- \mid x, y_{<t}^-)}\big]\Big), \quad (1)$$

Here σ is the logistic function. The contrast constrains only how likely y^+ is relative to y^-, so we add a term that supervises the student's own correct responses directly:

$$\mathcal{L}_{\text{Projected-Distillation}} = \mathcal{L}_{\text{Diff-Local-DPO}} + \mathcal{L}_{\text{SFT}}, \text{ where } \mathcal{L}_{\text{SFT}} = -\mathbb{E}_x \sum_t \log \pi_{\theta_S}(y^c_t \mid x, y^c_{<t}). \quad (2)$$

Therefore, the wrong responses supply the contrast and the correct ones the supervised term. The next round re-prompts the student on the same problems and projects whatever it gets wrong then, so the targets follow the current policy. Fig. 2 summarizes one round of our approach. Sec. 4.3 compares rebuilding the targets each round against reusing a static correction pool, and Sec. 4.4 compares our loss (2) against five alternatives on the same data.

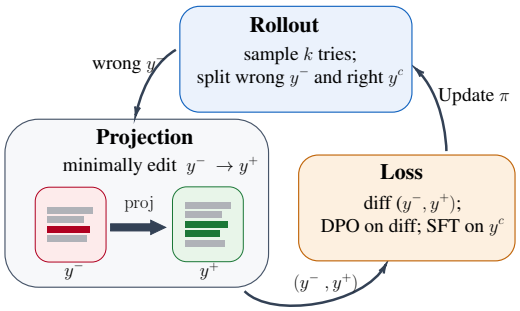


Figure 2: **One round of projected distillation.** The student answers each problem k times, and its responses split into the wrong ones y^- and the correct ones y^c. A projector rewrites a wrong response into a correct one y^+ with the smallest edit that works. The two are aligned token by token, and the preference term acts only on the spans that differ (removed, added), while y^c is supervised directly. The updated student begins the next round, so its corrections are recomputed as it changes.

4 Experiments

First, we compare projected distillation with standard post-training approaches in the hard-task regime (Sec. 4.1). The remaining sections examine the recipe itself, each component of which could have been set differently: the correction may change more or fewer of the attempt's tokens (Sec. 4.2), it may be computed once or recomputed as the student learns (Sec. 4.3), and several objectives may be applied to it (Sec. 4.4). We vary each in turn while holding the remainder fixed, and finally ask whether the projections can be improved further by test-time scaling (Sec. 4.5).

Experimental Setup: We study two instruction-tuned students from different families, **Qwen3-8B** (Qwen Team, 2025) and **OLMo-2-7B** (OLMo Team, 2024), on five benchmarks spanning math, code, tools and science: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), LiveCodeBench v6 (Jain et al., 2024), SciKnowEval (Feng et al., 2024), and ToolAlpaca (Tang et al., 2023). For each (student, benchmark) pair we train *only* on its hard set: the problems on which the base student fails all k=16 sampled attempts at temperature 0.8. The hard set is mined from the benchmark's training split (Sec. D.2), and retention is measured in-domain on a disjoint held-out slice of the same benchmark. All methods use matched budgets for optimization updates and per-step generation. The remaining configuration is in Sec. D.1, with prompt templates in Sec. D.3 and further training details in Sec. D.4. Unless noted, results are averaged over 3 training seeds, and Sec. D.5 separates training-seed from evaluation-seed spread. Sec. E.3 additionally evaluates the trained checkpoints on suites they never saw.

Baselines. We compare against four baselines, each probing a different axis of the post-training design space. **GRPO** (Shao et al., 2024) performs on-policy RL with verifiable rewards, where positive rewards are scarce. **OPD** (Agarwal et al., 2023) performs white-box distillation from the token-level logits of a strong teacher conditioned on the student's own prefixes, using a same-family 32B teacher (Qwen3-32B or OLMo-2-32B) for each student; our method uses these same 32B teachers as its projection models, and both methods are given the gold answer as privileged information. **Surgical Post-Training** (SPOT; Lin & Han, 2026) minimally corrects the student's initial attempts and trains on the resulting *frozen* preference data, while **Semi-On-policy Distillation** (SODA; Chen et al., 2026) fine-tunes the student on teacher corrections and then contrasts a teacher response against a frozen copy of that fine-tuned student. Against SPOT and SODA every method draws its corrections from one shared black-box model, which isolates the learning algorithm from the projector.

Metrics. We report **specialization**, pass@1 on the hard set, and **retention**, pass@1 on the held-out slice. A method can raise either at the other's expense, so neither effectively ranks methods on its own. Our headline metric is the **area under the specialization–retention Pareto frontier** (AUPF), the retention a method sustains as it specializes further, anchored at a shared untrained base. AUPF is low both for a method that never specializes and for one that specializes by giving up retention. We

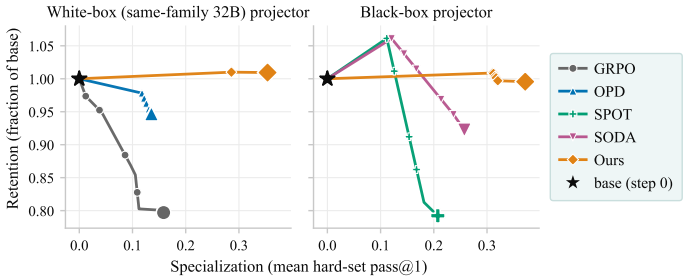


Figure 3: **Aggregate specialization–retention Pareto frontier**, one curve per method pooled over both students, all benchmarks and seeds, split by projector class. **Left:** white-box (same-family) projectors. **Right:** black-box projector. GRPO uses no projector. y-axis is retention measured as held-out accuracy as a fraction of the untrained base. x-axis is specialization measured as average hard-set accuracy. Each curve is the monotone Pareto frontier of the pooled trajectory anchored at $\star = (0, 1.0)$; *up-and-to-the-right is better*. **Every baseline ends below its untrained base**: GRPO and OPD decline from the outset, while SPOT and SODA exceed base early and then fall below it. Projected distillation holds retention at base across the range and attains the highest specialization.

Table 1: **RQ1: area under the specialization–retention frontier (AUPF).** Higher is better (Sec. E); mean $\pm$ s.e. over 3 seeds. GRPO uses no projector and is shown for reference. The OLMo-2-7B LiveCodeBench cell is omitted, since no method moves specialization appreciably above zero there. **Bold** = best per row within each projector block. Ours attains the highest AUPF on 8 of 9 cells with the same-family teacher and 7 of 9 with the black-box projector.

		RLVR	same-family 32B teacher		black-box projector		
		GRPO	OPD	Ours	SPOT	SODA	Ours
Qwen3-8B	GSM8K	0.061±.011	0.145±.006	**0.205±.012**	0.209±.030	0.231±.016	**0.262±.020**
	MATH	0.000±.000	0.003±.001	**0.008±.002**	0.145±.010	**0.159±.009**	0.035±.010
	LCB	0.000±.000	0.068±.018	**0.138±.003**	0.095±.005	0.091±.001	**0.123±.006**
	SciKnow	0.045±.008	0.027±.002	**0.179±.005**	0.124±.008	**0.155±.013**	0.124±.005
	Tool	0.248±.152	0.128±.004	**0.676±.013**	0.594±.010	0.635±.014	**0.667±.015**
OLMo-2-7B	GSM8K	0.195±.084	**0.417±.011**	0.217±.014	0.186±.007	0.210±.004	**0.336±.067**
	MATH	0.009±.003	0.014±.002	**0.017±.002**	0.013±.001	0.015±.002	**0.016±.001**
	SciKnow	0.198±.022	0.060±.013	**0.236±.010**	0.116±.011	0.101±.001	**0.234±.015**
	Tool	0.171±.111	0.319±.038	**0.626±.018**	0.222±.031	0.488±.029	**0.682±.038**

report it alongside three continual-learning summaries of the retention trajectory, backward transfer (Lopez-Paz & Ranzato, 2017), forgetting (Chaudhry et al., 2018) and training-averaged accuracy (Kemker et al., 2018), and retention-safe specialization, the highest specialization reached while retention stays within ε of base. Full definitions are in Sec. E.

4.1 RQ1: How does projected distillation compare with baselines?

We compare against the four baselines, drawing our corrections first from a same-family 32B teacher and then from a shared black-box projector, and report AUPF in Tab. 1, the continual-learning metrics in Tab. 2, and the pooled frontiers in Fig. 3.

Projected distillation attains the highest AUPF on eight of nine cells against GRPO and OPD and on seven of nine against SPOT and SODA (Tab. 1), under both projector settings. The margins are largest on SciKnowEval and ToolAlpaca, reaching 0.179 against OPD's 0.027 on Qwen SciKnowEval and 0.676 against 0.128 on Qwen ToolAlpaca. The baselines are not short of specialization; SPOT and SODA both reach real hard-set accuracy (Tab. 2). However, they retain poorly. In Fig. 3 our frontier runs flat along the untrained base across the whole specialization range while every baseline's falls away from it. Pooled across projectors, our mean AUPF is 0.256 with the mid-capacity same-family 32B teacher and 0.275 with the black-box projector, so the mid-capacity projector performs slightly worse. In Sec. 4.5, we investigate how to improve learning with such a teacher.

The two baseline families fall short for different reasons. GRPO has little to learn from, since the hard set is mined at pass@k=0; continued sampling still recovers occasional correct rollouts, enough for the highest specialization of any method on OLMo SciKnowEval, but its retention is the lowest there as well (Tab. 6), because reinforcing a rare correct rollout sharpens the student onto that solution at the expense of what it produces elsewhere (Yue et al., 2025). SPOT and SODA hold correct targets

Table 2: **RQ1: continual-learning metrics**, pooled over students, benchmarks and seeds (mean ± s.e.), with arrows marking the better direction. $\bar{x}$ and $\bar{y}_{\text{norm}}$ are specialization and retention averaged over training, BWT is backward transfer against the base, F the drop from a run's retention peak, and $\text{SafeSpec}_{\varepsilon}$ the most specialization reached at any checkpoint whose retention is within ε of base (Sec. E). **Bold** = best, underline = second best per column. Our variants specialize most while holding retention at base, where the baselines either barely specialize or forget.

Method	$\bar{x}\uparrow$	$\bar{y}_{\text{norm}}\uparrow$	BWT$\uparrow$	$F\downarrow$	$\text{SafeSpec}_{2\%}\uparrow$	$\text{SafeSpec}_{5\%}\uparrow$
GRPO	0.114±0.034	0.826±0.048	−0.203±0.058	0.221±0.057	0.035±0.012	0.056±0.022
OPD	0.118±0.022	0.951±0.031	−0.062±0.043	0.136±0.041	0.101±0.019	0.135±0.026
SPOT	0.184±0.031	0.706±0.074	−0.380±0.082	0.509±0.064	0.147±0.020	0.149±0.020
SODA	0.224±0.031	0.776±0.053	−0.266±0.059	0.355±0.044	0.140±0.021	0.190±0.022
Ours (32B)	0.264±0.042	**1.000**±0.020	**0.009**±0.028	**0.084**±0.015	**0.202**±0.040	**0.295**±0.048
Ours (bb)	**0.276**±0.042	0.995±0.024	−0.004±0.034	0.107±0.027	0.160±0.037	0.235±0.040

throughout, so they are not limited by correctness. They are rather limited by how far the correction moves the attempt and how often it is rebuilt, as shown further in sections 4.2 and 4.3. Three AUPF result cells are not in our favour. On Qwen MATH, SPOT and SODA fit the hard-set answers more closely, SODA reaching 0.136 specialization against our 0.031, and give up some retention for it (0.664 against our 0.760; Tab. 6). On Qwen SciKnowEval SODA leads on both axes, in the one cell whose retention carries a standard error of 0.131 across seeds, several times that of any other method there. On OLMo GSM8K, OPD reaches 0.496 specialization against our 0.236 at comparable retention, so token-level feedback apparently keeps its effectiveness there.

Training on a hard set risks catastrophic forgetting due to limited model capacity, so Tab. 2 reports four continual-learning measures, each summarizing a whole run. Training-averaged specialization and retention give the mean of each axis over the run: ours is highest on specialization ($\bar{x} = 0.264$ and 0.276 against SODA's 0.224) and holds retention at base ($\bar{y}_{\text{norm}} = 1.000$ and 0.995 against 0.776), so what it gains is not paid for out of retention during training. Backward transfer is the net change in retention from the untrained base to the final checkpoint, and ours are the only values that are not significantly negative (0.009 and −0.004). Forgetting is the drop from a run's highest retention to its final retention, so it registers ability lost during training even when the endpoint looks acceptable: ours are 0.084 and 0.107 against 0.509 for SPOT and 0.355 for SODA, which specialize early and lose retention through the rest of training (Fig. 10). Retention-safe specialization, the most a run reaches at any checkpoint whose retention is within ε of base, is highest for both our variants at either tolerance. Retention here is measured on a held-out slice of the trained benchmark; Sec. E.3 evaluates the math-trained checkpoints on suites they never saw.

4.2 RQ2: How does edit size influence learning in projected distillation?

A correction may be correct and still lie far from what the student would have produced, and this question concerns the cost of that distance. We construct four arms that hold correctness fixed and vary only how much the correction rewrites the attempt: a minimal edit, two progressively larger rewrites obtained by re-prompting the projector, and *gold*, the dataset's GSM8K reference solution, which discards the attempt altogether. Each arm trains a separate Qwen3-8B student from the same base under identical hyperparameters, so only the target y^+ varies; the rejected side remains the student's own wrong rollout y^-. The projector rather than the prompt settles how much text is replaced, so we identify the arms by the edit size they realize and measure the distance a correction imposes by the likelihood shift it demands, $\Lambda = \log \pi_\theta(y^+ \mid x) - \log \pi_\theta(y^- \mid x)$, reported as $|\Lambda|$.

Fig. 4 shows that among the three arms that differ only in edit size, relative specialization falls to 0.54 and then 0.45 of the minimal-edit arm and retention from 0.916 to 0.807, while $|\Lambda|$ rises from 86 to 433. The arms are equally correct, so what the larger rewrites cost is distance from the student. *Gold* sits outside that ordering: its edit is 926 tokens, between the two rewrites, yet it specializes better than either (0.87 of the minimal edit) and returns the lowest retention of any arm (0.744). The token count understates its distance, since gold keeps none of the attempt and counts small only because a reference solution is short; its likelihood shift says the same, $|\Lambda| = 3969$ against 433 at the largest rewrite, although that value is dominated by the length difference and is not on the rewrites' scale. We therefore read edit size as the quantity governing the trade-off among matched rewrites, with the minimal edit best balanced, and report *gold* on the behavioral axes only; Sec. F gives the per-arm values and the mechanism measurements.

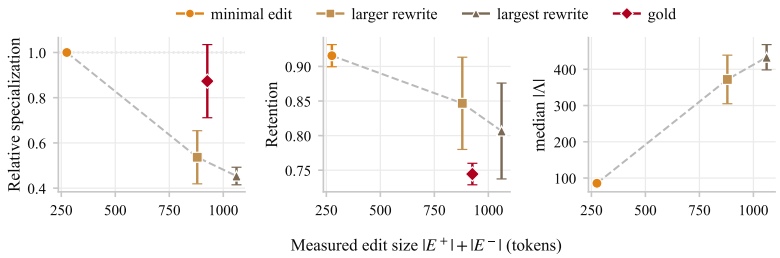


Figure 4: **RQ2: what a larger edit costs.** Qwen3-8B student per arm on the GSM8K hard set, varying only how much a verified-correct correction rewrites the student's own attempt (mean ± s.e. over 3 seeds). Arms sit at their *measured* edit size, since the projector rather than the prompt sets it, and the dashed line joins the three matched arms. **Left to right:** hard-set pass@1 relative to the minimal-edit arm, absolute held-out pass@1, and the median likelihood shift $|\Lambda|$. Both behavioural axes fall as the edit grows while $|\Lambda|$ rises. *Gold* appears on the first two panels only: it keeps none of the attempt, so its edit count understates how far it sits, and its $|\Lambda|$ of 3969 is off the scale.

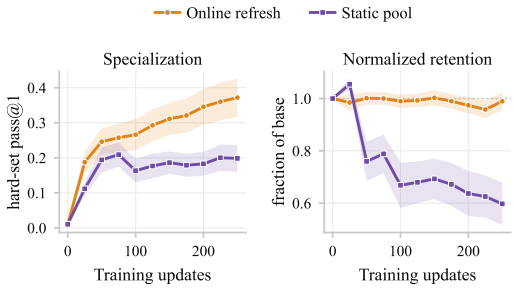


Figure 5: **RQ3: online refresh against a static pool**, pooled over all model×benchmark cells and 3 seeds. Both arms take minimal edits from the same black-box projector; the static pool mines them once against the base checkpoint and reuses them for the run, the regime of mined-once correction methods. Refreshing holds retention at base throughout, while the pool climbs briefly above base and then falls to 0.60.

4.3 RQ3: How much does refreshing corrections online help?

Projected distillation recomputes its corrections at every round, so that each target is an edit of an attempt the current student has just produced. Naturally, we ask what role this mechanism plays in projected distillation learning. For comparison, we gather a *static pool*: minimal-edit corrections of the initial checkpoint's responses, mined once before training and reused for the remainder of the run. This is the same regime occupied by methods like SPOT (Lin & Han, 2026) and LeMa (An et al., 2023). The projector for both static and online-refreshed arms is the same black-box projector. We report results pooled over all model×benchmark cells and three seeds in figure 5.

Refreshing the corrections is essential. Online refresh improves specialization steadily throughout training, reaching 0.37 pass@1, whereas the static pool rises over the first few updates and then flattens near 0.20 (Fig. 5, left). Retention separates more sharply: online refresh holds it at base for the whole run, while the static pool climbs slightly above base over the first 25 steps, falls to 0.76 by step 50, and declines from there to 0.60 (Fig. 5, right). The fall sets in once the pool has been reused for several rounds. A static pool cannot follow the student to the failures it develops afterwards, so repeated passes fit it to a fixed set of targets, while its generalization degrades.

4.4 RQ4: How best to learn from projections of wrong responses?

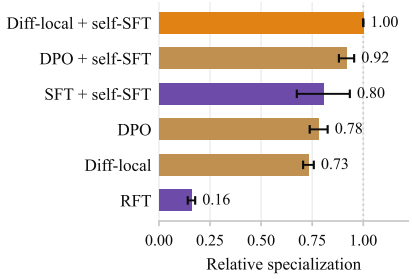


Figure 6: **RQ4: loss recipes.** Specialization of each recipe as a fraction of diff-local DPO + self-SFT, the recipe adopted in Sec. 3. Qwen3-8B, GSM8K hard set, mean and stderr on 3 seeds. The adopted recipe is orange, the other contrastive arms amber, and the arms without a contrast violet. Held-out retention stays between 0.88 and 0.94 for every arm, so the ordering reflects specialization and not differences in forgetting. SFT on the projection carries a wider spread across seeds.

In a typical round of projected distillation, we have access to wrong attempts y^-, their corrections y^+ and correct student responses y^c. These three quantities admit several learning objectives, and in this

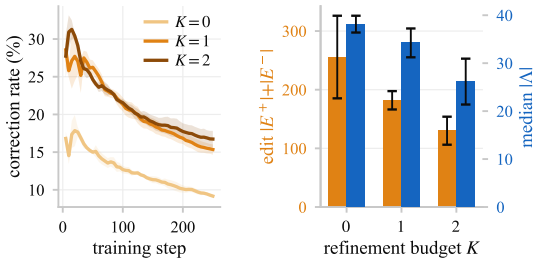


Figure 7: **RQ5: refining the projector at test time.** A frozen Qwen3-32B projector iteratively refines its own correction and the Qwen3-8B student trains on the result (GSM8K hard set, $K \in \{0, 1, 2\}$, 3 seeds), with the running correction rate on the left and edit size against off-policyness on the right. More refinement corrects more wrong attempts with a smaller edit, and the projector is never finetuned.

section we study an effective objective for learning. The simplest is supervised fine-tuning on the projection, optionally combined with a term on the student's own correct responses. Keeping only that second term is rejection fine-tuning (*RFT*; Yuan et al., 2023), which learns from the student's successes alone and ignores the projection. Further, because the wrong attempt is also retained, a contrastive objective is available: *DPO* (Rafailov et al., 2023) over the full response, again optionally combined with the self-SFT term. Since a minimal edit leaves most of the response unchanged, the contrast may instead be localized to the edited spans ((1)). We compare the six resulting recipes on the GSM8K hard set and report the outcome in Fig. 6.

The projection is necessary: training on the student's own correct responses alone recovers 0.16 of the adopted recipe, since the hard set is mined at pass@k=0 and the student's rollouts stay correct on about 6% of attempts through the run, so RFT has almost nothing to train on. Among objectives that do use the projection, the contrast is the more reliable choice: SFT on the projection reaches 0.81 on average but carries a standard error of 0.078 across seeds, three to fourteen times that of any contrastive arm. The self-SFT term then settles the ordering among those arms. A contrast constrains the likelihood of y^+ only relative to y^-, and a localized one does so over a handful of tokens, so both gain from a term that raises the likelihood of correct text outright, and the localized one gains more: without it the diff-localized contrast underperforms the full-sequence one (0.73 against 0.78), and with it the ordering reverses (1.00 against 0.92). We adopt the diff-localized contrast with self-SFT as the default recipe (Sec. 3); per-arm objectives and absolute values are in Sec. G.

4.5 RQ5: Can we improve the quality of projections for better learning?

In this section, we explore ways to make projections from a mid-capacity projector more effective. The natural approach would be to fine-tune the projector; we instead take a computationally efficient test-time route and ask a frozen Qwen3-32B to refine its own correction over several rounds. At each round it is re-prompted with its previous correction and the spans it edited, and across rounds we retain the valid correction most likely under the base student, measured by log-likelihood over the edited tokens. We sweep the refinement budget over $K \in \{0, 1, 2\}$ on a Qwen3-8B student and the GSM8K hard set with the default recipe.

Refinement improves the corrections in two ways (Fig. 7). More of the student's failures become correctable, the correction rate rising from 9.2% at K=0 to 15.4% at K=1 and 16.7% at K=2 (Fig. 7, left), so a larger part of the hard set contributes a training pair at all. At the same time the edits become smaller, from 256 to 130 tokens (Fig. 7, right), which by the account of Sec. 4.2 should make each target easier to learn from. Downstream, hard-set pass@1 rises from 0.37 to 0.44 while retention stays near 0.91–0.93, with most of the gain arriving in the first round. Test-time refinement can therefore improve the projections, leading to effective performance. Additional details are in Sec. H.

5 A Patch-MDP Analysis of Projected Distillation

Setup. We model generation as a deterministic *patch MDP* with a token budget H (Definition 2). A state is a prompt together with the tokens generated so far. An action is a *patch*, a span of tokens appended to that state, and every token carries a reward in $[0, 1]$. The student is a policy π in a class Π, with expected return $J(\pi)$ and optimum $J^\star$. Choosing action w at state s forfeits some return against the best action available in s. We denote this *optimality gap* with $\Delta(s, w)$, which is upper-bounded by a constant $\Delta_{\max}$ (Definition 3). Owing to telescoping (Lemma 1), a full rollout's shortfall can be accounted for one token at a time. The regret $J^\star - J(\pi)$ then equals H times the average per-token optimality gap, taken over π's state occupancy (Lemma 2). We write m for the *edit size*, the number of tokens the correction changes relative to the attempt. Secs. I.1 and I.2 give the full treatment.

Learning protocol. Training runs for M *rounds*. Each round is one batch of rollouts, the corrections made to them, and the gradient update that follows. π_i is the policy entering round i. Ideally we would query the projector at every student mistake as the student generates a response. That would let us minimize the gap under the student's own state occupancy. This is exactly the DAgger (Ross et al., 2011) approach, where an expert labels states under the learner's own occupancy, leading to a learning regret of $O(H)$, linear in the horizon. Per-mistake labeling as an LLM student responds is infeasible, since it demands a projector call and a forward pass at every mistake. In projected distillation, we instead correct the rollout once and perform a single forward pass over the response. This minimizes the optimality gap under a state occupancy drifted away from the student's own.
Fitting error. The projector edits the student's wrong response such that for each state in the rewritten response, the next token has zero optimality gap. We define two further quantities: (i) the policy fit error $\tilde{\varepsilon}$, the irreducible error in fitting the best policy in the class to the expert projections; and (ii) the optimization term $\tilde{\gamma}_M$, the student's excess disagreement over that best policy, which vanishes for a no-regret learner (Definition 5). Since a single disagreement costs at most $\Delta_{\max}$, the gap loss at these prefixes is bounded by $\Delta_{\max}$ times the disagreement rate (Lemma 3).
We assume that the drift caused by earlier token edits at a given decision point has a smooth, bounded effect on the optimality gap. We denote the maximum per-token effect on the optimality gap by η.

Assumption 1 (Prefix sensitivity)**.** Write $e(s, s')$ for the number of insertions and deletions carrying the generated tokens of one history into those of the other (Definition 4). The map $s \mapsto \mathbb{E}_{w\sim\pi(\cdot|s)}[\Delta(s, w)]$ is η-Lipschitz in that count:

$$\left|\mathbb{E}_{w\sim\pi(\cdot|s')}[\Delta(s', w)] - \mathbb{E}_{w\sim\pi(\cdot|s)}[\Delta(s, w)]\right| \;\le\; \eta\, e(s, s') \qquad \forall\, \pi \in \Pi,\ \forall\, s, s'.$$

Theorem 1 (Learning regret of projected distillation)**.** *Let Assumption 1 hold, let the projector be as above, and let m be the largest edit size over the rounds. Then after M rounds*

$$\min_{i\le M}\left[J^\star - J(\pi_i)\right] \;\le\; H\,\Delta_{\max}\left(\tilde{\varepsilon} + \tilde{\gamma}_M\right) \;+\; \eta\, m\, H.$$

The first term collects the fitting and optimization costs. The second is drift, which does not decrease with M; a smaller edit is the only lever on it.
Horizon regimes. For a minimal edit the edit size is $m = O(1)$, making the drift $O(\eta H)$ and hence the regret $O(H)$, linear in the horizon. A complete rewrite of the response leads to $m = \Theta(H)$, making the drift $O(\eta H^2)$ and therefore the regret $O(H^2)$, quadratic in the horizon (Corollary 1). In a controlled experiment in Sec. 4.2 we vary m (Definition 4) and observe that the student's performance suffers as the edit size increases The two endpoints recover the rates of DAgger and of behavior cloning, though not those algorithms themselves.

6 Conclusion

We introduced *projected distillation*, a post-training recipe to enable lower-capacity LLMs to solve hard problems beyond their current reach. We demonstrated that standard on-policy post-training methods such as RLVR and OPD struggle to both specialize on the hard task, whereas SFT on gold solutions leads to catastrophic forgetting. Projected distillation augments these training paradigms by iteratively transforming the student's failed attempts into correct solutions through minimal edits, keeping supervision close to the evolving student distribution. Our extensive empirical analyses showed that our approach achieved a better specialization–retention Pareto frontier than previous approaches. We showed that it is important to learn from minimal edits rather than from larger edits or full rewrites, and from the mistakes of the latest checkpoint rather than only those of the base model. Theoretical analysis with a smoothness assumption on the effects of token edits described a spectrum where minimal edits lead to DAgger-like linear-in-horizon regret, while full rewrites lead to behavior-cloning-like quadratic regret. Thus, our work provides empirical and theoretical evidence on the utility of the underlying continuum of on-policyness for continual learning on hard tasks.

Limitations and future work. In our setting, the projector uses a stronger external model; *self-projection*, where the student edits its own attempts, is a natural next step. We instantiate the projection as a minimal token edit; exploring likelihood-based alternatives is also an important future direction. The uniform per-token sensitivity assumed in Assumption 1 may be strong and needs further investigation. Our recipe assumes a cheap correctness check, leaving open the domains without a readily available verifier.

AI USE STATEMENT

Theory. We used generative AI tools to understand original DAgger line of work. While we manually designed the patch MDP, we took AI's help to provide a structure to prove edit-size dependence given the smoothness assumption. We manually checked, rederived and updated the initial proof structures. Later, the AI assisted us in checking the resulting arguments and flagging gaps. We adopted the theory after rigorous manual checks.

Experiments. We used generative AI tools to help implement our method and to help run the baselines from their publicly available implementations. The implementation of our method was reviewed by the authors, and the public baselines were tuned and sanity-checked manually to confirm they behaved as their original papers report. We did not use generative AI tools to propose or refine our hypotheses, to design the experiments or the evaluation protocol, or to generate, clean or reformat any dataset; the benchmarks are public and the hard sets are mined from them by a procedure the authors specified.

Results and presentation. We did not use generative AI tools to interpret our results. Empirical claims were drawn by the authors after visualization of the logged runs. We did use AI to produce the figures from our numerical results files, and to draft, edit and structure parts of the manuscript. Assistance with translation, with survey or interview questions, and with qualitative and thematic data analysis is not applicable to this work.

We take responsibility for the final content of this work, including all text, claims and artifacts produced with the aid of generative AI.

Separately from the above, the method we study uses a frozen language model as its projector, and the students we train are language models. These are the objects of study rather than tools used in preparing the paper, and are described in Secs. 3 and 4.

ETHICS STATEMENT

This work involves no human subjects and no personal or private data; the benchmarks and models we use are public, and the hard sets are derived from their training splits (Sec. D.2). In our black-box experiments, we draw corrections from a proprietary model, which we use only as a research instrument. We do not endorse distilling from proprietary models as a general practice.

REPRODUCIBILITY STATEMENT

The assumption behind our analysis is stated in Sec. 5 and discussed in Sec. I.1, with complete proofs of every lemma, the theorem and its corollaries in Sec. I.2 and the limits of the analysis in Sec. I.2.9. For the experiments, the method is specified in Sec. 3: one round of the procedure is summarized in Fig. 2, and the two objectives it optimizes are given in (1) and (2). Sec. D.1 lists the exact model checkpoints for every student and projector together with the optimizer, learning rates and per-step generation budget; Sec. D.2 gives the hard-set construction, including the sampling temperature and the number of attempts used to mine it, and the per-benchmark statistics; Sec. D.3 gives the prompt templates issued to the projector; Sec. D.4 the remaining training hyperparameters; and Sec. D.5 the evaluation protocol. Per-cell results behind every pooled number in the main text are in Sec. E.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. *arXiv preprint arXiv:2306.13649*, 2023. URL https://arxiv.org/abs/2306.13649.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*, 2023. URL https://arxiv.org/abs/2310.20689. LeMa: a strong model identifies, explains and repairs the faulty step of a wrong reasoning path; the student is fine-tuned on the mistake–correction pairs.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision (ECCV)*, 2018. URL https://arxiv.org/abs/1801.10112.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xiwen Chen, Jingjing Wang, Wenhui Zhu, Peijie Qiu, Xuanzhao Dong, Yueyue Deng, Hejian Sang, Zhipeng Wang, Alborz Geramifard, and Feng Luo. Soda: Semi on-policy black-box distillation for large language models. *arXiv preprint arXiv:2604.03873*, 2026. URL https://arxiv.org/abs/2604.03873.

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

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*, 2024. URL https://arxiv.org/abs/2406.09098.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023. URL https://arxiv.org/abs/2308.08998.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021. URL https://arxiv.org/abs/2103.03874.

D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975. URL https://doi.org/10.1145/360825.360861.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026. URL https://arxiv.org/abs/2601.20802. Introduces Self-Distillation Policy Optimization (SDPO) for RL with rich (textual) feedback.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. URL https://arxiv.org/abs/1708.02072.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in

neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017. URL https://arxiv.org/abs/1612.00796.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2202.10054.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024. URL https://arxiv.org/abs/2409.12917. SCoRe: multi-turn online RL for self-correction from entirely self-generated data.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-DPO: Step-wise preference optimization for long-chain reasoning of LLMs. *arXiv preprint arXiv:2406.18629*, 2024. URL https://arxiv.org/abs/2406.18629.

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

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

Wenye Lin and Kai Han. Surgical post-training: Proximal on-policy distillation for reasoning with knowledge retention. *arXiv preprint arXiv:2603.01683*, 2026. URL https://arxiv.org/abs/2603.01683. Introduces SPOT: an Oracle minimally edits erroneous reasoning steps to produce proximal on-policy data.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Lin, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances LLM's reasoning capability. *arXiv preprint arXiv:2411.19943*, 2024. URL https://arxiv.org/abs/2411.19943.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1706.08840.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.17651.

OLMo Team. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024. URL https://arxiv.org/abs/2501.00656.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023. URL https://arxiv.org/abs/2305.18290.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. URL https://arxiv.org/abs/1011.0686.

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

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

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of LLMs should leverage suboptimal, on-policy data. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2404.14367.

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. URL https://arxiv.org/abs/2306.05301.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974. URL https://doi.org/10.1145/321796.321811.

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. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2109.01652.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E Gonzalez, Bin Cui, and Shuicheng Yan. Supercorrect: Advancing small llm reasoning with thought template distillation and self-correction. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.09008.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2402.13669.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023. URL https://arxiv.org/abs/2308.01825. Introduces Rejection sampling Fine-Tuning (RFT).

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025. URL https://arxiv.org/abs/2504.13837. RLVR biases the output distribution toward rewarded paths, improving sampling efficiency at small k while narrowing the pass@k reasoning boundary at large k.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022. URL https://arxiv.org/abs/2203.14465.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*, 2024. URL https://arxiv.org/abs/2404.11999.

Yaocheng Zhang, Jiajun Chai, Yuqian Fu, Songjun Tu, Xiaohan Wang, Wei Lin, Guojun Yin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Are full rollouts necessary for on-policy distillation? *arXiv preprint arXiv:2605.31490*, 2026a.

Zhaoyang Zhang, Shuli Jiang, Yantao Shen, Yuting Zhang, Dhananjay Ram, Shuo Yang, Zhuowen Tu, Wei Xia, and Stefano Soatto. Reinforcement-aware knowledge distillation for llm reasoning. *arXiv preprint arXiv:2602.22495*, 2026b. URL https://arxiv.org/abs/2602.22495. Introduces RL-aware distillation (RLAD) and Trust Region Ratio Distillation (TRRD).

A NOTATION

Tab. 3 consolidates the notation used throughout the paper, with each symbol's meaning and where it is defined or used.

Table 3: Notation reference.

Problem, student, verifier (Sec. 3)	
x	problem / prompt (drawn from a problem distribution)
$y \in \mathcal{V}^{\leq H}$	answer: a token sequence of length $\leq H$
π_{θ_S}	student policy (autoregressive, parameters θ_S)
$\pi_{\text{ref}} = \pi_{\theta_{S,0}}$	frozen initial student, used as the DPO reference
$R(x, y) \in \{0, 1\}$	verifier (marks an answer correct or not)
$\mathcal{C}(x)$	set of correct answers $\{y : R(x, y) = 1\}$
pass@k=0	problems on which all k sampled attempts are wrong
Projection and objective (Definition 1, Equation (1))	
Λ	likelihood shift the correction demands of the student
$\text{proj}(x)$	projection: the most likely correct answer (Definition 1)
$y^- = y$	student's wrong attempt (rejected)
y^+	the projector's minimal-edit correction (chosen)
y^{c}	student's own correct rollout (self-SFT target)
$E^-,\ E^+$	mistake span (unlearn) and fix span (learn) from the token-diff
β	DPO contrast strength (temperature)
$\mathcal{L}_{\text{Diff-Local-DPO}},\ \mathcal{L}_{\text{SFT}},\ \mathcal{L}$	diff-local contrast; self-SFT; full objective
Patch MDP (Definition 2, Sec. I.1)	
$\mathcal{V},\ H$	vocabulary; horizon, the token budget of a rollout
$s_t = (x, y_{1:t})$	state before decision t: prompt and the first t generated tokens
w	action: a variable-length patch (a continuation that fits the budget)
$\mathcal{A}(s)$	admissible actions $\{w \in \mathcal{V}^h : 1 \leq h \leq H - \lvert s\rvert\}$
$p:\ s' = s \oplus w$	deterministic transition (append the patch)
$r_{\text{tok}} \in [0, 1]$	per-token reward; $r(s, w)$ sums it over the patch
ρ	initial state distribution: the distribution over problems
$\gamma = 1$	no discounting
$\pi,\ \pi_\theta$	policy; underlying token model
$V^\star,\ Q^\star$	optimal value and action-value
$J(\pi),\ J^\star$	expected return of π; its optimum, (6)
$d^\pi,\ \tilde{d}_i$	occupancy induced by π; the corrected prefixes of round i
$\Delta(s, w)$	optimality gap $V^\star(s) - Q^\star(s, w) \in [0, \Delta_{\max}]$
The regret analysis (Sec. 5, Theorem 1)	
$M,\ \pi_i$	rounds of training; the learner entering round i
$\Pi,\ \Theta$	policy class and its parameter set (Sec. I.1)
$w^\star(s)$	the projector's token at s, assumed optimal there
m	edit size of a correction (Definition 4); in Theorem 1, its largest value over rounds
$e(s, s')$	edit count between two histories (Definition 4)
$e_t,\ \bar{e}_i$	edit count at decision t; its mean over round i (Lemma 4)
η	per-edited-token drift (Assumption 1)
$\Delta_{\max}$	largest return one decision can forfeit (Definition 3)
$L(d, \pi)$	mean gap of π under a state distribution d, (7)
$\tilde{\varepsilon}_i(\pi)$	rate at which π disagrees with $w^\star$ at those prefixes
$\tilde{\varepsilon}$	approximation error: the disagreement of the best policy in the class
$\tilde{\gamma}_M$	optimization error: the learner's excess disagreement over it
$D,\ G$	parameter-set diameter; gradient bound (Proposition 1)

B EXTENDED RELATED WORK

We expand the related work summarized in Sec. 2.

On-policy and self-distillation. On-policy distillation supervises the student's own generations with a teacher, closing the exposure-bias gap of sequence-level KD; GKD (Agarwal et al., 2023) is the canonical instance and our OPD baseline, and recent work analyzes when it succeeds (Li et al., 2026). It needs the teacher's token distribution, which a black-box model does not expose. Self-training removes the teacher instead: STaR (Zelikman et al., 2022) and ReST (Gulcehre et al., 2023) keep whatever verifies and fine-tune on it, which is the self-SFT term of (2), and self-distillation (Yang et al., 2024; Hübotter et al., 2026) drops it in a different way. Both depend on the model already producing something correct, which on the $\mathrm{pass@}k{=}0$ set it does not. STaR's rationalization is the nearest prior analogue of our projector prompt; it writes a fresh rationale from the correct answer rather than editing the attempt, so nothing holds the target near the model.

Preference-optimization distillation. Casting black-box distillation as preference optimization (Rafailov et al., 2023) removes the need for teacher logits, and this is the family we belong to. The concurrent work differs from us in the negative: SODA (Chen et al., 2026) draws it from a frozen copy of a student already fine-tuned on the teacher's corrections, and SuperCorrect (Yang et al., 2025) follows teacher-authored traces, where ours is the student's own error sampled in the round it is used. Restricting a preference objective below the response level is also established, through per-token KL (Zeng et al., 2024), step boundaries (Lai et al., 2024), or tokens found critical by search (Lin et al., 2024). We share the restriction and differ in where the region comes from: a minimal edit makes the token-diff itself the segmentation, so no annotation, reward model or search is needed.

RL from verifiable rewards. Verifiable-reward RL (Shao et al., 2024; Schulman et al., 2017) is the on-policy ideal, and it stops where the reward does. Yue et al. (2025) characterize what it achieves: better sampling efficiency within the base model's reasoning boundary rather than an extension of that boundary. On the $\mathrm{pass@}k{=}0$ set there is nothing inside the boundary to sharpen, and the projection supplies a reachable target. Tajwar et al. (2024) find that on-policy sampling with an objective that pushes down on bad responses beats offline maximum-likelihood, and that poor on-policy data is still worth using, which is why we keep the model's own wrong attempt as the negative. Their setting assumes an on-policy positive; ours has none, so we manufacture one.

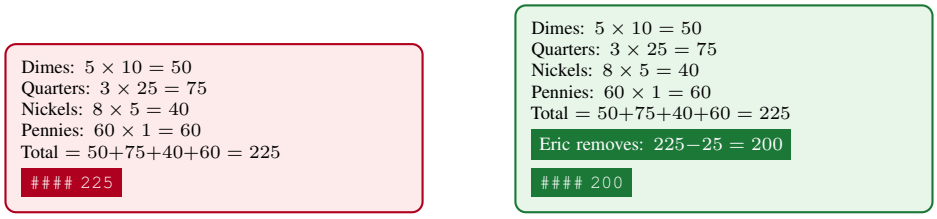
Imitation learning and DAgger. The analysis of Sec. 5 instantiates the DAgger (Ross et al., 2011) dichotomy between linear and quadratic regret in the horizon. In DAgger the regime is set by the environment's cost-to-go; here it is set by the size of the edit, which the method chooses. Concurrent RLAD (Zhang et al., 2026b) also learns from student rollouts, through a trust-region likelihood-ratio objective rather than a contrast on the edited spans.

Correction and refinement distillation. Supervising a model on corrections of its flawed attempts is an established signal (Yang et al., 2025; Madaan et al., 2023), usually aimed at teaching self-correction at inference time rather than at solving problems the model cannot. LeMa (An et al., 2023) established corrections as training data, repairing a wrong reasoning path and explaining the mistake. It sits far from us on every axis we measure: its wrong paths come from other models, its corrections are mined once, its repairs are free-form, and it learns by fine-tuning. SPOT and SODA, which we run as baselines, sit close on several. SPOT treats proximity as a filter, discarding corrections that rewrite too much of the chain; that also discards the problems which needed the most rewriting, so the comparison cannot separate the size of an edit from the difficulty of the problem that required it. We hold the problem set fixed and vary the rewrite instead (Sec. 4.2). Neither SPOT nor LeMa recomputes its corrections as the student changes (Sec. 4.3). SCoRe (Kumar et al., 2024) trains self-correction as a test-time skill and finds that offline correction traces fail through distribution mismatch, the same failure our static pool shows.

Catastrophic forgetting. Fine-tuning a strong model on a narrow task erodes unrelated abilities. Classical remedies act on the optimizer, through regularization (Kirkpatrick et al., 2017), replay, or adapters. We act on the target instead: a correct answer that stays near what the model already writes moves it less and forgets less (Secs. 4.1 and 4.2). SPOT (Lin & Han, 2026) argues that a KL-constrained reward preserves knowledge, and the reference term of (1) plays the same role. We add the choice of target to that account.

C PROJECTION EXAMPLES

This appendix shows complete wrong→projected pairs, so the minimal-edit property can be read at the level of full solutions. In each pair the projector reuses the student's derivation verbatim and changes only what is needed to make the answer correct; the changed spans are the token-diff E^- (mistake) and E^+ (fix) that the diff-local DPO loss (1) contrasts. All student answers are from Qwen3-8B (base); all projections are from the black-box projector. Sec. C.1 works through the single-edit example of Fig. 8 in full, and Sec. C.2 a multi-edit one.



(a) Student's wrong attempt (keeps a quarter).

(b) Correction inserts the single missing step.

Figure 8: **A minimal-edit projection** on GSM8K (*Cindy's wishing pond*, gold answer 200). The student's wrong attempt (a) and the projector's correction (b) are token-identical except for one inserted step. Only the changed spans $E^- \to E^+$ enter the contrast in (1).

C.1 A SINGLE EDIT: CINDY'S WISHING POND

Problem. Cindy tosses 5 dimes into the wishing pond. Eric flips 3 quarters into the pond. Garrick throws in 8 nickels. Ivy then drops 60 pennies in. *If Eric dips his hands into the water and pulls out a quarter*, how much money, in cents, did they put into the pond? **(Gold: 200.)**

y^- (rejected, wrong — #### 225)

Dimes: $5 \times 10 = 50$. Quarters: $3 \times 25 = 75$. Nickels: $8 \times 5 = 40$. Pennies: $60 \times 1 = 60$. Total $= 50 + 75 + 40 + 60 = 225$. `#### 225`

y^+ (chosen, minimal edit — #### 200)

Dimes: $5 \times 10 = 50$. Quarters: $3 \times 25 = 75$. Nickels: $8 \times 5 = 40$. Pennies: $60 \times 1 = 60$. Total $= 50 + 75 + 40 + 60 = 225$. **Eric removes a quarter:** $225 - 25 = 200$**.** `#### 200`

The entire count is reused verbatim; the projector inserts the single step the student omitted (subtracting the withdrawn quarter). The token-diff is a one-span edit: $E^- = \{$`#### 225`$\}$, $E^+ = \{$"Eric removes... $= 200$", `#### 200`$\}$.

C.2 SEVERAL EDITS: JOHN'S COMPUTER

Problem. John's new computer cost $1500. He needs a monitor and other peripherals that cost 1/5 that much. He also wants to upgrade the $300 video card with one that costs *twice as much*. How much did he spend? **(Gold: 2100.)**

y^- (rejected, wrong — #### 2400)

Computer: $1500. Peripherals: $\frac{1}{5} \times 1500 = 300$. Upgraded video card: $2 \times 300 = 600$. Total: $1500 + 300 + 600 = 2400$. `#### 2400`

y^+ (chosen, minimal edit — #### 2100)

Computer: $1500. Peripherals: $\frac{1}{5} \times 1500 = 300$. Upgraded video card: $2 \times 300 = 600$. **Since the base computer already includes the $300 card, he pays only the extra:** $600 - 300 = 300$**.** Total: $1500 + 300 +$ **300** $= 2100$. `#### 2100`

Here two spans change—the video-card line and the total—while the computer and peripheral steps are reused unchanged. This is a multi-edit projection: the correction threads the fix (charging only the upgrade difference) through to the final sum.

D EXPERIMENTAL SETUP

D.1 REPRODUCIBILITY DETAILS

Model families and projectors. The two students are open-weight instruction-tuned models from different families, **Qwen3-8B** (Qwen Team, 2025) and **OLMo-2-7B** (OLMo Team, 2024), chosen so the effect is tested as a property of the method rather than of one pretraining recipe. The same-family projector shares the student's tokenizer, as OPD requires, and unless noted it is **Qwen3-32B** for the Qwen student and **OLMo-2-32B** for the OLMo student. The black-box path (Sec. 3) needs only the correction text, so it is tokenizer-free and draws its corrections from a single black-box model shared across students. The exact checkpoints are: students `Qwen/Qwen3-8B` and `allenai/OLMo-2-1124-7B-Instruct`; same-family 32B teachers `Qwen/Qwen3-32B` and `allenai/OLMo-2-0325-32B-Instruct`.

Benchmarks. Five benchmarks spanning math, code, tools, and science, each contributing a mined hard set for training and a held-out slice for evaluation:

- **GSM8K** (Cobbe et al., 2021): grade-school math, numeric answers graded by exact match. Mine from the 7,473-example `train` split; evaluate on a 200-problem held-out slice of the `test` split.
- **MATH** (Hendrycks et al., 2021): competition math, `\boxed{}` answers graded by symbolic equivalence. Mine from the 12k `train` split; evaluate on the MATH-500 slice (Lightman et al., 2023).
- **ToolAlpaca** (Tang et al., 2023): tool use and API calling. Mine from the 468-API, ∼4,255-instance release. The model must emit the first tool call for a request, and a call counts as correct when the action is a valid function of that API, its arguments parse as JSON with all required parameters present, and the action matches the reference call.
- **LiveCodeBench v6** (Jain et al., 2024): contamination-resistant competitive-programming problems, graded by unit-test execution (pass@1). Mine hard problems from the v6 release; evaluate on a held-out slice.
- **SciKnowEval** (Feng et al., 2024): multi-level scientific knowledge. We use the L3 four-choice subset (4,135 items) of the 28,392-item release, graded by a letter match against the reference option.

Evaluation protocol. Specialization and retention are evaluated identically at step 0 and every 25 steps thereafter, so each student and benchmark serves as its own reference; the continual-learning summaries built from these two series are defined in Sec. E. Every benchmark is scored by the grader listed above, always as sampled pass@1 on its held-out slice, a 200-problem slice for GSM8K. The full-suite evaluations of Sec. E.3 instead score the final checkpoints on complete test splits. Answer extraction is per benchmark: the `####`-delimited final number for GSM8K, with a last-number fallback for free-form generations; the `\boxed{}` expression for MATH, compared by symbolic equivalence; the final `Answer:` letter for SciKnowEval; the first `Action` and `Action Input` pair for ToolAlpaca; and the extracted program for LiveCodeBench, executed against the unit tests.

Hardware and software. All experiments ran on a single pod of 8× NVIDIA A100-SXM4 (40 GB) GPUs, which was the compute available to us. The stack is PyTorch 2.8.0 (CUDA 12.8), TRL 1.6.0, Transformers 5.12.1, and Accelerate 1.14.0. Multi-GPU training uses fully-sharded data parallelism (FSDP2, full-shard) with gradient checkpointing. Final checkpoints are gathered and consolidated from the sharded (DTensor) state.

Training configuration. Unless noted, all methods train for 250 update steps at a constant learning rate with the Adam optimizer in bf16, computing loss on assistant response tokens only. GRPO and OPD use 5×10^{-6}, and SPOT, SODA, and projected distillation use 1×10^{-5}. The per-step generation budget is matched at 8 rollouts: the distillation methods (OPD, SPOT, SODA, and ours) draw 8 problems per step, while GRPO draws a group of 8 samples on a single problem. The

DPO-based objectives use β=0.1; SODA runs a brief SFT warmup on y^+ (first 25 steps) before its preference phase. GRPO keeps TRL's default KL coefficient of zero toward the reference policy.

D.2 Hard-Set Construction and Statistics

Mining rule. Training targets problems the student cannot already solve. For each candidate problem we draw k=16 samples from the *untrained* student at temperature 0.8 (seed 0) and keep the problem only if all 16 are graded wrong by that benchmark's grader, i.e. pass@16 = 0. The retained problems form the frozen hard set; the 16 failed samples are stored with each record, which is what allows an offline arm to be built without re-generating.

Because the criterion is defined against a specific student, **hard sets are per student**: the same benchmark yields a different hard set for Qwen3-8B than for OLMo-2-7B, and a checkpoint is only ever scored against the hard set mined for its own base model.

Tab. 4 reports the mining statistics for Qwen3-8B and Tab. 5 the split sizes for OLMo-2-7B. Hard-set size is a property of the *student*, not of the benchmark: GSM8K is hard for only 1.63% of Qwen3-8B's training pool, whereas MATH yields roughly twice as many hard problems for OLMo-2-7B (4,046) as for Qwen3-8B (1,991).

Table 4: **Hard-set mining for Qwen3-8B.** A problem enters the hard set when all 16 samples drawn from the untrained student at T=0.8 are wrong. "Base sample accuracy" is the fraction of all 16×(screened) samples that were correct, i.e. the student's accuracy over the pool before any training. The evaluation slice is disjoint from the hard set and comes from the benchmark's `test` split.

Benchmark	Screened	Hard set	Hard %	Base sample acc.	Eval slice
GSM8K	7,473	122	1.63	0.949	200
MATH	12,000	1,991	16.59	0.703	500
SciKnowEval	3,721	1,003	26.96	0.323	400
ToolAlpaca	3,584	467	13.03	0.845	300
LiveCodeBench	—	100	—	—	50

Table 5: **Hard-set sizes for OLMo-2-7B.** The mining summary files were not retained for these runs, so the screened pool size and base sample accuracy are unavailable; the split sizes are taken directly from the mined files.

Benchmark	Hard set	Eval slice
GSM8K	230	200
MATH	4,046	200
SciKnowEval	493	200
ToolAlpaca	471	200
LiveCodeBench	810	131

Train/evaluation disjointness. We verify rather than assume separation. Comparing question text lower-cased with LaTeX commands stripped and non-alphanumeric characters removed — so that a problem reformatted between two source releases is still caught, which a byte-level comparison would miss — gives zero overlap between hard set and evaluation slice for both math benchmarks: 1,991 versus 500 for MATH and 122 versus 200 for GSM8K. AMC23, MMLU and HumanEval are evaluation-only and were never trained on by any method. Four problems appear twice inside the MATH hard set (1,991 records, 1,987 unique) and therefore carry double weight during training.

D.3 Prompt Templates

All prompts are single user turns passed through the model's own chat template. For Qwen3 the thinking mode is disabled in mining, training and evaluation alike, so the same decoding regime holds throughout. The instruction appended to a question is selected by the benchmark's answer type, which fixes the output format the grader expects; the grader always follows the *evaluation* benchmark, not the benchmark a checkpoint was trained on.

D.3.1 STUDENT PROMPTS

The student prompt is the question followed by one of the four instructions below.

Numeric (GSM8K):

```
Solve the problem by showing your full step-by-step
reasoning.  After the reasoning, write the final answer on a
new line by itself in the exact format "#### N", where N is
just the final number (no units, words, or commas).  Always
include this #### line as the last line.
```

Boxed (MATH, MATH-500, AMC23):

```
Solve the problem by showing your full step-by-step
reasoning.  Put your final answer inside \boxed{...} at
the end.  The content of the box must be exactly the final
answer (a number or a simplified expression), with no
surrounding words.
```

Multiple choice (SciKnowEval, MMLU):

```
Answer the multiple-choice question.  Think step by step,
then write the final answer on a new line by itself as
"Answer:  X" where X is A, B, C, or D.
```

Code (LiveCodeBench, HumanEval):

```
Complete the Python function.  Return the full function
definition inside a "'python code block.  Do not include
tests or explanation.
```

D.3.2 PROJECTION PROMPT

The projector is instructed to *edit* the student's wrong attempt rather than to solve the problem afresh; this is what keeps the target near the student's own distribution and makes the token-level diff small. The role is set per domain ("expert math tutor" for numeric and boxed answers, "expert science tutor" for multiple choice, "expert assistant for tool/function calling" for tool use).

System message:

```
You are an {role}.  A student answered a problem incorrectly.
PROJECT their attempt onto a correct solution:  keep the
student's overall approach, wording, and structure as
much as possible, changing ONLY what is needed to make it
correct.  Do NOT rewrite from scratch in your own style
-- stay as close to how the student writes as possible.
{end-format requirement}
```

User message:

```
Problem:
{problem}
Student's incorrect attempt:
{wrong attempt}
Minimally edit the student's attempt into a correct
solution, staying as close to their style as possible.
{end-format requirement}
```

The end-format requirement names the graded target exactly: `End with the correct final answer, {gold}.` for math, the correct option for multiple choice, and the required `Action` / `Action Input` pair for tool calling. A projection that does not end in the graded format is **rejected by the verifier**, so a correction cannot be accepted on the strength of its prose alone; every accepted projection has been re-graded by the same grader used for evaluation.

Code is different in kind. For programming problems the correctness signal given to the projector is *execution feedback on the student's own attempt* — which tests failed, with input, expected and actual values — and not the hidden test suite:

```
You are an expert competitive programmer helping a student.
The student wrote a solution that FAILS some tests.  PROJECT
their attempt onto a correct one:  keep the student's
overall approach, structure, and variable names as much as
possible, changing ONLY what is needed to make it pass.  Do
NOT rewrite from scratch in your own style -- stay as close
to how the student writes as possible.  Return the corrected
solution as a single code block.
```

D.4 Training Configuration: Additional Details

Sec. D.1 gives the models, learning rates, optimizer, per-step generation budget and hardware. This section records the details that a re-implementation additionally needs.

Generation budget per domain, and why it varies. The rollout budget must be at least the budget the hard set was mined with. A problem is in the hard set precisely because the student failed it under the mining budget, so giving the student a *smaller* budget during training truncates its derivation mid-proof and the problem cannot be solved even in principle. Competition MATH attempts run a median of roughly 805 tokens, with 61% needing at least 690, so a 700-token budget both pinned hard-set accuracy near zero and trained the model on truncated derivations, collapsing held-out accuracy from 0.39 to 0.16. The budgets used are therefore $1{,}024$ tokens for MATH, $1{,}536$ for LiveCodeBench, and 700 for GSM8K, SciKnowEval and ToolAlpaca, whose chains genuinely are short. Rollouts are sampled at $T{=}0.8$. The edit-size runs of Sec. 4.2 are the one exception: they were launched with a $2{,}048$-token GSM8K budget rather than 700, which is why the mean deleted span reported for the *gold* arm in Tab. 11 exceeds 700 tokens.

Projector configuration. The black-box projector is a frontier instruction-tuned model accessed through a hosted API and run off-GPU, which is what allows the whole pod to be spent on the student. Each wrong sample gets at most two projection attempts and the first verified-correct projection is kept; if neither verifies, the sample contributes no preference pair for that step. The same-family projector variant is served locally.

GPU allocation. Of the eight GPUs in the pod, seven hold FSDP2 shards of the student and one is reserved for the in-loop vLLM generator that produces on-policy rollouts and runs the periodic evaluations. Sharing the generator GPU with a training rank starves the generator and stalls the step, so the split is fixed rather than dynamic.

Loss variants. The arms and their objectives are given in Sec. G. Two further switches are fixed throughout: the adopted default uses unnormalized rather than token-mean-normalized DPO log-probabilities, and the auxiliary SFT term acts on the student's own correct rollouts y^{c}, as in (2), not on the projection y^{+}.

D.5 Evaluation: Statistical Resolution and Seed Variance

Two kinds of seed. Two sources of variation are easy to conflate and differ by roughly a factor of fifty in magnitude.

- A **training seed** indexes an independently trained checkpoint. Different weights. The spread is large: the three math-trained projected-distillation seeds score 0.749, 0.311 and 0.688 on MATH-500.
- An **evaluation seed** indexes which generations were sampled from *one fixed* checkpoint. The model is identical across replicates. The spread is small, typically ± 0.006 on the large suites.

The $\pm$ reported with every sampled pass@1 number is the standard error over evaluation seeds (Sec. E.3). It therefore certifies that a measurement would reproduce under re-sampling; it says nothing about whether a different training run would land near it. Claims about a *method* rather than about a particular checkpoint have to be read against training-seed spread, which is the dominant term.

Greedy decoding cannot be averaged this way. It is deterministic, so replicates return the same number and a standard deviation of exactly zero, which reports nothing about how well the mean is pinned down. Sampling one generation per problem and repeating over independent seeds gives an error bar that means something. For pass@k with $k > 1$ we use the unbiased estimator of Chen et al. (2021), pass@$k = 1 - \binom{n-c}{k} / \binom{n}{k}$, drawing n=8 samples rather than estimating pass@k from exactly k samples, which is high-variance and biased upward.

Resolution limits of the small suites. Suite size sets the granularity of any difference that can be claimed. AMC23 has only 40 problems, so one problem is 0.025 and the base standard error alone is ± 0.030; differences below roughly 0.06 there should not be argued. GSM8K (1,319) and MMLU (14,042) are the suites where small differences resolve.

A note on greedy reproducibility. Repeating a greedy evaluation with identical prompts and identical suite size did not return bit-identical scores. The cause is continuous-batching non-determinism in the inference server, which changes numerics enough to flip borderline argmax tokens. Greedy numbers therefore carry a small amount of unreported jitter of their own, which is a further reason to report a sampled mean with an explicit error bar.

E RQ1: Comparison with baselines

E.1 Metrics

Area under the Pareto frontier (AUPF). For a single run, evaluation at checkpoints $i = 1, \ldots, n$ yields points $(x_i, y_i) \in [0,1]^2$, where x_i is specialization (hard-set pass@1) and y_i is retention (held-out pass@1). A point is *dominated* if another point is at least as good on both axes, and the Pareto frontier is the non-dominated set

$$\mathcal{F} = \{\, i : \nexists\, j \text{ with } x_j \geq x_i,\ y_j \geq y_i,\ (x_j, y_j) \neq (x_i, y_i) \,\}. \tag{3}$$

We anchor the frontier at a *shared untrained base* $B = (0, H_0)$: specialization 0 and retention H_0, the untrained held-out accuracy, both estimated as the mean step-0 value over the methods that start untrained. Specialization at the anchor is measured at step 0 rather than inferred from the mining criterion; it averages 0.012 across those methods, which we round to zero. B is a *forced* left anchor (it is never subject to domination); using a shared base rather than each run's own first checkpoint prevents methods that begin with an SFT warmup (e.g. SODA) from being scored against an already-elevated starting specialization. Indexing $\{B\} \cup \mathcal{F}$ in order of increasing specialization, $0 = \tilde{x}_0 < \tilde{x}_1 < \cdots < \tilde{x}_K$, AUPF is the area under the piecewise-linear frontier $y_{\mathcal{F}}(\cdot)$ by the trapezoidal rule:

$$\text{AUPF} \;=\; \int_0^{\tilde{x}_K} y_{\mathcal{F}}(x)\, dx \;=\; \sum_{k=1}^{K} \tfrac{1}{2}\,(\tilde{y}_{k-1} + \tilde{y}_k)\,(\tilde{x}_k - \tilde{x}_{k-1}). \tag{4}$$

We compute AUPF per seed and report the mean $\pm$ s.e. across seeds. Because retention is integrated over specialization from the base at x=0, a method that never specializes ($\tilde{x}_K \approx 0$) has AUPF ≈ 0; a higher AUPF means pushing specialization further while holding retention.

Continual-learning metrics (Tab. 2). For a run with per-step specialization x_t and retention y_t ($t = 1, \ldots, T$) and the shared untrained base $(0, H_0)$, define

$$\bar{x} = \tfrac{1}{T}\textstyle\sum_{t=1}^{T} x_t, \qquad \bar{y} = \tfrac{1}{T}\textstyle\sum_{t=1}^{T} y_t,$$
$$\text{BWT} = y_T - H_0, \qquad F = \max_{\tau \leq T} y_\tau - y_T \text{ with } y_0{:=}H_0,$$
$$\text{SafeSpec}_\varepsilon = \max\{x_t : y_t \geq H_0 - \varepsilon\}, \qquad 0 \text{ if the set is empty.}$$

$\bar{x}, \bar{y}$ are training-averaged accuracies in the spirit of the AUC metrics of Kemker et al. (2018); BWT is the backward transfer of Lopez-Paz & Ranzato (2017) and F the forgetting measure of Chaudhry et al. (2018). $\text{SafeSpec}_\varepsilon$ is the highest specialization reached at any checkpoint whose retention is within ε of the untrained base. For the aggregate row in Tab. 2, retention is expressed as a fraction of base so panels with different retention scales are commensurate: $\bar{y}_{\text{norm}} = \bar{y}/H_0$, $F_{\text{norm}} = F/H_0$, $\text{BWT}_{\text{norm}} = (y_T - H_0)/H_0$, and pooled SafeSpec uses $y_t/H_0 \geq 1 - \varepsilon_{\text{rel}}$ with $\varepsilon_{\text{rel}} \in \{0.02, 0.05\}$. Each metric is computed per seed and averaged over 3 seeds; the aggregate pools further over (family, benchmark), with GRPO on Qwen-LiveCodeBench entered as no learning.

E.2 PER-CELL RESULTS

Tab. 6 gives the exact final-checkpoint accuracies behind Fig. 3: hard-set pass@1 (specialization) and held-out pass@1 (retention), as mean ± standard error across seeds at 250 steps. Fig. 9 breaks that figure down per benchmark, Fig. 10 gives the pooled trajectories over training, and Tabs. 7 and 8 break the two continual-learning summaries of Tab. 2 down per cell.

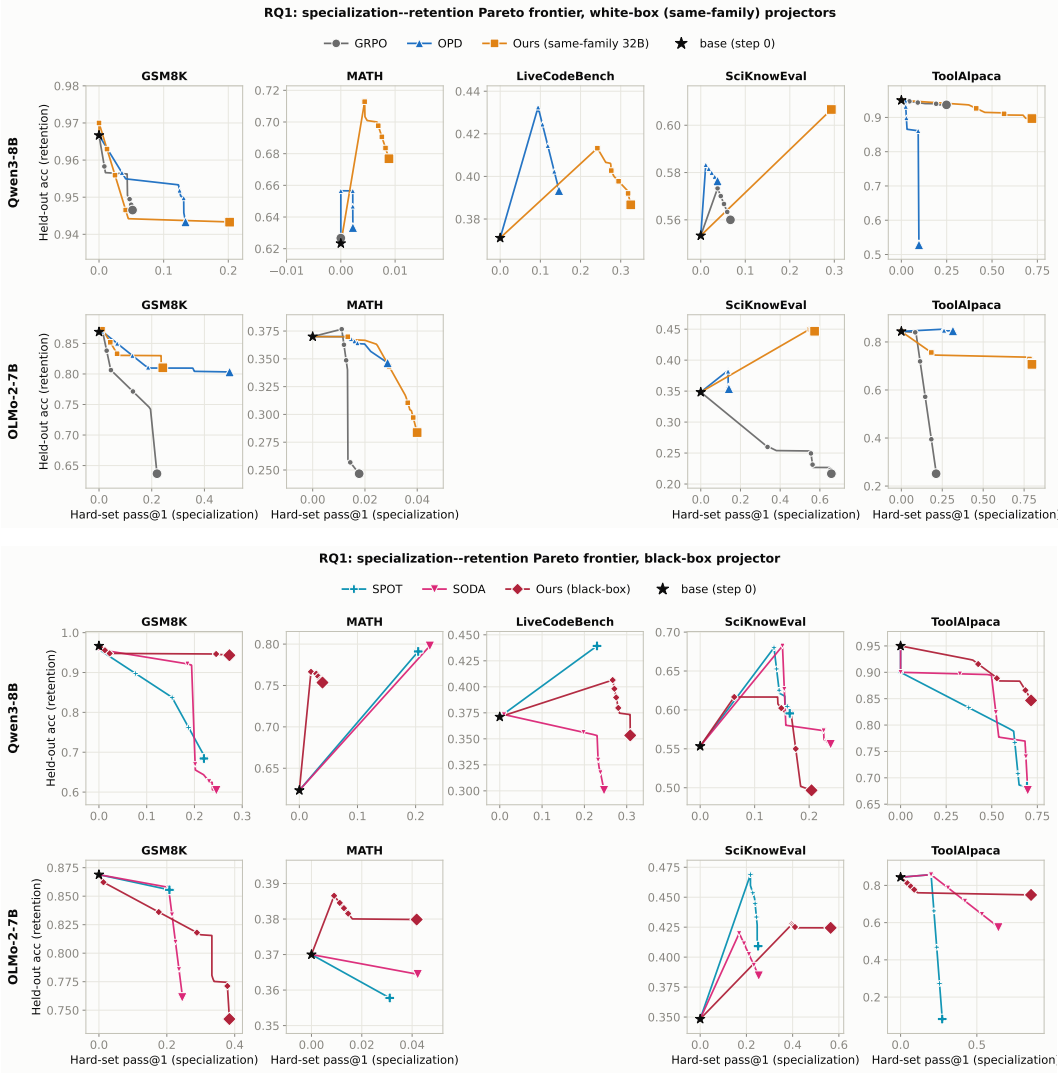


Figure 9: **Per-benchmark specialization–retention Pareto frontier** (detailed breakdown of Fig. 3). **Top:** white-box (same-family) projectors; **bottom:** black-box projector. Each block has two rows (Qwen3-8B, OLMo-2-7B) and one column per benchmark. Axes: hard-set pass@1 (x, specialization) and held-out pass@1 (y, retention). Bold line = monotone Pareto frontier of the seed-averaged trajectory (larger marker = final checkpoint) anchored at the untrained base $\star$ (specialization 0); axes auto-scale per panel. GRPO on Qwen-LiveCodeBench is entered as no learning.

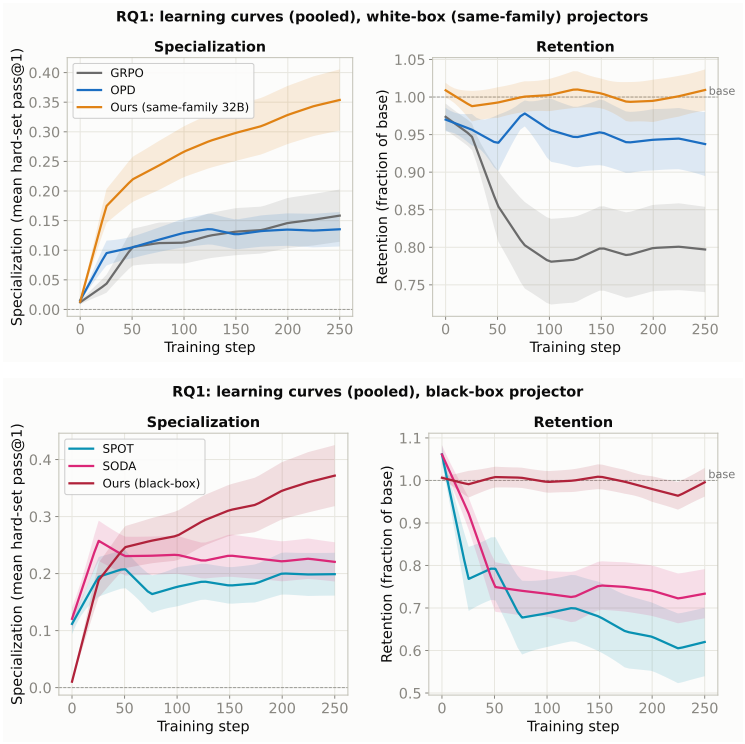


Figure 10: **RQ1 pooled learning curves. Top:** white-box (same-family) projectors; **bottom:** black-box projector. *Left panels:* specialization (mean hard-set pass@1) over training. *Right panels:* retention as a fraction of the untrained base (base = 1.0 dashed). Curves pool over students, benchmarks, and seeds (same pooling as Fig. 3); shaded bands are s.e. over the pooled runs. The early specialization and subsequent retention loss of SPOT and SODA (retention falls by step 50) and the retention preservation of both Ours variants are visible directly. GRPO on Qwen-LiveCodeBench entered as no learning.

Table 6: **RQ1 per-cell final accuracies** at the 250-step checkpoint, mean ± s.e. across seeds. Hard = hard-set pass@1 (specialization); Held = held-out pass@1 (in-domain retention). Higher is better on both, and a method is preferable where it raises Hard without lowering Held. **Bold** = best on each axis within a (student, benchmark) block. Our two variants take the best Hard in 6 of 9 blocks and the best Held in 5 of 9, so the AUPF ordering in Tab. 1 rests on the two axes together rather than on either alone.

Student	Benchmark	Method	Hard ↑	Held ↑
Qwen3-8B	GSM8K	GRPO	0.044 ± 0.012	0.950 ± 0.006
		OPD	0.123 ± 0.013	**0.953** ± 0.007
		Ours (32B)	0.197 ± 0.019	0.943 ± 0.019
		SPOT	0.090 ± 0.005	0.216 ± 0.039
		SODA	0.246 ± 0.021	0.604 ± 0.140
		Ours (bb)	**0.273** ± 0.022	0.943 ± 0.009
	MATH	GRPO	0.000 ± 0.000	0.627 ± 0.003
		OPD	0.002 ± 0.002	0.633 ± 0.017
		Ours (32B)	0.007 ± 0.004	0.700 ± 0.015
		SPOT	0.087 ± 0.008	0.258 ± 0.028
		SODA	**0.136** ± 0.023	0.664 ± 0.002
		Ours (bb)	0.031 ± 0.004	**0.760** ± 0.006
	LiveCodeBench	GRPO	0.000 ± 0.000	0.371 ± 0.000
		OPD	0.093 ± 0.032	0.367 ± 0.018
		Ours (32B)	**0.325** ± 0.014	0.387 ± 0.013
		SPOT	0.217 ± 0.003	**0.413** ± 0.027
		SODA	0.237 ± 0.009	0.313 ± 0.041
		Ours (bb)	0.308 ± 0.008	0.353 ± 0.007
	SciKnowEval	GRPO	0.067 ± 0.024	0.560 ± 0.000
		OPD	0.027 ± 0.004	0.567 ± 0.026
		Ours (32B)	**0.293** ± 0.015	**0.607** ± 0.022
		SPOT	0.164 ± 0.026	0.596 ± 0.004
		SODA	0.227 ± 0.033	0.573 ± 0.037
		Ours (bb)	0.204 ± 0.019	0.497 ± 0.131
	ToolAlpaca	GRPO	0.249 ± 0.154	**0.937** ± 0.009
		OPD	0.078 ± 0.039	0.463 ± 0.274
		Ours (32B)	**0.720** ± 0.010	0.897 ± 0.015
		SPOT	0.691 ± 0.006	0.678 ± 0.073
		SODA	0.667 ± 0.023	0.691 ± 0.067
		Ours (bb)	0.716 ± 0.008	0.847 ± 0.023
OLMo-2-7B	GSM8K	GRPO	0.220 ± 0.070	0.637 ± 0.048
		OPD	**0.496** ± 0.016	0.803 ± 0.015
		Ours (32B)	0.236 ± 0.008	**0.830** ± 0.006
		SPOT	0.115 ± 0.045	0.313 ± 0.153
		SODA	0.153 ± 0.007	0.527 ± 0.020
		Ours (bb)	0.378 ± 0.091	0.774 ± 0.018
	MATH	GRPO	0.020 ± 0.000	0.330 ± 0.000
		OPD	0.013 ± 0.004	0.370 ± 0.025
		Ours (32B)	**0.036** ± 0.008	0.307 ± 0.009
		SPOT	0.009 ± 0.006	0.133 ± 0.020
		SODA	0.011 ± 0.002	0.151 ± 0.008
		Ours (bb)	0.016 ± 0.006	**0.380** ± 0.010
	SciKnowEval	GRPO	**0.658** ± 0.037	0.217 ± 0.047
		OPD	0.113 ± 0.024	0.353 ± 0.033
		Ours (32B)	0.573 ± 0.023	**0.447** ± 0.003
		SPOT	0.224 ± 0.008	0.431 ± 0.019
		SODA	0.089 ± 0.016	0.371 ± 0.002
		Ours (bb)	0.564 ± 0.035	0.424 ± 0.024
	ToolAlpaca	GRPO	0.171 ± 0.171	0.140 ± 0.140
		OPD	0.273 ± 0.015	**0.840** ± 0.015
		Ours (32B)	0.798 ± 0.006	0.707 ± 0.047
		SPOT	0.193 ± 0.112	0.069 ± 0.049
		SODA	0.220 ± 0.070	0.189 ± 0.059
		Ours (bb)	**0.856** ± 0.026	0.749 ± 0.036

Table 7: **RQ1 per-cell backward transfer**, $\mathrm{BWT} = y_T - H_0$ (Lopez-Paz & Ranzato, 2017), mean $\pm$ s.e. across seeds in raw retention points. Higher is better: a positive value means the run ended above the untrained base, a negative one that it ended below, and zero that it returned to base. **Bold** = highest per row. Our variants take the highest value in 5 of 9 cells and SPOT the lowest in 5. Every method ends below base on GSM8K and ToolAlpaca for both students, so those two benchmarks carry the retention cost regardless of how the correction is obtained. GRPO on Qwen-LiveCodeBench is entered as no learning.

		RLVR	same-family 32B teacher		black-box projector		
Student	Benchmark	GRPO	OPD	Ours	SPOT	SODA	Ours
Qwen3-8B	GSM8K	-0.017 ± 0.006	$\mathbf{-0.013} \pm 0.007$	-0.023 ± 0.019	-0.751 ± 0.039	-0.362 ± 0.140	-0.023 ± 0.009
	MATH	$+0.003 \pm 0.003$	$+0.010 \pm 0.017$	$+0.077 \pm 0.015$	-0.366 ± 0.028	$+0.041 \pm 0.002$	$\mathbf{+0.137} \pm 0.006$
	LiveCodeBench	$+0.000 \pm 0.000$	-0.004 ± 0.018	$+0.016 \pm 0.013$	$\mathbf{+0.042} \pm 0.027$	-0.058 ± 0.041	-0.018 ± 0.007
	SciKnowEval	$+0.007 \pm 0.000$	$+0.013 \pm 0.026$	$\mathbf{+0.053} \pm 0.022$	$+0.042 \pm 0.004$	$+0.020 \pm 0.037$	-0.057 ± 0.131
	ToolAlpaca	$\mathbf{-0.013} \pm 0.009$	-0.487 ± 0.274	-0.053 ± 0.015	-0.272 ± 0.073	-0.259 ± 0.067	-0.103 ± 0.023
OLMo-2-7B	GSM8K	-0.232 ± 0.048	-0.066 ± 0.015	$\mathbf{-0.039} \pm 0.006$	-0.556 ± 0.153	-0.342 ± 0.020	-0.094 ± 0.018
	MATH	-0.123 ± 0.049	-0.000 ± 0.025	-0.063 ± 0.009	-0.237 ± 0.020	-0.219 ± 0.008	$\mathbf{+0.010} \pm 0.010$
	SciKnowEval	-0.132 ± 0.047	$+0.005 \pm 0.033$	$\mathbf{+0.098} \pm 0.003$	$+0.083 \pm 0.019$	$+0.023 \pm 0.002$	$+0.076 \pm 0.024$
	ToolAlpaca	-0.704 ± 0.140	$\mathbf{-0.004} \pm 0.015$	-0.137 ± 0.047	-0.775 ± 0.049	-0.655 ± 0.059	-0.095 ± 0.036

Table 8: **RQ1 per-cell forgetting**, $F = \max_{\tau \le T} y_\tau - y_T$ with H_0 included in the max (Chaudhry et al., 2018), mean $\pm$ s.e. across seeds in raw retention points. Smaller is better: F is the retention a run gave up between its best checkpoint and its last, so a small value means retention was held throughout rather than lost and partly regained. **Bold** = smallest per row. SPOT is largest in 5 of 9 cells. The smallest is split evenly between OPD, GRPO and our same-family variant, so F alone does not separate methods and should be read alongside specialization, since a run that moves the student little also has little to give up. GRPO on Qwen-LiveCodeBench is entered as no learning.

		RLVR	same-family 32B teacher		black-box projector		
Student	Benchmark	GRPO	OPD	Ours	SPOT	SODA	Ours
Qwen3-8B	GSM8K	0.020 ± 0.006	$\mathbf{0.016} \pm 0.007$	0.030 ± 0.015	0.751 ± 0.039	0.362 ± 0.140	0.024 ± 0.009
	MATH	$\mathbf{0.001} \pm 0.001$	0.030 ± 0.012	0.027 ± 0.015	0.533 ± 0.030	0.133 ± 0.004	0.010 ± 0.006
	LiveCodeBench	$\mathbf{0.000} \pm 0.000$	0.073 ± 0.029	0.047 ± 0.018	0.040 ± 0.020	0.070 ± 0.038	0.053 ± 0.007
	SciKnowEval	0.040 ± 0.010	0.047 ± 0.026	$\mathbf{0.023} \pm 0.012$	0.084 ± 0.009	0.109 ± 0.046	0.153 ± 0.129
	ToolAlpaca	$\mathbf{0.013} \pm 0.009$	0.490 ± 0.272	0.053 ± 0.015	0.272 ± 0.073	0.259 ± 0.067	0.103 ± 0.023
OLMo-2-7B	GSM8K	0.233 ± 0.048	0.070 ± 0.016	$\mathbf{0.043} \pm 0.007$	0.556 ± 0.153	0.342 ± 0.020	0.095 ± 0.018
	MATH	0.147 ± 0.038	$\mathbf{0.023} \pm 0.015$	0.077 ± 0.009	0.240 ± 0.022	0.219 ± 0.008	0.027 ± 0.015
	SciKnowEval	0.132 ± 0.047	0.063 ± 0.032	$\mathbf{0.013} \pm 0.009$	0.062 ± 0.006	0.060 ± 0.014	0.033 ± 0.018
	ToolAlpaca	0.704 ± 0.140	$\mathbf{0.030} \pm 0.006$	0.139 ± 0.048	0.789 ± 0.048	0.669 ± 0.058	0.098 ± 0.037

E.3 MATH-DOMAIN EVALUATION: IN-DOMAIN AND OUT-OF-DOMAIN BEHAVIOR

Sec. 4.1 measures retention on a held-out slice of the benchmark a method was trained on. Here we ask what training on one domain does to abilities elsewhere. For each math-trained Qwen3-8B method we evaluate its final checkpoint on five full test suites: GSM8K (1,319 problems) and MATH-500 (500) in the trained domain, and AMC23 (40), MMLU (14,042) and HumanEval (164), none of which any method was trained on. The metric is sampled pass@1, one generation per problem at T=0.8 and top-p 0.95. Each evaluation is repeated with three *evaluation* seeds and averaged, with the checkpoint held fixed: an evaluation seed changes only which generations are sampled from that one model. The reported $\pm$ is the standard error of that mean across the three replicates. Training and evaluation data are disjoint, with zero overlap between the 1,991-problem math hard set and the 500-problem evaluation slice on normalized, LaTeX-stripped text. Tab. 9 reports the absolute scores and Tab. 10 the change relative to the untrained student.

Table 9: **Training on math corrections suggests retention is held outside the trained domain.** Final-checkpoint evaluation of the *training-seed*-0 math-trained Qwen3-8B methods, sampled pass@1 at T=0.8 averaged over three evaluation seeds, ± the standard error of that mean, on full test splits. GSM8K and MATH-500 are in the trained domain; AMC23, MMLU and HumanEval were never trained on. Projected distillation is the only method whose score rises on the held-out split of the domain it trained on (0.709 → 0.749 on MATH-500, where SPOT reaches 0.513 and SODA 0.590), and it reaches 0.765 on MMLU while remaining within a few points of base on GSM8K and HumanEval. **Bold** = best and underline = second best per column among the six trained methods; the untrained base is a reference row and is not ranked. GRPO ranks high on several columns because it moves the student least, as Tab. 10 makes plain, so these absolute scores are best read alongside that table. These are single checkpoints on one domain, so the pattern is suggestive rather than conclusive, and AMC23 has only 40 problems, where small differences should not be read as meaningful.

Method	GSM8K	MATH-500	AMC23	MMLU	HumanEval
Base (Qwen3-8B)	0.934 ± 0.003	0.709 ± 0.005	0.433 ± 0.030	0.702 ± 0.002	0.821 ± 0.002
OURS	0.915 ± 0.001	**0.749** ± 0.003	**0.492** ± 0.036	0.765 ± 0.001	0.787 ± 0.000
OURS-32B	0.917 ± 0.001	0.615 ± 0.005	0.400 ± 0.050	0.724 ± 0.001	**0.827** ± 0.005
SODA	0.882 ± 0.004	0.590 ± 0.019	0.342 ± 0.022	**0.766** ± 0.002	0.797 ± 0.005
SPOT	0.916 ± 0.002	0.513 ± 0.011	0.325 ± 0.029	0.750 ± 0.002	0.813 ± 0.014
GRPO	**0.932** ± 0.004	0.700 ± 0.008	0.425 ± 0.029	0.702 ± 0.002	0.821 ± 0.002
OPD	0.927 ± 0.002	0.692 ± 0.005	0.408 ± 0.008	0.677 ± 0.002	0.815 ± 0.005

Table 10: **Change relative to the untrained Qwen3-8B**, from the same evaluation as Tab. 9. The correction-based methods all move up on MMLU (+0.063, +0.064, +0.047) while GRPO is flat and OPD moves down, which suggests that learning from corrections, rather than our variant in particular, is what carries over. In the trained domain only ours moves up, while the mined-once and static-snapshot corrections lose between 0.09 and 0.20 on MATH-500. **Bold** = largest change and underline = second largest per column; GRPO is near zero everywhere, so where it places second it has simply moved the least.

Method	GSM8K	MATH-500	AMC23	MMLU	HumanEval
OURS	−0.019	**+0.041**	**+0.058**	+0.063	−0.035
OURS-32B	−0.017	−0.093	−0.033	+0.022	**+0.006**
SODA	−0.052	−0.119	−0.092	**+0.064**	−0.024
SPOT	−0.018	−0.195	−0.108	+0.047	−0.008
GRPO	**−0.002**	−0.009	−0.008	−0.001	+0.000
OPD	−0.006	−0.017	−0.025	−0.025	−0.006

E.4 DISCUSSION

Sec. 4.1 names three cells that go to a baseline; the full rows are in Tab. 6. On Qwen MATH, SODA reaches 0.136 specialization at 0.664 retention and SPOT 0.087 at 0.258, against our 0.031 at 0.760: both fit the hard set further than we do, and SPOT gives up most of its held-out accuracy doing so. On Qwen SciKnowEval, SODA reaches 0.227 at 0.573 against our 0.204 at 0.497, leading on both axes; our retention there carries a standard error of 0.131 across seeds, against 0.037 for SODA and 0.004 for SPOT, so that cell is the least stable in the panel. On OLMo GSM8K, OPD reaches 0.496 at 0.803 against our 0.236 at 0.830.

The OLMo-2-7B LiveCodeBench cell is omitted from every RQ1 table and figure. No method moves specialization appreciably above zero there, so a frontier anchored at the base is degenerate and the cell contributes an area near zero for every method, our own included.

F RQ2: EDIT SIZE

F.1 SETUP AND MEASUREMENT

Qwen3-8B student, GSM8K hard set (mined pass@k=0, 122 problems), black-box projector, one objective shared by every arm, lr 1e$-$5, β=0.1, 250 steps, 3 seeds. The axis is realized by *edit magnitude*: the minimal edit, then re-prompting the projector for progressively larger rewrites of the same attempt, plus the student-agnostic *gold* anchor. Because the projector rather than the prompt determines how much text it actually replaces, arms are identified by their measured edit size (Tab. 11); the two larger arms realize 3.2$\times$ and 3.9$\times$ the minimal edit. Off-policyness is the log-likelihood difference between the correction and the attempt under the current student π_θ,

$$\Lambda \;=\; \log \pi_\theta(y^+ \mid x) \;-\; \log \pi_\theta(y^- \mid x), \tag{5}$$

and a larger $|\Lambda|$ means the correction sits farther from what the student would itself produce. We summarize Λ by its *median* over training steps rather than its mean: the per-step quantity is heavy-tailed, since a rollout is scored against a policy that has advanced by up to five optimizer steps since it was sampled, and occasional large excursions dominate an average. The median is stable across seeds (Tab. 11) whereas the mean is not. For *gold*, which replaces a long attempt with a short reference solution, Λ moves chiefly through length; Tab. 11 reports its value with that caveat, and it is omitted from the $|\Lambda|$ panel of Fig. 4.

F.2 ABLATION ARMS

Tab. 11 lists the per-arm endpoint accuracies with the off-policyness scalar $|\Lambda|$ and the mean edit-span sizes $|E^+|, |E^-|$ that define the arms. Both spans grow together across the edit arms, so the two are not separable here. *gold* is the exception in kind rather than in degree, inserting the fewest tokens of any arm while deleting the entire attempt.

Table 11: RQ2 edit-size ablation (Qwen3-8B, GSM8K hard set, 3 seeds, mean $\pm$ s.e.). *hard* and *held* are final sampled pass@1; $|\Lambda|$ is the magnitude of the median likelihood shift over training under the current student ((5)). $|E^+|, |E^-|$ are mean edit-span sizes in tokens, and $|E^+| + |E^-|$ is the measured edit size by which the arms are identified. In the three outcome columns **bold** = best and underline = second best, with $|\Lambda|$ read as smaller is better; the two edit-size columns describe the arms and are not ranked. †*gold* replaces a long attempt with a short reference solution, so its $|\Lambda|$ moves chiefly through length and is not on the scale of the rewrites; its standard error is 62% of the value against 5 to 18% for the edit arms.

Arm	$\lvert E^+\rvert+\lvert E^-\rvert$	$\lvert E^+\rvert$ / $\lvert E^-\rvert$	hard	held	$\lvert\Lambda\rvert$
minimal (ours)	276	160 / 116	**0.560**$\pm$0.015	**0.916**$\pm$0.016	**86**$\pm$4
3.2$\times$	879	308 / 572	0.301$\pm$0.065	<u>0.847</u>$\pm$0.067	<u>372</u>$\pm$67
3.9$\times$	1061	359 / 702	0.254$\pm$0.021	0.807$\pm$0.069	433$\pm$35
gold (anchor)	926	85 / 841	<u>0.489</u>$\pm$0.090	0.744$\pm$0.016	3969$\pm$2471†

F.3 MECHANISM

Full-utterance and diff-localized shift. Fig. 11 compares Λ measured over the whole utterance against the same quantity restricted to the changed spans E^-/E^+. The two agree at $r = 0.9999$ across all 2,249 logged steps of the edit arms, with the localized quantity retaining 94 to 95% of the magnitude, because outside the changed spans y^+ and y^- are token-identical and their contributions cancel. Localizing the measurement, as the objective of (1) does, therefore discards no information.

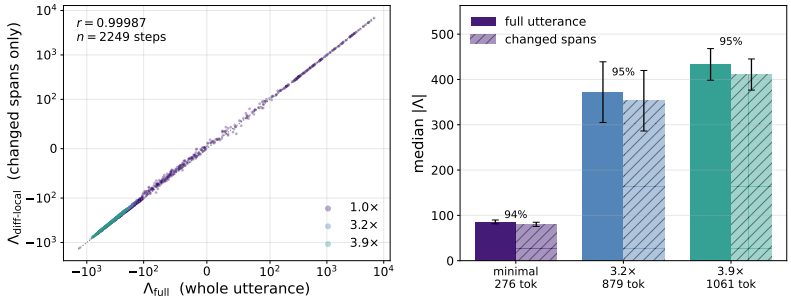


Figure 11: Full-utterance against diff-localized likelihood shift. **Left:** every logged step of the three edit arms, on symmetric log axes with the identity line dotted. **Right:** median $|\Lambda|$ per arm under both measurements, with the localized value as a percentage of the full one.

Edit size and the likelihood shift, step by step. Fig. 12 plots every logged step of the edit arms, with the edit size realized at that step against the likelihood shift it demands. The relation is strongly monotone but not linear. Pooling the arms, the median $|\Lambda|$ rises across every edit-size decile, from 82 in the smallest to 570 in the largest (Spearman $\rho = 0.83$), while the Pearson correlation is only 0.28 because the curve is concave. The effect also diminishes with the size of the edit: the per-step Theil–Sen slope is 0.72 [0.63, 0.82] at the minimal edit against 0.17 [0.16, 0.18] and 0.14 [0.13, 0.15] at the two larger rewrites, and the relation holds within each arm on its own ($\rho = 0.52$, 0.82, 0.78). We therefore read edit size as a monotone indicator of the likelihood shift rather than a proportional one.

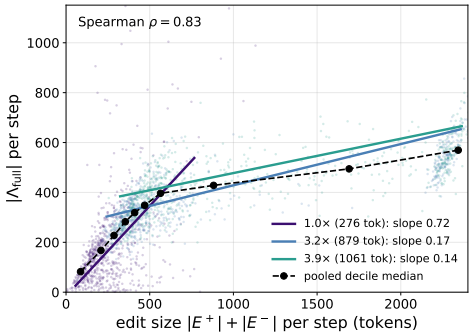


Figure 12: Per-step edit size against the likelihood shift it demands, for the three edit arms (one point per logged step, $n = 2249$). Colored lines are per-arm Theil–Sen fits with their slopes; the dashed black line joins the pooled decile medians. Axes are clipped at the 98th/99.5th percentile, so a few staleness-affected steps fall outside the view.

Insertion and deletion across the arms. Fig. 13 reports the per-step ratio $|E^+|/|E^-|$. It falls monotonically along the axis, from 2.46 at the minimal edit through 1.33 and 0.99 at the larger rewrites to 0.13 for *gold*, so what changes across the arms is not only the amount rewritten but whether the correction predominantly adds to the attempt or replaces it.

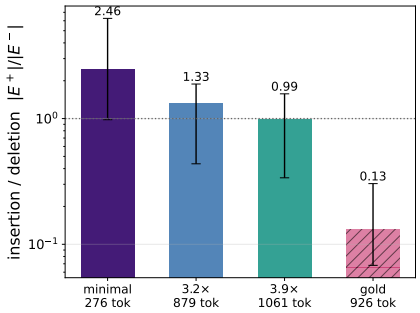


Figure 13: Insertion-to-deletion balance per arm, per-step median with interquartile range on a logarithmic axis. The dotted line marks equal insertion and deletion.

F.4 Additional results

Specialization–retention plane. Fig. 14 plots the two axes against each other. The minimal edit occupies the upper-right region, dominating both larger rewrites on both axes, while *gold* attains intermediate specialization at the lowest retention of any arm.

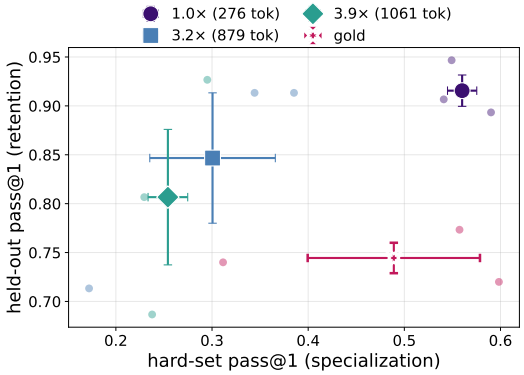


Figure 14: RQ2 specialization–retention plane (3 seeds; faint markers are individual seeds). The minimal edit dominates both larger rewrites on both axes; *gold* is dominated on retention.

Held-out trajectory. Fig. 15 tracks held-out accuracy over training. The edit arms separate in edit-size order and decline smoothly, the minimal edit holding near its base value throughout. *gold* instead destabilizes early, falling to 0.54 by step 75 and remaining volatile for the rest of the run before settling at 0.744.

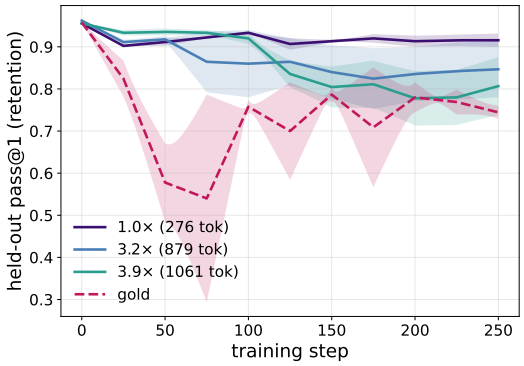


Figure 15: Held-out accuracy over training (mean ± s.e. over 3 seeds). The edit arms decline smoothly and in edit-size order; only *gold* destabilizes.

G RQ4: LOSS-RECIPE DETAILS

G.1 OBJECTIVES

Rollout terminology. Each training step, the student rolls out attempts on the hard problems and the verifier splits them: a *correct rollout* y^{c} is one it solves on its own ($R(x, y^{\mathrm{c}}) = 1$); a *wrong rollout* y^{-} is one it fails ($R = 0$). For each wrong rollout the frozen projector produces the minimal-edit *projection* y^{+} (verified-correct, Sec. 3), and a token-diff aligns (y^{-}, y^{+}) into the mistake span E^{-} and fix span E^{+} used by (1). The *full-sequence* DPO variant replaces E^{+}, E^{-} in (1) with the full sequences y^{+}, y^{-}.

Per-arm objectives. The four contrastive arms optimize $\mathcal{L}^{E}_{\mathrm{DPO}}$ (diff-local, on the changed spans $E^{\pm}$) or $\mathcal{L}^{\mathrm{seq}}_{\mathrm{DPO}}$ (full-sequence, on $y^{\pm}$), each with or without an added self-SFT term $\mathcal{L}_{\mathrm{SFT}}(y^{\mathrm{c}})$ on the student's own correct rollouts. *RFT* optimizes $\mathcal{L}_{\mathrm{SFT}}(y^{\mathrm{c}})$ alone (no projected target, no contrast); *SFT on projection + self-SFT* optimizes $\mathcal{L}_{\mathrm{SFT}}(y^{+}) + \mathcal{L}_{\mathrm{SFT}}(y^{\mathrm{c}})$ (SFT on the projected correction y^{+} plus self-SFT on the student's own correct rollouts, no contrast).

G.2 RESULTS

Compute (batch size), number of epochs and learning rate are matched across arms. Tab. 12 gives the absolute numbers behind Fig. 6.

Table 12: **RQ4 per-arm results** on the GSM8K hard set with Qwen3-8B, mean ± s.e. across 3 seeds. *hard* is hard-set pass@1 and *rel.* the same quantity as a fraction of the adopted recipe, which is also the best arm; *held* is held-out pass@1. **Bold** = best per column. SFT on the projection carries the widest spread on both axes.

Arm	hard ↑	rel. ↑	held ↑
Diff-local DPO + self-SFT	$\mathbf{0.601}$±0.005	$\mathbf{1.000}$	0.938±0.002
Full-sequence DPO + self-SFT	0.552±0.022	0.918	0.911±0.014
SFT on projection + self-SFT	0.484±0.078	0.805	0.877±0.054
Full-sequence DPO	0.470±0.026	0.782	0.902±0.002
Diff-local DPO	0.440±0.015	0.732	0.920±0.014
RFT	0.096±0.011	0.159	$\mathbf{0.944}$±0.002

Three things the table does not show. RFT is limited by how rarely the student succeeds unaided: the hard set is mined at $\mathrm{pass@}k{=}0$, and its rollouts stay correct on about 6% of attempts through the run, so it trains on very little. The instability of SFT on the projection runs across both axes, its standard error of 0.078 on specialization and 0.054 on retention being three to fourteen and four to twenty-seven times those of any other arm. And the self-SFT term is worth $+0.082$ to the full-sequence contrast and $+0.161$ to the diff-local one in absolute pass@1: because the projection edits few tokens, a full-sequence contrast is spread over spans that are already correct while a diff-local one acts only where the edit fell, and that concentration leaves it with less absolute signal of its own, so it gains more from a term that raises the likelihood of correct text outright.

H RQ5: REFINEMENT DETAILS

H.1 REFINEMENT PROTOCOL

A correction is *valid* only if a reasoning check confirms that its derivation process actually yields the answer, so a projection that reaches the right answer by an unsound route is discarded. Round 0 is a single correction pass. In each subsequent round the frozen projector is re-prompted with its own previous correction together with the spans it edited in that correction, and returns a refined correction. Across rounds we keep the valid correction that the student itself is most likely to produce, scored by the base student's log-likelihood on the edited tokens, which selects for corrections the student can absorb rather than for the ones the projector prefers. The projector is never updated. All arms train the Qwen3-8B student for 250 steps with the default recipe of Sec. 3 over 3 seeds.

H.2 RESULTS

Table 13: **RQ5 per-budget results** on the GSM8K hard set with Qwen3-8B, mean ± s.e. across 3 seeds. The first three columns describe the corrections the frozen projector supplies, averaged over training; the last two the student that learns from them, at the final checkpoint. *corr.* is the running correction rate $\sum_t n_t^{\text{corr}} / \sum_t n_t^{\text{wrong}}$ plotted in Fig. 7 (left), and *edit* and median $|\Lambda|$ are the two bars in Fig. 7 (right). **Bold** = best per column. Every column describing the corrections improves monotonically with the budget; downstream, the specialization gain arrives in the first round.

	Corrections			Student	
K	corr. (%)↑	edit $\lvert E^+\rvert+\lvert E^-\rvert$↓	median $\lvert\Lambda\rvert$↓	hard↑	held↑
0	9.2±0.2	256±70	38.1±1.8	0.366±0.023	0.911±0.014
1	15.4±0.7	182±16	34.2±3.0	**0.440**±0.010	0.909±0.010
2	**16.7**±1.1	**130**±24	**26.2**±4.8	**0.440**±0.021	**0.929**±0.006

Projection quality. Refinement improves the corrections on three counts at once, and the three are linked. Re-prompting the projector with the spans it has already edited pushes it to consolidate rather than rewrite, so the correction it returns reaches the right answer more often and departs from the student's own attempt less: the median $|\Lambda|$ falls from 38.1 to 26.2 over the two rounds, placing the refined targets nearer the student in the sense Sec. 4.2 measures, and the edit size falls with it. The K=0 edit size carries a standard error of 70 tokens across seeds, so the level of that row is the least settled number in the table; the ordering across budgets is what the comparison rests on.

Downstream trade-off. Fig. 16 traces the attainable frontier over training for each budget. Refinement extends it: the furthest specialization reached on the frontier rises from 0.393 at K=0 to 0.459 at K=1 and 0.475 at K=2. The K=2 frontier also lies above the other two over the whole sweep, and is the only one that rises above the shared base in retention, reaching 0.960 against the base's 0.955, while K=0 and K=1 only decline from it; the extension in specialization is therefore not paid for out of retention. At the final checkpoint (Tab. 13) the two budgets that refine are level on specialization and K=2 holds retention slightly higher; against standard errors of 0.010–0.021 neither difference is resolved, and what separates the three budgets is the first round.

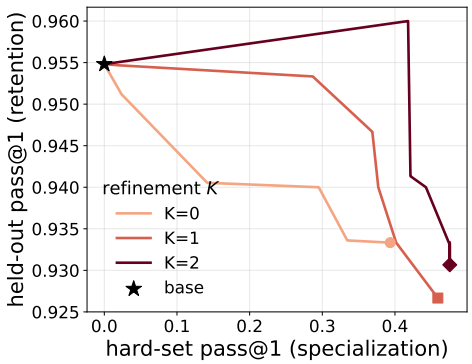


Figure 16: Specialization–retention frontiers over training for $K \in \{0, 1, 2\}$ (3 seeds pooled, frontier over the pooled points), gold-known, with the shared base marked ⋆. More refinement extends the frontier to higher specialization, and the K=2 frontier dominates the other two: it is the only one that rises above the base in retention. Note the narrow retention axis, which spans 0.925–0.960.

Correction rate over training. Fig. 17 shows the *per-step* correction rate for $K \in \{0, 1, 2\}$. For every K the rate rises early and then declines as training proceeds, because the student learns the easily-correctable problems and the wrong attempts that remain in later training are the hardest ones, which even the refined projector rarely corrects. The main-text Fig. 7 reports the running (cumulative) version of this quantity.

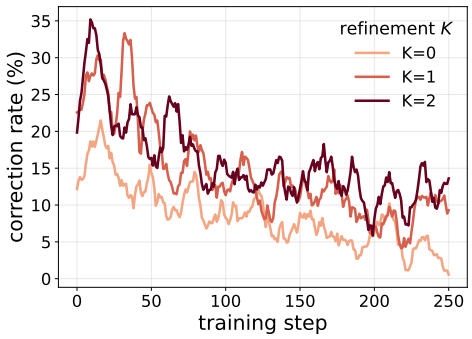


Figure 17: Per-step correction rate over training (pooled over 3 seeds, 11-step moving average), gold-known, $K \in \{0, 1, 2\}$. The rate peaks early and declines as only the hardest problems remain uncorrected; more refinement holds it higher throughout.

I Theoretical analysis

I.1 The Patch MDP: Model and Assumption

This appendix gives the model and the assumption behind Sec. 5; Sec. I.2 gives the proofs.

Definition 2 (Patch MDP)**.** We define a *patch MDP* as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \rho, \gamma)$ over a vocabulary $\mathcal{V}$ with token budget H:

- $\mathcal{S}$: a state is a full history $s = (x, y')$, a problem together with the tokens y' generated so far; $s_t = (x, y_{1:t})$ is the state before decision t, and $|s|$ counts the generated tokens.
- $\mathcal{A}$: an action is a *patch*, a token span fitting the remaining budget, $\mathcal{A}(s) = \{w \in \mathcal{V}^h : 1 \le h \le H - |s|\}$.
- p: transitions are deterministic and append the patch, $p(s \oplus w \mid s, w) = 1$.
- r: the reward is token-additive, $r(s, w) = \sum_{j=1}^{|w|} r_{\text{tok}}(s \oplus w_{<j}, w_j)$ with $r_{\text{tok}} \in [0, 1]$, so a rollout returns at most H.
- ρ: the initial state distribution is the distribution over problems, $s_0 = (x, \varnothing)$ with $x \sim \rho$.
- $\gamma = 1$: no discounting. $\mathcal{V}$ contains a terminal symbol, and a rollout that emits it moves to an absorbing state of zero reward and zero value.

Each decision emits one token, so a rollout takes exactly H of them, indexed $t = 0, \dots, H-1$ with s_H absorbing. Then $d^\pi(s) = \frac{1}{H} \sum_{t=0}^{H-1} \Pr_\pi[s_t = s]$ is a probability distribution over states, the *normalized occupancy* of π.

Write $V^\star$ and $Q^\star$ for the optimal value and action-value of $\mathcal{M}$. The return of a policy and its optimum are

$$J(\pi) \;=\; \mathbb{E}_{s_0 \sim \rho}\, \mathbb{E}_\pi \Big[\sum_{t=0}^{H-1} r_{\text{tok}}(s_t, w_t) \Big], \qquad J^\star \;=\; \max_\pi J(\pi) \;=\; \mathbb{E}_{s_0 \sim \rho} \big[V^\star(s_0) \big], \tag{6}$$

the second equality because an optimal policy is optimal from every state.

Definition 3 (Optimality gap)**.** $\Delta(s, w) := V^\star(s) - Q^\star(s, w)$ is the return an action forfeits against the best one available at s, and $\Delta_{\max} := \sup_{s,w} \Delta(s, w)$, finite because returns lie in $[0, H]$. It is a constant of $\mathcal{M}$, and plays the role of the cost-to-go constant u of Ross et al. (2011).

Definition 4 (Edit count and edit size)**.** For histories s, s',

$$e(s, s') \;:=\; |s| + |s'| - 2\,\big|\mathrm{LCS}(s, s')\big|,$$

where $\mathrm{LCS}(s, s')$ is a longest common subsequence (Hirschberg, 1975) of their generated tokens. This counts the insertions and deletions carrying one into the other, the string-to-string correction

distance of Wagner & Fischer (1974) restricted to those two operations. The *edit size* of a correction is $m := e(y^-, y^+)$, the two rollouts compared at the H decisions Definition 2 gives them, absorbing padding included. It exceeds the token-diff count $|E^-| + |E^+|$ of Sec. 3, which Sec. 4.2 measures, by exactly the change in answer length: $m = |E^-| + |E^+| + \big|\,|y^-| - |y^+|\,\big|$.

The learning problem. Training runs for M rounds. In round i the learner π_i samples rollouts, a frozen projector rewrites the wrong ones, and the learner is scored on those rewrites and updated once. The learner ranges over a policy class $\Pi = \{\pi_\theta : \theta \in \Theta\}$ with $\Theta \subseteq \mathbb{R}^d$ convex of Euclidean diameter D, so minimizing over $\pi \in \Pi$ and over $\theta \in \Theta$ are the same operation; only Proposition 1 uses this.

Where the loss is read. The learner corrects a whole rollout once and scores the correction in a single forward pass, so the loss is evaluated at *corrected* prefixes, not at the states the learner visits. Write $\tilde{d}_i(\tilde{s}) = \frac{1}{H}\sum_{t=0}^{H-1} \Pr[\tilde{s}_t = \tilde{s}]$ for the occupancy of those corrected prefixes in round i, the state-visitation distribution the loss is read under and the counterpart of d^{π_i}. For any distribution d over states, let

$$L(d, \pi) \;:=\; \mathbb{E}_{s\sim d}\, \mathbb{E}_{w\sim\pi(\cdot|s)}\big[\Delta(s, w)\big] \tag{7}$$

be the mean gap of π under d. Performance is controlled by $L(d^{\pi_i}, \pi_i)$, but the algorithm can only reduce $L(\tilde{d}_i, \pi)$.

The projector. The projector edits a wrong attempt into a correct one, and $w^\star(s)$ denotes its token at s. We idealize it as an expert: its token is optimal wherever we read it, so $\Delta(s, w^\star(s)) = 0$. Corollary 2 gives the bound for a projector that is suboptimal.

Definition 5 (Surrogate, realizability, average regret)**.** The surrogate objective $\tilde{\varepsilon}_i(\pi)$ for policy π is the 0–1 disagreement with the projector, measured under $\tilde{d}_i$:

$$\tilde{\varepsilon}_i(\pi) \;:=\; \mathbb{E}_{\tilde{s}\sim\tilde{d}_i}\Big[\Pr_{w\sim\pi(\cdot|\tilde{s})}\big[w \neq w^\star(\tilde{s})\big]\Big] \;=\; \mathbb{E}_{\tilde{s}\sim\tilde{d}_i}\big[\,1 - \pi(w^\star(\tilde{s}) \mid \tilde{s})\,\big].$$

The approximation and optimization terms are

$$\tilde{\varepsilon} \;:=\; \min_{\pi\in\Pi} \frac{1}{M}\sum_{i=1}^{M} \tilde{\varepsilon}_i(\pi), \qquad \tilde{\gamma}_M \;:=\; \frac{1}{M}\Big(\sum_{i=1}^{M} \tilde{\varepsilon}_i(\pi_i) - \min_{\pi\in\Pi}\sum_{i=1}^{M} \tilde{\varepsilon}_i(\pi)\Big),$$

so that $\frac{1}{M}\sum_i \tilde{\varepsilon}_i(\pi_i) = \tilde{\varepsilon} + \tilde{\gamma}_M$ and the learner is *no-regret* when $\tilde{\gamma}_M \to 0$. Here $\tilde{\varepsilon}$ is a minimum over the policy class, so it measures how well the class can reproduce the projector's tokens. It is unaffected by how much return those tokens forfeit; that enters separately, as $\bar{\delta}$ in Corollary 2.

I.1.1 The assumption

The analysis rests on Assumption 1, stated in Sec. 5: the map $s \mapsto \mathbb{E}_{w\sim\pi(\cdot|s)}[\Delta(s, w)]$ is η-Lipschitz in the edit count $e(\cdot, \cdot)$, uniformly over $\pi \in \Pi$. It is the only assumption the analysis makes about the MDP. Three points about it follow.

The Lipschitz constant is uniform over Π and over pairs of histories, but the proof invokes it only at the pairs $(s_t, \tilde{s}_t)$ that Lemma 4 compares, so η may be taken as the largest shift over those pairs.

The condition is placed on the *averaged* gap rather than on $\Delta(s, w)$ at a fixed w. The proof compares $\mathbb{E}_{w\sim\pi}[\Delta]$ at two different prefixes, and moving from one to the other shifts the action distribution as well as the gap, so both must be controlled. Two pointwise conditions suffice. Suppose the gap is itself Lipschitz, $|\Delta(s', w) - \Delta(s, w)| \le \eta_\Delta\, e(s, s')$, and the policy is smooth in the same count, $\mathrm{TV}\big(\pi(\cdot \mid s), \pi(\cdot \mid s')\big) \le \kappa\, e(s, s')$. Adding and subtracting $\mathbb{E}_{w\sim\pi(\cdot|s')}[\Delta(s, w)]$ splits the difference in two, bounded by $\eta_\Delta e$ and by $2\Delta_{\max}\kappa e$. Together they give Assumption 1 with $\eta \le \eta_\Delta + 2\Delta_{\max}\kappa$.

Assumption 1 is also the only reason any result below depends on the size of an edit. An argument that knows only *which* decisions sit downstream of an edit pays $\Delta_{\max}$ at each of them, which is identical for a one-token edit and for a hundred-token edit at the same position: "the prefix differs" is a binary predicate and it saturates. Assumption 1 instead charges in proportion to $e(\cdot, \cdot)$, which every edit-size claim below uses.

I.2 PROOFS

I.2.1 THE GAP DECOMPOSES ACROSS A PATCH

Lemma 1 (Patch additivity). *For patch MDP,* $\Delta(s, w_{1:h}) = \sum_{j=1}^{h} \Delta(s \oplus w_{<j}, w_j)$.

Proof. Since the transitions of Patch MDP are deterministic (c.f. Definition 2), $Q^\star(s, w_{1:h}) = r(s, w_{1:h}) + V^\star(s \oplus w_{1:h})$, and by Definition 2 the reward splits as $r(s, w_{1:h}) = \sum_{j=1}^{h} r_{\text{tok}}(s \oplus w_{<j}, w_j)$. Hence

$$\begin{aligned}
\Delta(s, w_{1:h}) &= V^\star(s) - r(s, w_{1:h}) - V^\star(s \oplus w_{1:h}) && \text{(by determinism)} \\
&= V^\star(s) - V^\star(s \oplus w_{1:h}) - \sum_{j=1}^{h} r_{\text{tok}}(s \oplus w_{<j}, w_j) && \text{(by Definition 2)} \\
&= \sum_{j=1}^{h} \big[V^\star(s \oplus w_{<j}) - V^\star(s \oplus w_{<j+1})\big] - \sum_{j=1}^{h} r_{\text{tok}}(s \oplus w_{<j}, w_j) && \text{(telescoping)} \\
&= \sum_{j=1}^{h} \big[V^\star(s \oplus w_{<j}) - r_{\text{tok}}(s \oplus w_{<j}, w_j) - V^\star(s \oplus w_{<j+1})\big] \\
&= \sum_{j=1}^{h} \Delta(s \oplus w_{<j}, w_j) && \text{(by determinism)}
\end{aligned}$$

where the last step uses $Q^\star(s \oplus w_{<j}, w_j) = r_{\text{tok}}(s \oplus w_{<j}, w_j) + V^\star(s \oplus w_{<j+1})$. □

The total gap along a rollout is therefore the same however the rollout is cut into patches; only the per-step decomposition changes.

I.2.2 REGRET IS THE ACCUMULATED GAP, UNDER THE LEARNER'S OWN OCCUPANCY

Lemma 2 (No distribution shift). *Under Definition 2, for every round i,*

$$J^\star - J(\pi_i) \;=\; \mathbb{E}_{\pi_i}\Big[\sum_{t=0}^{H-1} \Delta(s_t, w_t)\Big] \;=\; H\, L(d^{\pi_i}, \pi_i),$$

where $\mathbb{E}_{\pi_i}$ is over trajectories $(s_0, w_0, \ldots, s_{H-1}, w_{H-1})$ generated by π_i from $s_0 \sim \rho$, and L is as in (7).

Proof. Fix one trajectory $(s_0, w_0, \ldots, s_{H-1}, w_{H-1})$ generated by π_i. By Definition 2 it has exactly H decisions and $V^\star(s_H) = 0$, since s_H is the absorbing state with zero value. Then

$$\begin{aligned}
V^\star(s_0) - \sum_{t=0}^{H-1} r_{\text{tok}}(s_t, w_t) &= \sum_{t=0}^{H-1} \big[V^\star(s_t) - r_{\text{tok}}(s_t, w_t) - V^\star(s_{t+1})\big] \\
&\qquad \text{(telescoping, } V^\star(s_H) = 0) \\
&= \sum_{t=0}^{H-1} \Delta(s_t, w_t) \qquad (Q^\star(s_t, w_t) = r_{\text{tok}}(s_t, w_t) + V^\star(s_{t+1}))
\end{aligned}$$

Taking $\mathbb{E}_{\pi_i}$ over trajectories turns the left side into $\mathbb{E}[V^\star(s_0)] - \mathbb{E}\big[\sum_t r_{\text{tok}}(s_t, w_t)\big] = J^\star - J(\pi_i)$, which is the stated identity. For the second equality,

$$\begin{aligned}
\mathbb{E}_{\pi_i}\Big[\sum_{t=0}^{H-1} \Delta(s_t, w_t)\Big] &= \sum_{t=0}^{H-1} \mathbb{E}\big[\Delta(s_t, w_t)\big] && \text{(exactly } H \text{ terms, by Definition 2)} \\
&= H\, \mathbb{E}_{s \sim d^{\pi_i}} \mathbb{E}_{w \sim \pi_i}\big[\Delta(s, w)\big] && \text{(by the definition of } d^{\pi_i}) \\
&= H\, L(d^{\pi_i}, \pi_i)
\end{aligned}$$

□

Both expectations in $L(d^{\pi_i}, \pi_i)$ are taken under π_i's own occupancy. No change of measure is performed, so no concentrability coefficient appears.

I.2.3 Disagreement with the projector dominates the gap

Lemma 3 (Reduction). *Under Definition 3, with the projector idealized as above, for every $\pi \in \Pi$ and every round i,*

$$L(\tilde{d}_i, \pi) \;\le\; \Delta_{\max}\, \tilde{\varepsilon}_i(\pi).$$

Proof. Fix a corrected state $\tilde{s}$ and split the expectation over the projector's token and the rest:

$$\begin{aligned}
\mathbb{E}_{w\sim\pi}\big[\Delta(\tilde{s}, w)\big] &= \pi(w^\star \mid \tilde{s})\,\Delta(\tilde{s}, w^\star) + \sum_{w\neq w^\star} \pi(w \mid \tilde{s})\,\Delta(\tilde{s}, w) \\
&= \sum_{w\neq w^\star} \pi(w \mid \tilde{s})\,\Delta(\tilde{s}, w) && (\Delta(\tilde{s}, w^\star) = 0, \text{ idealized projector}) \\
&\le \Delta_{\max} \sum_{w\neq w^\star} \pi(w \mid \tilde{s}) && (\text{by Definition 3}) \\
&= \Delta_{\max}\big(1 - \pi(w^\star \mid \tilde{s})\big)
\end{aligned}$$

Taking $\mathbb{E}_{\tilde{s}\sim\tilde{d}_i}$ of both sides and applying Definition 5 gives the claim. □

This is the reduction of Ross et al. (2011), applied at the corrected prefixes rather than at the learner's own states.

I.2.4 Transfer from the corrected prefixes to the learner's own

Lemma 4 (Transfer). *Under Assumption 1 and Definition 2, for every round i,*

$$L(d^{\pi_i}, \pi_i) \;\le\; L(\tilde{d}_i, \pi_i) + \eta\,\bar{e}_i, \;\; \textit{where } \bar{e}_i \;\coloneqq\; \mathbb{E}_i\Big[\tfrac{1}{H}\sum_{t=0}^{H-1} e_t\Big]$$

and $e_t \coloneqq e(s_t, \tilde{s}_t)$ is the edit count between the learner's own prefix at decision t and the corrected prefix at that decision. A round draws one problem, one rollout of π_i on it, and the one correction the projector returns for that rollout; $\mathbb{E}_i$ is the average over that draw.

Proof. Pair the two distributions decision by decision: the t-th term pairs the learner's own prefix s_t with the corrected prefix $\tilde{s}_t$, which differ by e_t tokens (Definition 4). By Definition 2 both sides of (7) are averages of exactly H such terms, so

$$\begin{aligned}
L(d^{\pi_i}, \pi_i) &= \tfrac{1}{H}\sum_{t=0}^{H-1} \mathbb{E}_i\, \mathbb{E}_{w\sim\pi_i(\cdot\mid s_t)}\big[\Delta(s_t, w)\big] && (\text{by Definition 2}) \\
&\le \tfrac{1}{H}\sum_{t=0}^{H-1} \mathbb{E}_i\Big[\mathbb{E}_{w\sim\pi_i(\cdot\mid \tilde{s}_t)}\big[\Delta(\tilde{s}_t, w)\big] + \eta\, e_t\Big] && (\text{by Assumption 1}) \\
&= L(\tilde{d}_i, \pi_i) + \eta\,\mathbb{E}_i\Big[\tfrac{1}{H}\sum_{t=0}^{H-1} e_t\Big] \\
&= L(\tilde{d}_i, \pi_i) + \eta\,\bar{e}_i
\end{aligned}$$

□

I.2.5 How much edited text a rollout carries

Lemma 5 (Edit budget). *Let y be the learner's rollout in round i and $\tilde{y}$ its correction, each of H decisions by Definition 2, and let $m = e(y, \tilde{y})$ be the edit size, with e_t as in Lemma 4. Then $e_t \le m$*

at every decision $0 \le t < H$*, and consequently*

$$\sum_{t=0}^{H-1} e_t \;\le\; m\,H.$$

Proof. Let u index the tokens of y and v those of $\tilde{y}$. A *monotone matching* between them is a set Φ of pairs (u, v) with $y_u = \tilde{y}_v$, whose order agrees on both sides: $u < u' \iff v < v'$ for distinct $(u, v), (u', v') \in \Phi$. The matched tokens read in order form a common subsequence of y and $\tilde{y}$, so a largest matching has $|\Phi| = |\mathrm{LCS}(y, \tilde{y})|$. Take such a Φ. Since y and $\tilde{y}$ each have H tokens, $m = 2H - 2|\Phi|$ by Definition 4. For a decision t put $\Phi_t = \{(u, v) \in \Phi : u \le t,\ v \le t\}$, the pairs falling inside both prefixes, whose matched tokens form a common subsequence of $y_{1:t}$ and $\tilde{y}_{1:t}$. Both coordinates increase along Φ, so $\max(u, v)$ does too, strictly; every pair outside Φ_t has $\max(u, v) \ge t + 1$, and distinct pairs take distinct values there, so

$$\left|\Phi \setminus \Phi_t\right| \;\le\; H - t. \tag{8}$$

Therefore

$$\begin{aligned}
e_t &= 2t - 2\left|\mathrm{LCS}(y_{1:t}, \tilde{y}_{1:t})\right| && \text{(by Definition 4)} \\
&\le 2t - 2\left|\Phi_t\right| && (\Phi_t \text{ is a matching}) \\
&= 2t - 2\left|\Phi\right| + 2\left|\Phi \setminus \Phi_t\right| \\
&\le 2t - 2\left|\Phi\right| + 2\,(H - t) && \text{(by (8))} \\
&= 2H - 2\left|\Phi\right| \;=\; m && \text{(by Definition 4)}
\end{aligned}$$

Summing over the H decisions gives the claim. □

An insertion early in the answer shifts every later prefix, and a count of mismatched positions would read that shift as a rewrite of the whole suffix. Definition 4 charges it at the size of the insertion instead, which is why the drift of Theorem 1 scales with m and not with H.

I.2.6 The bound

Theorem 1 of Sec. 5 states that under Definitions 2 and 3, Assumption 1, an idealized projector and m the largest edit size over the rounds,

$$\min_{i \le M}\left[J^\star - J(\pi_i)\right] \;\le\; \underbrace{H\,\Delta_{\max}\left(\tilde{\varepsilon} + \tilde{\gamma}_M\right)}_{\text{fit and optimization}} + \underbrace{\eta\, m\, H}_{\text{corrected-prefix drift}}.$$

Proof of Theorem 1.

$$\begin{aligned}
\min_{i \le M}[J^\star - J(\pi_i)] &\le \tfrac{1}{M}\sum_{i=1}^{M}[J^\star - J(\pi_i)] \\
&\le \tfrac{H}{M}\sum_{i} L(d^{\pi_i}, \pi_i) && \text{(by Lemma 2)} \\
&\le \tfrac{H}{M}\sum_{i} L(\tilde{d}_i, \pi_i) + \eta H \bar{e} && \text{(by Lemma 4)} \\
&\le H\Delta_{\max} \cdot \tfrac{1}{M}\sum_{i} \tilde{\varepsilon}_i(\pi_i) + \eta H \bar{e} && \text{(by Lemma 3)} \\
&\le H\Delta_{\max}(\tilde{\varepsilon} + \tilde{\gamma}_M) + \eta H \bar{e} && \text{(by Definition 5)} \\
&\le H\Delta_{\max}(\tilde{\varepsilon} + \tilde{\gamma}_M) + \eta\, m\, H && \text{(by Lemma 5)}
\end{aligned}$$

where $\bar{e} = \frac{1}{M}\sum_i \bar{e}_i$, and the last step holds because $H\bar{e}_i = \mathbb{E}_i\big[\sum_{t=0}^{H-1} e_t\big] \le m\,H$ in every round. □

I.2.7 Rate in the number of rounds

Proposition 1 (Convex reference rate). *Let Π and Θ be as in Sec. I.1, and suppose in addition that $\theta \mapsto \tilde{\varepsilon}_i(\pi_\theta)$ is convex on Θ for every i, with gradient bound $G = \max_i \sup_{\theta\in\Theta} \|\nabla_\theta \tilde{\varepsilon}_i(\pi_\theta)\|$. Then online gradient descent with step size $\alpha = D/(G\sqrt{M})$ attains $\tilde{\gamma}_M \le DG/\sqrt{M}$.*

Proof. Since Π is parameterized by Θ, the minimum over $\pi \in \Pi$ in Definition 5 is a minimum over $\theta \in \Theta$. Write $\theta^\star \in \arg\min_{\theta\in\Theta} \sum_i \tilde{\varepsilon}_i(\pi_\theta)$ for a minimiser, so that the comparator in the definition and the comparator in the analysis are the same point. Then, with $g_i = \nabla_\theta \tilde{\varepsilon}_i(\pi_{\theta_i})$,

$$\begin{aligned} M\tilde{\gamma}_M &= \sum_i \tilde{\varepsilon}_i(\pi_{\theta_i}) - \sum_i \tilde{\varepsilon}_i(\pi_{\theta^\star}) && \text{(by Definition 5)} \\ &\le \sum_i g_i^\top(\theta_i - \theta^\star) && \text{(convexity)} \\ &\le \tfrac{D^2}{2\alpha} + \tfrac{\alpha}{2} M G^2 && \text{(potential } \tfrac{1}{2\alpha}\|\theta_i - \theta^\star\|^2 \text{ telescopes)} \\ &\le DG\sqrt{M} && \text{(at } \alpha = D/(G\sqrt{M})\text{)} \end{aligned}$$

where D is the Euclidean diameter of Θ and G the gradient bound. Dividing by M gives $\tilde{\gamma}_M \le DG/\sqrt{M}$. □

Remark 1 (Scope of the rate). Proposition 1 is the standard online-convex-optimization guarantee, stated to show *how* regret decays in the number of rounds. Its convexity hypothesis does not hold for a neural policy. Theorem 1 uses only $\tilde{\gamma}_M$ as defined, so nothing below depends on Proposition 1; the $O(1/\sqrt{M})$ form should be read as a convex reference point.

I.2.8 Reading the bound

Corollary 1 (Edit size affects the horizon dependence). *Under the hypotheses of Theorem 1, the drift term has two extremes:*

(a) minimal edits, $m = O(1)$, give drift $O(\eta H)$;

(b) a correction rewriting the whole answer, $m = \Theta(H)$, gives drift $O(\eta H^2)$.

The quantity being bounded is itself of order H, since the return lies in $[0, H]$ by Definition 2. Read against that scale, case (a) puts the drift at the same order as the fit term and the bound is informative, while case (b) puts it an order above the trivial bound $J^\star - J \le H$ and the bound says nothing. The horizon dependence is therefore linear at one end and quadratic at the other, and what selects between them is the edit we choose rather than any property of the environment.

I.2.9 Scope of the analysis

Remark 2 (A statement about horizon dependence, not about algorithm identity). Setting $e_t \equiv 0$, which corresponds to a projector queried at every mistake on the learner's own prefix so that no edited token enters a conditioning prefix, removes the drift and leaves $H\Delta_{\max}(\tilde{\varepsilon} + \tilde{\gamma}_M)$, the shape of the DAgger guarantee (Ross et al., 2011). The linear–quadratic dichotomy of Corollary 1 is the same dichotomy that separates DAgger from behavior cloning. It does not follow that projected distillation *is* either algorithm: minimal-edit projection does not query a projector at every visited state and carries a drift term DAgger does not have, and a full rewrite is not behavior cloning for the reason in Remark 3. The correspondence is between rates.

Remark 3 (The drift does not vanish under a perfect fit). In Ross et al. (2011) the quadratic term for behavior cloning scales with the fit error, so a learner that imitates perfectly suffers no compounding. The drift $\eta m H$ of Theorem 1 carries no factor of $\tilde{\varepsilon}$. A learner that matches the projector exactly at every corrected prefix still pays it, because the mismatch between $\tilde{d}_i$ and d^{π_i} is exogenous, fixed by the edit that was chosen, rather than a consequence of imperfect fitting.

Remark 4 (The analyzed loss is not the deployed loss). Theorem 1 is proved for the 0–1 disagreement of Definition 5. Three consequences.

First, no gradient method minimizes a 0–1 loss. Following Ross et al. (2011) we state the guarantee for the 0–1 loss and note that any convex upper bound may be substituted, with $\tilde{\varepsilon}$ and $\tilde{\gamma}_M$ read as the floor and regret of that surrogate; the bound holds verbatim with a larger $\tilde{\varepsilon}$.

Second, cross-entropy on the correction is such a bound. Since $\log p \le p - 1$ for $p \in (0, 1]$, we have $1 - \pi(w^\star \mid \tilde{s}) \le -\log \pi(w^\star \mid \tilde{s})$, so Theorem 1 covers maximum-likelihood fine-tuning on the projection.

Third, the deployed objective is not covered. The recipe minimizes a diff-localized contrast on the changed spans together with a self-SFT term on the learner's own correct rollouts. A preference contrast is not an upper bound on the disagreement with the projector: it can be reduced by lowering the likelihood of y^- without raising that of y^+, so a small contrast loss does not imply a small $\tilde{\varepsilon}$. The analysis characterizes the *target*: how far the correction lies from the attempt, the quantity the edit-size experiment varies and the one both objectives share. It does not characterize the contrast.

Corollary 2 (A projector that is not optimal)**.** *Suppose the projector's token forfeits* $\delta(\tilde{s}) := \Delta(\tilde{s}, w^\star(\tilde{s})) \ge 0$ *rather than nothing, and let* $\bar{\delta}$ *be its average over the corrected prefixes. Then, under the remaining hypotheses of Theorem 1,*

$$\min_{i \le M} \left[J^\star - J(\pi_i) \right] \;\le\; H\bar{\delta} \;+\; H\,\Delta_{\max}\,(\tilde{\varepsilon} + \tilde{\gamma}_M) \;+\; \eta\, m\, H.$$

Proof. The term $\pi(w^\star \mid \tilde{s})\,\Delta(\tilde{s}, w^\star)$ dropped in Lemma 3 no longer vanishes, so that lemma reads $L(\tilde{d}_i, \pi) \le \bar{\delta} + \Delta_{\max}\tilde{\varepsilon}_i(\pi)$. Substituting it at the Lemma 3 step of Theorem 1 adds $H\bar{\delta}$, every term there already carrying the factor H from Lemma 2, and leaves the rest of the chain unchanged. □

The two error sources stay separate: $\bar{\delta}$ measures the projector and shrinks only with a better one, while $\tilde{\varepsilon}$ measures the policy class and shrinks with a larger one.