CAN NEWER MODELS' GAINS BE BACKPORTED TO OLDER LANGUAGE MODELS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Organizations keep a deployed language model for its latency, license, or validated behavior long after a successor release solves tasks it cannot. Distillation imitates a teacher wholesale, and model evolution migrates assets forward. Neither moves one verified gain backward into a model that must stay in service. A newer release fixes some of its predecessor's mistakes but makes new ones of its own. A backport should inherit the fixes, verified task by task, and none of the new mistakes. **CBack** tests whether that is possible, first refusing version gaps indistinguishable from zero. Each verifier-rejected predecessor trajectory is then paired with a verifier-accepted repair, a *version event*. Preservation is normalized by the predecessor's own resampling churn, and cost separates successor queries from a model's own tokens. Of eight audited scopes in two model lineages, four carry a resolvable gain. All four recover, only one under successor guidance, and none needs successor access. Failure-gated querying instead yields $3.2\times$ more admissible repairs per successor token than uniform querying. On GSM8K, two successor-free arms learn only from the predecessor's verified samples, and only the one that also trains against its own failures recovers on a sealed test split. We conclude that failure-aware contrast improves recovery and the repair's source decides how far. With its own repairs and no successor query, the deployed 1.5B model recovers 0.575 of that held-out gap and flips fewer of its successes than resampling does.

1 INTRODUCTION

Model releases create an awkward choice for deployed language-model systems. A successor may solve tasks a deployment cannot. Yet swapping it in can violate contracts on latency, hardware, licensing, or behavior (Xie et al., 2021; Träuble et al., 2021). Predecessors therefore stay in service with their capability set frozen. Software engineering names that middle ground: a **backport** moves one needed change into an older release without replacing it (Shariffdeen et al., 2021). Whether such an operation is available for a language model has not been measured, and neither has its price.

Every established route for moving capability between models runs the other way. Distillation asks a student to reproduce a teacher (Hinton et al., 2015; Kim & Rush, 2016), so it spends supervision where both versions already agree and copies a successor where it regressed. Model evolution migrates assets into a newer release, by training a compatible successor or porting an adapter forward (Echterhoff et al., 2024; Gu et al., 2025). Both take a newer model as destination. Neither asks what a deployment that cannot move should do with a release it cannot adopt. Measurement is missing too. A benchmark delta between two releases conflates real gain, sampling noise, and reversal. Recovering "80% of an upgrade" therefore means nothing until that upgrade is shown to exist. A newer release fixes some of its predecessor's mistakes and makes new ones of its own (Figure 1a). A backport should inherit the fixes and none of the new mistakes. Each fix must exist, be reachable inside the predecessor's deployment envelope, and leave predecessor successes intact.

We introduce **CBack**[1], an operational test of *capability backporting* that establishes all three before it compares routes. CBack is built on *version events*, smallest units of a version difference an executable verifier can certify. Each event pairs a predecessor trajectory that the verifier rejected with a response it accepted (Figure 1b, ACQUIRE). CBack then asks whether each gain is resolvable,

[1]Code: `https://anonymous.4open.science/r/CBack-C2BC`.

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.

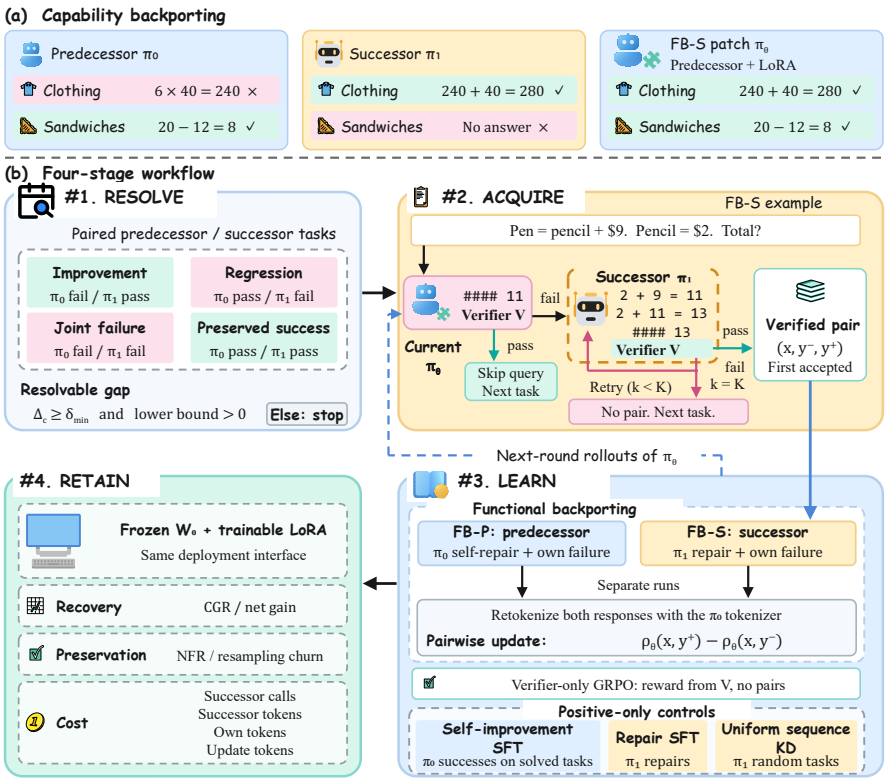

Figure 1: Capability backporting and its measurement. (a) A patch installs a successor gain while retaining a predecessor success its successor lost, on examples from Appendix O. (b) Each scope passes through RESOLVE, ACQUIRE, LEARN and RETAIN. Functional backporting trains on verified failure–repair pairs, and controls change where an accepted response comes from or drop the pairs.

whether its repairs are acquirable at affordable query cost, and whether a predecessor absorbs them (Figure 1b). Three measurements follow. A resolvable-gap gate refuses scopes whose version delta is indistinguishable from zero or reversed, so no training compute is spent on them. Preservation is normalized by a predecessor's own resampling churn, which keeps it defined under stochastic decoding. Cost stays in separate coordinates, so successor queries are never netted against a model's own tokens. A verifier rather than a teacher decides which side of a pair is positive, so one update can take its positive from a successor or from the predecessor itself.

Our contributions are as follows. **An operational test of whether a verified version gain is backportable.** CBack makes transferability testable rather than assumed, and evaluates successor-guided and successor-free routes under one protocol. **A finding that holds across scopes and lineages.** Of eight audited scopes in two lineages, four carry a resolvable gain. All four recover, only one under successor guidance. None needed successor access to recover, and successor access instead bought acquisition efficiency. **An isolation of the mechanism on GSM8K.** Two successor-free arms learn only from the predecessor's verified samples, and only one trains against the model's own failures. On a sealed test split they land at best and worst recovery. With successor repairs held fixed, the same contrast lifts FB-S above repair SFT, so failure-aware contrast improves recovery there. A 1.5B deployment recovers 0.575 of that held-out gap with no successor query.

2 Related Work

Learning from a stronger model. Knowledge distillation matches a compact model to teacher predictions or to teacher-generated sequences (Hinton et al., 2015; Kim & Rush, 2016). Teacher-only data leave an autoregressive learner untrained on its own errors (Ross et al., 2011). Later methods

therefore mix teacher and student rollouts (Agarwal et al., 2024) or choose divergences that limit mode averaging (Gu et al., 2024; Ko et al., 2024). A larger teacher can even yield a poorer student (Zhong et al., 2024), and errors in teacher states can pass to a student unless it learns a residual beyond them (On et al., 2026). DistiLLM-2 raises the likelihood of teacher responses and lowers that of student responses on one prompt (Ko et al., 2025). Among published objectives, DistiLLM-2 comes closest to a contrast between a repair and a failure. Selective variants transfer a teacher's post-training delta (Heo et al., 2026) or map its distribution across tokenizers (Niu et al., 2026; Wang et al., 2026). Preference optimization instead ranks a chosen response over a rejected one relative to a reference policy (Rafailov et al., 2023), with reference-free and length-normalized forms (Meng et al., 2024). Self-training keeps a model's own verified successes (Zelikman et al., 2022; Yuan et al., 2023; Singh et al., 2024), and mistake-driven fine-tuning adds corrections of its failures (An et al., 2023). Reinforcement learning from verifiable rewards uses that same executable signal with no teacher (Lambert et al., 2024; Shao et al., 2024; DeepSeek-AI, 2025). Distillation assumes a teacher worth imitating throughout, or logit access to one, and no route above certifies a version difference by outcome. Selective variants define their delta by teacher parameters, and pairwise methods compare two candidates for one prompt. We ask which ingredient matters when a teacher is a black-box successor with its own regressions, and whether a teacher is needed once a verifier is available.

Model evolution, compatibility, and targeted updates. Continual learning studies adaptation without forgetting (Kirkpatrick et al., 2017), and model editing narrows its intervention to selected facts and measures locality (Meng et al., 2022; Mitchell et al., 2022). Regression-free update methods measure negative flips between model versions and train a newer model to avoid them (Yan et al., 2021; Xie et al., 2021; Träuble et al., 2021). Work on model evolution mostly moves assets forward. One line trains a successor to stay representationally compatible with its predecessor (Echterhoff et al., 2024). Another maps an old adapter into a newer backbone (Gu et al., 2025) or ports a weight delta between compatible parameter spaces (Lin et al., 2025). Training-free routes add a stored delta or task patch onto a newer release (Cao et al., 2025; Khan et al., 2025). Each needs the weights of both releases and moves whatever a parameter difference holds. MT-PATCHER avoids teaching a machine-translation student what it already knows (Li et al., 2024). We study the reverse: one gain of a newer release installed into an older model under black-box access, judged by a verifier.

3 Problem Setting

Let π_0 denote a deployed legacy policy and π_1 a released successor. A trainable patch with parameters θ produces π_θ while leaving its deployment interface unchanged. A task $x \sim \mathcal{D}_c$ belongs to a capability slice c fixed before any training. Rollout y receives an executable score $V_x(y) \in \{0, 1\}$. Stochastic policies are evaluated with a fixed decoding protocol and K_{eval} samples. We define the capability score as

$$J_c(\pi) = \mathbb{E}_{x\sim\mathcal{D}_c,\, y\sim\pi(\cdot|x)}[V_x(y)]. \tag{1}$$

All version comparisons use paired task identities and common random-number seeds where possible.

Functional version events. Paired rollouts $y_0 \sim \pi_0$ and $y_1 \sim \pi_1$ fall into four exhaustive events by their verifier outcomes: improvement I_x (0→1), preserved success P_x (1→1), successor regression R_x (1→0), and joint failure F_x (0→0). Only I_x supplies a backport target. P_x supplies preservation evidence. R_x is recorded but never used as a positive training label, and F_x offers no repair.

Definition 1 (Capability-gap recovery)**.** *For a slice with $\Delta_c = J_c(\pi_1) - J_c(\pi_0) > 0$, the recovered fraction is*

$$\mathrm{CGR}_c(\theta) = \frac{J_c(\pi_\theta) - J_c(\pi_0)}{\Delta_c}. \tag{2}$$

CGR is reported only when a paired interval shows $\Delta_c \geq \delta_{\min}$, our preset margin, and is not clipped. A value above one means a patch outscores its successor there, as Section 4's union argument allows.

Preservation is measured conditionally on legacy success:

$$\mathrm{NFR}_c(\theta) = \Pr[V_x(y_\theta) = 0 \mid V_x(y_0) = 1], \tag{3}$$

where NFR denotes the negative-flip rate (Yan et al., 2021). Cost is recorded as a vector $C(\theta) = (Q, T, G, U)$ of successor queries, successor tokens, self-generated tokens and update tokens. No exchange rate between them is assumed.

Under stochastic decoding, however, a negative-flip rate has a non-zero null. Let p denote per-task success probability of π_0 under that decoding protocol. Pairing two independent samples of π_0 with itself gives the *churn floor* $\mathrm{NFR}_0 = \mathbb{E}[p(1-p)]/\mathbb{E}[p]$ (Fard et al., 2016), a value any behaviourally identical patch attains. Preservation is reported as ratio $\mathrm{NFR}_c/\mathrm{NFR}_0$, which equals one for an unchanged model. Our preservation tolerance is set on that scale rather than an absolute one.

4 BACKPORTING A VERIFIED VERSION GAIN

To move one verified gain backward without disturbing what a deployment already does, we build CBack around four stages that follow our data (Figure 1b). A version gap is first confirmed resolvable, and failure–repair pairs are then acquired. A patch trains on those pairs and is judged on recovery, retained successes, and cost. Each stage below states which difficulty it answers and which comparison tests it. We contribute functional backporting, the pairwise update of Section 4.3. Functional backporting draws its repair from a successor (**FB-S**) or from the predecessor's own samples (**FB-P**).

4.1 CONFIRMING A RESOLVABLE GAP

A recovery ratio divides by its version gap, so a gap indistinguishable from zero makes recovery meaningless and a reversed gap makes it misleading. Before training, a scope must show a successor advantage of at least $\delta_{\min} = 0.03$ on development, with a paired-bootstrap lower bound above zero. We fixed $\delta_{\min}$ before training. Gating excludes four of eight audited scopes (Section 6).

4.2 ACQUIRING VERIFIED FAILURE–REPAIR PAIRS

Training needs triples (x, y^-, y^+) in which y^- is a current-policy failure and y^+ a repair an executable verifier accepts. Pairs collect into a buffer $\mathcal{D}_\Delta$. A pairwise arm trains once the buffer is large enough, and our confirmatory runs require 500 pairs on GSM8K and MATH and 50 on MBPP. Two choices shape that buffer, labelling and query policy. First, both sides pass through our executed verifier, so a label records task outcome and never successor likelihood. Successor wording is not treated as uniquely correct, and a successor regression cannot enter as a positive. Second, we query a successor only where our current policy has just failed. Where both versions agree, a successor adds nothing version-specific. Where a predecessor fails, verified successor success is exactly one version event of Section 3. Figure 1b traces one recorded pair. Our predecessor answers a GSM8K training item with a pen's price alone, which our verifier rejects. A successor's first sample adds that pencil and enters the buffer. Tasks come from public training splits (Appendix C). At round t our current policy samples y_t^-. Each failure spends up to K successor samples, and a first accepted sample joins the buffer. Every run we report uses $K = 1$. No score-based selection among candidates is used, so collection follows a learner's own failure distribution (Ross et al., 2011; Agarwal et al., 2024). Further rounds would expose failures created by earlier updates, and every run we report uses one. Gating is priced against a uniform policy that queries a successor on random training tasks at an identical call budget. Both policies share one admissibility rule and differ only in where calls go.

4.3 UPDATING THE PATCH

Both response strings are retokenized by our legacy tokenizer, so a successor may use a different tokenizer or expose only sampled text. Our update is a pairwise objective over those triples. With length-normalized response score

$$s_\pi(y \mid x) = \frac{1}{|y|} \sum_{j=1}^{|y|} \log \pi(y_j \mid x, y_{<j}) \tag{4}$$

and patch-induced change $\rho_\theta(x, y) = s_{\pi_\theta}(y \mid x) - s_{\pi_0}(y \mid x)$, our patch minimizes

$$\mathcal{L}_\Delta(\theta) = -\mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}_\Delta} \log \sigma\big(\beta[\rho_\theta(x, y^+) - \rho_\theta(x, y^-)]\big)\,. \tag{5}$$

In Equation 5, σ is a logistic function and β a preference temperature. We call this update *functional backporting*. A patch is never asked to match its successor, only to rank a verified repair above its own failure, so a successor regression has no path into the update. Subtracting reference score s_{π_0}

follows direct preference optimization (Rafailov et al., 2023) and asks a patch to prefer each repair more than an unmodified predecessor did. Averaging log-likelihoods over length follows SimPO (Meng et al., 2024), and setting s_{π_0} to zero gives its reference-free, zero-margin form. Every patch is a LoRA adapter (Hu et al., 2022), leaving deployment interface and serving stack unchanged.

4.4 Controls that separate the questions

A successor-guided recovery leaves two questions that one arm cannot answer. Does an update work because a model's own failure enters it, or because of where its positive comes from? Three positive-only arms train on accepted responses alone. Positive-only arms take positives from a successor on gated failures (An et al., 2023), from a successor on random tasks, or from predecessor successes (Zelikman et al., 2022; Singh et al., 2024). Comparing them with FB-S isolates failure-aware credit from supervision source. Only GSM8K runs every arm, so the isolation is complete there alone. And is a successor necessary at all? Two routes never query one. FB-P runs Equation 5 with its positive drawn from a verified predecessor success on that same failed task. FB-P is FB-S's pre-registered matched zero-successor control and shares capacity, pair count, and update count. Group-relative policy optimization (Shao et al., 2024) instead applies our executable reward to predecessor samples, with no pairs at all. If either recovers a gap, successor supervision is not necessary on that scope.

4.5 Judging the patch

Recovery is a ratio of Equation 2, and preservation a ratio of Equation 3 to our churn floor from Section 3. Normalisation is needed because an unchanged model already fails some of its own successes on resampling. Cost keeps four separate coordinates: successor calls, successor tokens, self-generated tokens and update tokens. A call-matched comparison alone would hide response length, and successor-free routes spend in another currency altogether. A forward-KL anchor on predecessor successes is tested as a safeguard (Appendix E). Let $S_i = \{x : V_x(\pi_i(x)) = 1\}$ collect tasks each released policy solves. An oracle switch that uses π_1 on $S_1 \setminus S_0$ and π_0 elsewhere succeeds on $S_0 \cup S_1$ and weakly dominates either source (Appendix A). A backport approximates that union inside one policy, so its score can exceed a successor's without exceeding successor capability.

5 Experiments

5.1 Models, scopes, and scoring

Qwen2-1.5B-Instruct serves as π_0 and Qwen2.5-1.5B-Instruct as π_1 (Yang et al., 2024a;b), both public under Apache 2.0. Same-generation 7B and 0.5B pairs fail our resolvable-gap gate on GSM8K (Appendix F), so three scopes remain. GSM8K (Cobbe et al., 2021) has gap 0.1089 $[0.0935, 0.1242]$. MATH (Hendrycks et al., 2021), a longer-horizon benchmark in the same family, has gap 0.1920 $[0.1743, 0.2094]$. MBPP (Austin et al., 2021) has gap 0.0941 $[0.0524, 0.1358]$ and is verified by executing each task's own asserts in a resource-limited subprocess (Chen et al., 2021). A second lineage repeats the audit with a different vendor and tokenizer. Phi-3-mini-4k-instruct serves as π_0 and Phi-3.5-mini-instruct as π_1 (Abdin et al., 2024), both 3.8B and MIT-licensed. Phi runs follow the protocol below with no re-selection, and Table 1 records where each Phi scope stops. Collection and development splits come from dataset-native training data under a published seed. Official test items never enter collection or any selection decision. Each official test split stays sealed until Section 6 opens it once for confirmation, after every operating point has been fixed.

Answers are scored semantically, accepting any stated final answer (on MATH, the last boxed one) as in the flexible-extraction filter of LM Evaluation Harness[2]. A second coordinate scores interface compliance. Scoring compliance alone conflates capability with formatting (Sclar et al., 2024). Compliance-only scoring reorders methods and can leave a version gap unresolvable (Appendix G).

5.2 Arms and protocol

An unmodified predecessor and a released successor define zero recovery and a version ceiling. Successor-guided arms are FB-S (Equation 5), its reference-free variant, direct preference optimiza-

[2] https://github.com/EleutherAI/lm-evaluation-harness

Table 1: Version audit, grouped by lineage. Each row is an audited scope, with a solid square for a stage passed and a chip where the gain stops. Events count paired development rollouts with a predecessor failure and a successor success. Verdicts give five-seed recovery of FB-S, FB-P and verifier-only GRPO (VO) where each installs the gain. FB-S and FB-P are ours, with successor and predecessor repairs. Phi-3 MBPP's verdict is an absolute net gain, since a ratio has no interval there.

		version gap				
Pair	Scope	π_0	π_1	events	stages	Verdict
Qwen 1.5B	● GSM8K	0.711	0.820	1,086	■■■■	installed FB-S 1.159 / VO 1.231 / FB-P 1.322
Qwen 1.5B	■ MATH	0.368	0.560	1,496	■■■■	installed VO 0.134 / FB-P 0.041
Qwen 1.5B	♦ MBPP	0.386	0.480	122	■■■■	installed VO 0.346
Qwen 7B	○ GSM8K	0.946	0.944	162	stop –––	no usable gap -0.003
Qwen 0.5B	○ GSM8K	0.475	0.314	816	stop –––	gap reversed -0.161
Phi 3.8B	♦ MBPP	0.575	0.620	68	■■■■	installed VO net +0.016
Phi 3.8B	○ GSM8K	0.908	0.899	235	stop –––	no usable gap -0.009
Phi 3.8B	○ MATH	0.452	0.440	494	stop –––	no usable gap -0.012

tion, and SimPO. Running published objectives over identical pairs isolates which ingredient recovery depends on. Alongside them run three positive-only arms from Section 4.4: uniform sequence KD, repair SFT, and self-improvement SFT. Successor-free arms are FB-P and verifier-only GRPO. Both run on GSM8K and MATH, and GRPO also on MBPP (Appendix J). Two comparators read successor logits instead. On-policy GKD (Agarwal et al., 2024) runs its released trainer unmodified, with β selected on GSM8K development. DistiLLM-2 (Ko et al., 2025) runs its released implementation and configuration. Logit access breaks the black-box assumption our other arms keep, so both are reported apart from them (Section 6.2). Successor-supervised arms are matched on successor calls and training pairs. Every arm but DistiLLM-2 shares LoRA modules, optimizer and budget.

Each collection round samples one current-policy rollout per task. Attempts, tokens, and verifier executions all count toward cost. Every arm runs five seeds with paired task-and-seed bootstrap intervals (Koehn, 2004). Every pairwise and supervised arm trains for one epoch, and GRPO runs a round schedule from Appendix D. We select on development by recovery, subject to a negative-flip ratio at or below one. Selection moves the pre-specified LoRA rank from 16 to 8. Only FB-S needs the move. At rank 16, FB-S recovers 0.540, while FB-P recovers 1.418 $[1.275, 1.594]$, above its own rank-8 value. Every other pre-specified value is left unchanged. Every confirmatory run on MATH and MBPP uses the selected point, and four further MATH points at seed 17 serve as diagnostics (Appendix M). Both operating points are reported. Appendices J, Q, and R give protocol revisions, reproducibility details and pre-registered hypotheses.

6 Results

Development results use 1,495 GSM8K, 1,496 MATH and 186 MBPP tasks at four samples each, scored semantically. Intervals are paired task-and-seed bootstrap over five seeds. We report negative flips as a ratio to the predecessor's own resampling churn, which is 0.183 on GSM8K. An unchanged patch scores 1.00, and a ratio below one means fewer flips than resampling, not none.

6.1 Every resolvable gap moves, but only one moves under successor guidance

Four of eight audited scopes pass our gate (Table 1). A scope counts as installed when some patch meets both recovery and retention criteria. At 7B, neither generation separates from the other. At 0.5B the successor is worse, and the reversal survives two extraction-insensitive checks (Appendix F). A newer release is not by itself evidence of a gain to backport. Half the scopes we audited carry none, and a recovery ratio on them would divide by a quantity that is not reliably positive.

Our second lineage carries the same gate to a different vendor. Phi-3.5-mini falls below its predecessor on GSM8K and on MATH. Both intervals exclude zero, both magnitudes stay below our 0.03, and neither scope reaches training. MBPP resolves at 0.044 $[0.013, 0.077]$. Phi-3's 80 failures yield 24 verified pairs from successor calls and 33 from self-repair. A pairwise arm needs 50, so both repair sources stop at ACQUIRE. A resolvable gain can still be out of reach for pairwise supervision, for

Table 2: Logit-reading comparators beside FB-S, FB-P and verifier-only GRPO, five seeds each. Settings are selected on GSM8K, except DistiLLM-2's released one, so MATH and MBPP columns report how a route transfers. Lower panel: the second lineage, as net gain since its gap is too small for a ratio. Teal clears zero, vermilion falls below it, and grey spans it. Bold marks column bests.

			GSM8K		MATH		MBPP	
Update	Access	Own failure	CGR ↑	NFR/churn ↓	CGR ↑	NFR/churn ↓	CGR ↑	NFR/churn ↓
On-policy GKD	`logits`	no	-0.310 $[-0.49, -0.17]$	1.26 ▲	-0.836 $[-0.97, -0.72]$	1.97 ▲	0.026 $[-0.20, 0.21]$	1.02 ▲
DistiLLM-2	`logits`	no	-0.442 $[-0.66, -0.27]$	1.33 ▲	-0.602 $[-0.72, -0.50]$	1.68 ▲	0.057 $[-0.34, 0.32]$	1.02 ▲
FB-S (ours)	`text`	yes	1.159 $[1.03, 1.31]$	0.49	-0.893 $[-1.05, -0.76]$	2.04 ▲	0.009 $[-0.18, 0.16]$	1.00 ▲
FB-P (ours)	`none`	yes	**1.322** $[1.19, 1.48]$	**0.40**	0.041 $[0.01, 0.08]$	0.94	stops at ACQUIRE	
Verifier-only GRPO	`none`	yes	1.231 $[1.07, 1.43]$	0.44	**0.134** $[0.09, 0.18]$	**0.85**	**0.346** $[0.03, 0.65]$	**0.89**

Second lineage, Phi-3 MBPP

Update	Access	Net gain ↑		NFR/churn ↓
On-policy GKD	`logits`	0.0051	$[-0.0081, 0.0183]$	0.94
Verifier-only GRPO	`none`	**0.0156**	$[0.0030, 0.0288]$	**0.89**

want of pairs. Verifier-only GRPO needs no pairs and installs the gain at $[0.003, 0.029]$ over five seeds, flipping at 0.89 of churn. Both MBPP scopes show one pattern. Either source yields only 24 to 60 verified pairs there, and the pair-free verifier route is the one that installs.

Among our Qwen lineage's three resolvable gaps, FB-S recovers only GSM8K, the intermediate one. MATH has the largest gap and MBPP the smallest. On both, five seeds at the confirmatory point leave FB-S below or indistinguishable from zero (Table 2). No MATH operating point at seed 17 lifts it above 0.03. MATH's shortfall is not a shortage of supervision. MATH is densest in verified repairs and gets our largest update (Table 8). Yet its pairwise loss moves $15\times$ less than on GSM8K at the same operating point. Each update either does nothing or destroys an answer interface (Appendix M). Repair source breaks it. FB-P installs MATH there at 0.041 $[0.006, 0.076]$, 0.935 $[0.842, 1.030]$ above FB-S. FB-P keeps the answer marker at 0.900, where FB-S falls to 0.621.

We therefore ran verifier-only GRPO on both resistant scopes at the frozen operating point, with a falsifier fixed in advance (Appendix J). Both scopes recover, at $[0.091, 0.177]$ on MATH and $[0.033, 0.651]$ on MBPP. Flip ratios stay at 0.85 and 0.89 of churn. What failed on MATH and MBPP was FB-S, not the scope. Which route a scope needs is itself for CBack to measure. A fixed development-only rule names an installing route on all four resolvable scopes (Appendix L). Section 6.4's mechanism is GSM8K's. On MATH, self-improvement SFT never sees a failure yet matches GRPO to within 0.002 $[-0.006, 0.010]$ in score, and gap size predicts none of it.

6.2 Two published methods with successor logits install nothing

We test whether FB-S rediscovers distillation through a weaker interface against two published methods that read successor logits. On-policy GKD matches those logits on learner samples (Agarwal et al., 2024). GKD runs at $\lambda = 1$ and the $\beta = 0.9$ its development sweep froze, and every swept value regressed (Appendix J). DistiLLM-2 raises the likelihood of successor responses and lowers that of learner responses under its released Qwen2.5-1.5B configuration (Ko et al., 2025). Neither recovers the gap, and both regress the predecessor (Table 2). Both flip more than the churn floor, and every GSM8K and MATH seed of both scores below the untrained predecessor. In Example 5 (Appendix O), DistiLLM-2 copies a successor error that FB-S avoids.

On GSM8K, access separates the arms by an order of magnitude more than their objectives do. Paired over identical tasks and seeds, every pair that crosses into logit access separates by at least 1.47 in recovery. Differences among the three recovering arms stay at or below 0.163 (Section 6.5), and the two logit readers differ by -0.132 $[-0.246, -0.023]$. Access dominates only on GSM8K. On MBPP, only the verifier-driven route installs anything. Remaining arms cannot be told apart, and FB-S minus GKD is -0.002 $[-0.017, 0.013]$ in score. On MATH, both successor-free routes install and neither logit reader does. FB-P stands 0.643 $[0.568, 0.716]$ above DistiLLM-2, while FB-S regresses furthest, 0.291 $[0.204, 0.381]$ below DistiLLM-2 at twice the churn floor. What holds on every scope we ran is that privileged access never bought reachability. Our second lineage repeats that distinction on its own terms (Table 2, lower panel). On the one Phi-3 scope whose gap clears the gate, GKD installs nothing at 0.005 $[-0.008, 0.018]$. GKD also lands -0.010 $[-0.021, -0.001]$ below verifier-only GRPO, which does install the gain. A different model family, a different tokenizer and an execution verifier give the same answer as Qwen's answer-matching scopes.

Table 3: Every GSM8K arm on development and on our sealed official test split. Arms that train on their own failures stay above those that do not, while magnitudes and within-group ordering do not carry across. Bold marks each column's best. The exhaustive development table is Table 12.

Objective	Successor access	Own failure	development CGR ↑		development NFR/churn ↓	sealed official test CGR ↑		sealed official test NFR/churn ↓
Self-improvement SFT	no	no	−0.339	[−0.53, −0.17]	1.25 ▲	−0.532	[−0.67, −0.41]	1.50 ▲
Uniform sequence KD	text	no	−0.155	[−0.32, −0.02]	1.18 ▲	0.241	[0.18, 0.30]	0.85
Repair SFT	text	no	−0.177	[−0.34, −0.05]	1.19 ▲	0.250	[0.19, 0.31]	0.84
SimPO, released setting	text	yes	1.170	[1.05, 1.31]	0.48	0.542	[0.49, 0.60]	0.58
Verifier-only GRPO	none	yes	1.231	[1.07, 1.43]	0.44	0.503	[0.43, 0.58]	0.60
FB-S (ours)	text	yes	1.159	[1.03, 1.31]	0.49	0.356	[0.29, 0.42]	0.77
FB-P (ours)	none	yes	**1.322**	[1.19, 1.48]	**0.40**	**0.575**	[0.52, 0.64]	**0.56**

Table 4: One objective, capacity, and pair count, five seeds, with repairs from the successor (FB-S) or predecessor (FB-P). Rank 16 is the pre-specified capacity. Every paired difference excludes zero.

CGR ↑	GSM8K, development	GSM8K, sealed test	GSM8K, rank 16	MATH, development
FB-S (ours), successor repairs	1.159	0.356	0.540	−0.893
FB-P (ours), predecessor repairs	**1.322**	**0.575**	**1.418**	**0.041**
FB-P − FB-S, paired	0.163 [0.07, 0.24]	0.220 [0.16, 0.29]	0.878 [0.75, 1.01]	0.935 [0.84, 1.03]

6.3 A SEALED SPLIT RESCALES RECOVERY AND REORDERS THE ARMS

We sealed GSM8K's official test split before collection and opened it once, after every operating point was selected (Appendix P). Sealed, the version gap grows to 0.241 $[0.224, 0.259]$ from 0.109. A development partition drawn from the dataset's own training data flatters both released models. Our predecessor reaches 0.711 on development against 0.471 sealed, and installing arms' recovery ratios fall accordingly. Our strongest arm recovers 0.575 $[0.516, 0.636]$ against 1.322 on development, so any reading that a patch matches its successor does not survive.

On 1,319 tasks, every arm that trains on its own failures stays above every arm that does not, as on development. FB-S, verifier-only GRPO and FB-P overlap partly on development but become pairwise distinct. Sealing also reorders them. SimPO at its frozen setting reaches 0.542 $[0.486, 0.599]$, above FB-S by 0.186 $[0.125, 0.248]$, and is indistinguishable from either successor-free route. A development tie need not hold out of sample. No arm exceeds FB-P on the sealed split. FB-P's gain has the shape of the version gap. FB-P earns 0.67 of its gain on tasks both releases solve, against 0.65 for the gap itself, and only $+0.005$ on tasks neither solves (Table 16). Most of what is backported is reliability rather than new capability, like most of the version gain. Sealing also changes a qualitative verdict. On development, every positive-only arm regresses its predecessor. On the sealed split, only self-improvement SFT does, at -0.532 $[-0.671, -0.409]$. A development regression therefore need not be a property of the method.

6.4 ON GSM8K, FAILURE-AWARE CONTRAST IMPROVES RECOVERY

Our decisive pair is self-improvement SFT against FB-P. Both learn only from the predecessor's verified samples and never query a successor. Self-improvement SFT imitates successes on tasks the predecessor already solves, while FB-P ranks a self-repair above the model's own failure on that task. Self-improvement SFT and FB-P land at worst and best recovery on the sealed split (Table 3). With identical successor positives, the same contrast lifts FB-S above repair SFT by 0.105 $[0.038, 0.172]$ sealed. Failure-aware contrast improves recovery, and the repair's source decides how far. FB-P leads FB-S in every setting where both ran, and FB-P installs MATH where FB-S collapses (Table 4).

Development arms separate the same way. Every positive-only arm finishes below an untrained predecessor and flips more of its own successes than resampling does. Section 6.3 shows that part of this regression belongs to the partition. Among pairwise objectives on identical pairs, only the length-normalized ones recover fully. Two SimPO settings, FB-S and its reference-free form do so at about half the churn floor. Overlapping intervals leave no ordering among these four. Length normalization alone separates partial from full recovery, while reference subtraction changes nothing

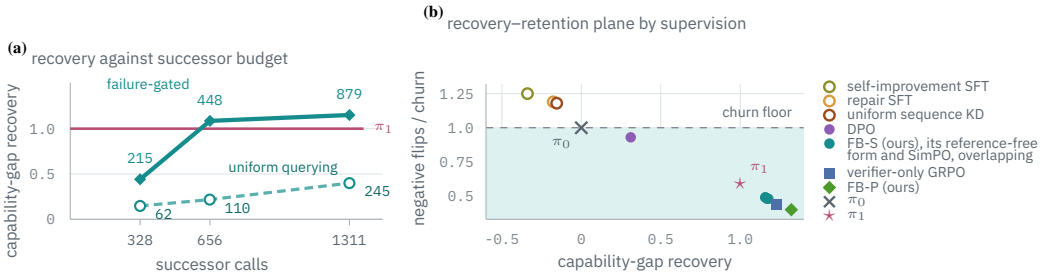

Figure 2: **(a)** Gap recovery against successor calls, a seed-17 diagnostic, with the admissible events collected at each budget. **(b)** Five-seed gap recovery against flips over churn for each black-box GSM8K arm.

measurable. Capacity shifts level but not ranking (Appendix I). None of this ordering carries to MATH, where an identical operating point collapses FB-S (Figure 6b) and not FB-P.

6.5 Successor access buys acquisition efficiency, not reachability

Two routes that never query a successor recover this gap, and FB-P is the strongest arm we measure (Table 3). FB-P differs from FB-S only in where the repair comes from. Predecessor samples repair $1{,}043$ of $1{,}311$ failed tasks. Paired over identical tasks and seeds, FB-P exceeds FB-S by 0.163 $[0.075, 0.244]$. GRPO against FB-S gives 0.072 $[-0.074, 0.228]$, and FB-P against GRPO gives 0.091 $[-0.051, 0.214]$, both including zero. Successor supervision is not necessary to recover this gap. A successor-guided recovery therefore does not show that its gain came from that successor. Only the reachability half of this extends to MATH and MBPP (Section 6.1).

Querying a successor changes acquisition cost, and how much depends on which cost is counted. With objective and admissibility fixed, failure gating returns 879 admissible events at $1{,}311$ calls, against 245 for uniform querying. Gating yields $3.6\times$ more per call and $3.2\times$ more per successor token (Figure 2a and Appendix M). Sampling a predecessor until it repairs each failure costs about $1{,}030$ own tokens per pair, against 460 successor tokens under gating. Recovery and retention move together, and no arm buys recovery by flipping more (Figure 2b). Successor access lowered what a pair costs, and it did not change whether this gap could be recovered.

Counting every generated and update token changes the ordering again. On GSM8K, verifier-only GRPO is the cheapest recovering arm at 1.43M tokens, against 2.00M for FB-S and 2.58M for FB-P (Table 15). On MATH, FB-P spends about $5{,}500$ predecessor tokens per verified pair against about $1{,}770$ successor tokens for FB-S, yet only FB-P installs the gain. A successor is thus a way to buy pairs cheaply, not a condition for recovery, and its value grows with the cost of self-repair on the task. Our development-only route rule (Appendix L) reaches the same conclusion. Our rule selects FB-P on GSM8K and verifier-only GRPO on the other three resolvable scopes, and none of the selected routes queries a successor. Safeguards, capacity sweeps and parser sensitivity are reported in Appendix K.

7 Conclusion

Can a newer model's gains be backported to an older one? Yes, on every scope where a gain could first be shown to exist, and each time without the newer model. Built on the version event, CBack keeps a successor's verified fixes apart from its new mistakes and turns a release delta into three measurements. Of eight audited scopes in two lineages, only four carry a resolvable gain. Gap size predicts neither which scopes recover nor which route reaches them. Successor access was never needed for reachability, though it changed acquisition efficiency. On GSM8K, of two successor-free arms learning from the predecessor's own verified samples, only the one trained against its own failures recovers on a sealed split. Failure-aware contrast improves recovery there, and a predecessor-drawn repair decides how much of it does. Two published methods that read successor logits regressed it there. A deployment can therefore narrow a verified version gap without calling the newer release. Whether a deployment needs its successor at all should be measured before it is assumed.

AI USE STATEMENT

We conceived the research question and the CBack protocol, and formulated its hypotheses. We designed and supervised every experiment and made all methodological and interpretive decisions. Within that work, we used generative AI tools as assistants in several required-disclosure tasks. Generative AI helped us refine hypotheses and formalize the conceptual framework and its mathematical claims, including the short proofs of Appendix A. AI tools also gave feedback on experiment design, helped implement and run methods, and helped interpret results. We have not used generative AI tools to generate synthetic data sets, to translate, or to clean and reformat datasets. Model rollouts that CBack trains on are the object of study and are described in Section 4. Qualitative or thematic data analysis is not applicable to this work. Among recommended-disclosure tasks, AI tools suggested experimental parameters and helped create and edit code and figures. AI tools also helped us search for and summarize literature, format references, and draft and edit text. We have reviewed all AI-assisted work. We checked all cited primary sources and verified the code against its unit tests and against the logged run behind every reported number. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Backporting installs whatever an executable verifier accepts, so the same procedure that carries a wanted gain can carry an unwanted one. A deployed patch therefore needs an allowlist over the capabilities it may install and a rollback path to the unmodified predecessor. Every model, dataset, and verifier we use is public and is cited with its revision.

REPRODUCIBILITY STATEMENT

Code for every stage is released anonymously at `https://anonymous.4open.science/r/CBack-C2BC`, with one entry point per stage and the pinned configuration of every scope. Section 5 and Appendix B give model and data revisions and verifier interfaces. Appendix C gives query accounting and stopping rules, and Appendix D gives training budgets and swept values. Protocol revisions are in Appendix J and the sealed-split procedure is in Appendix P. Statistical tests are described in Section 6, and Appendix Q lists what each run logs.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, et al. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219, 2024. doi: 10.48550/ARXIV.2404.14219. URL https://doi.org/10.48550/arXiv.2404.14219.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes LLM better reasoner. *CoRR*, abs/2310.20689, 2023. doi: 10.48550/ARXIV.2310.20689. URL https://doi.org/10.48550/arXiv.2310.20689.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Sheng Cao, Mingrui Wu, Karthik Prasad, Yuandong Tian, and Zechun Liu. ParamΔ for direct mixing: Post-train large language model at zero cost. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=vqbd2OQnGp.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé 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, Joshua 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. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

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. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Jessica Maria Echterhoff, Fartash Faghri, Raviteja Vemulapalli, Ting-Yao Hu, Chun-Liang Li, Oncel Tuzel, and Hadi Pouransari. MUSCLE: A model update strategy for compatible LLM evolution. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, volume EMNLP 2024 of *Findings of ACL*, pp. 7320–7332. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-EMNLP.430. URL https://doi.org/10.18653/v1/2024.findings-emnlp.430.

Mahdi Milani Fard, Quentin Cormier, Kevin Robert Canini, and Maya R. Gupta. Launch and iterate: Reducing prediction churn. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3171–3179, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/dc5c768b5dc76a084531934b34601977-Abstract.html.

Naibin Gu, Peng Fu, Xiyu Liu, Ke Ma, Zheng Lin, and Weiping Wang. Adapt once, thrive with updates: Transferable parameter-efficient fine-tuning on evolving base models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 14765–14783. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.719. URL https://doi.org/10.18653/v1/2025.acl-long.719.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

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 J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Byeongho Heo, Jaehui Hwang, Sangdoo Yun, and Dongyoon Han. On-policy delta distillation. *CoRR*, abs/2607.15161, 2026. doi: 10.48550/ARXIV.2607.15161. URL https://doi.org/10.48550/arXiv.2607.15161.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Rana Muhammad Shahroz Khan, Pingzhi Li, Sukwon Yun, Zhenyu Wang, Shahriar Nirjon, Chau-Wai Wong, and Tianlong Chen. PortLLM: Personalizing evolving large language models with training-free and portable model patches. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=gyHoR6uFhU.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Xavier Carreras, and Kevin Duh (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1317–1327. The Association for Computational Linguistics, 2016. doi: 10.18653/V1/D16-1139. URL https://doi.org/10.18653/v1/d16-1139.

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

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24872–24895. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/ko24c.html.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A contrastive approach boosts the distillation of LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/ko25a.html.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pp. 388–395. ACL, 2004. URL https://aclanthology.org/W04-3250/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124, 2024. doi: 10.48550/ARXIV.2411.15124. URL https://doi.org/10.48550/arXiv.2411.15124.

Jiahuan Li, Shanbo Cheng, Shujian Huang, and Jiajun Chen. MT-PATCHER: selective and extendable knowledge distillation from large language models for machine translation. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 6445–6459. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.358. URL https://doi.org/10.18653/v1/2024.naacl-long.358.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

Pin-Jie Lin, Rishab Balasubramanian, Fengyuan Liu, Nikhil Kandpal, and Tu Vu. Efficient model development through fine-tuning transfer. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 2617–2636. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.131. URL https://doi.org/10.18653/v1/2025.emnlp-main.131.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/e099c1c9699814af0be873a175361713-Abstract-Conference.html.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Yifan Niu, Han Xiao, Dongyi Liu, Zelong Wang, Dihong Gong, Yasheng Wang, and Jia Li. Breaking the tokenizer barrier: On-policy distillation across model families. *CoRR*, abs/2606.09456, 2026. doi: 10.48550/ARXIV.2606.09456. URL https://doi.org/10.48550/arXiv.2606.09456.

Thinh On, Hengzhi Pei, Leonard Lausen, and George Karypis. Knowledge distillation for large language models through residual learning. In *The Fourteenth International Conference on Learning Representations, ICLR 2026*, 2026. URL https://openreview.net/forum?id=Dh6KxUxG20.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In

Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pp. 627–635. JMLR.org, 2011. URL http://proceedings.mlr.press/v15/ross11a/ross11a.pdf.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=RIu5lyNXjT.

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. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Ridwan Shariffdeen, Xiang Gao, Gregory J. Duck, Shin Hwei Tan, Julia Lawall, and Abhik Roychoudhury. Automated patch backporting in Linux (experience paper). In Cristian Cadar and Xiangyu Zhang (eds.), *ISSTA '21: 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, Denmark, July 11-17, 2021*, pp. 633–645. ACM, 2021. doi: 10.1145/3460319.3464821. URL https://doi.org/10.1145/3460319.3464821.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T. Parisi, Abhishek Kumar, Alexander A. Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=lNAyUngGFK.

Frederik Träuble, Julius von Kügelgen, Matthäus Kleindessner, Francesco Locatello, Bernhard Schölkopf, and Peter V. Gehler. Backward-compatible prediction updates: A probabilistic approach. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 116–128, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/012d9fe15b2493f21902cd55603382ec-Abstract.html.

Hao Wang, Kun Yuan, Wenlin Zhong, Minglei Zhang, Han Xiao, Ming Sun, and Honggang Qi. Cross-tokenizer on-policy distillation via byte-prefix marginalization. *CoRR*, abs/2607.22334, 2026. doi: 10.48550/ARXIV.2607.22334. URL https://doi.org/10.48550/arXiv.2607.22334.

Yuqing Xie, Yi-An Lai, Yuanjun Xiong, Yi Zhang, and Stefano Soatto. Regression bugs are in your model! Measuring, reducing and analyzing regressions in NLP model updates. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 6589–6602. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.515. URL https://doi.org/10.18653/v1/2021.acl-long.515.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-congruent training: Towards regression-free model updates. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 14299–14308. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01407. URL https://openaccess.thecvf.com/content/CVPR2021/html/Yan_Positive-Congruent_Training_Towards_Regression-Free_Model_Updates_CVPR_2021_paper.html.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *CoRR*, abs/2407.10671, 2024a. doi: 10.48550/ARXIV.2407.10671. URL https://doi.org/10.48550/arXiv.2407.10671.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024b. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

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. *CoRR*, abs/2308.01825, 2023. doi: 10.48550/ARXIV.2308.01825. URL https://doi.org/10.48550/arXiv.2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Conference.html.

Qihuang Zhong, Liang Ding, Li Shen, Juhua Liu, Bo Du, and Dacheng Tao. Revisiting knowledge distillation for autoregressive language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 10900–10913. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.587. URL https://doi.org/10.18653/v1/2024.acl-long.587.

A Proofs and additional definitions

Deployment-interface obstruction. Assume $V_x(y) = 0$ for every $y \in \mathcal{Y}_0(x)$. Any patch that preserves the legacy deployment envelope has support contained in $\mathcal{Y}_0(x)$. Hence $\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[V_x(y)] = 0$ for every admissible θ. No such patch can backport the capability. □

Oracle functional switch. For deterministic policies, define $S_i = \{x : V_x(\pi_i(x)) = 1\}$. Let q use $\pi_1(x)$ when $x \in S_1 \setminus S_0$ and use $\pi_0(x)$ otherwise. Policy q succeeds exactly on $S_0 \cup S_1$. Since $S_0 \subseteq S_0 \cup S_1$ and $S_1 \subseteq S_0 \cup S_1$, its pointwise success weakly dominates both source policies. A finite model may not represent or learn q, and the proposition only identifies the verified target that dense imitation fails to isolate. □

Definition 2 (Backportability frontier)**.** *Given a preservation tolerance ϵ and a component-wise budget b, the backportability of slice c is*

$$\mathcal{B}_c(b, \epsilon) = \sup_{\theta:\, C(\theta) \preceq b,\, \mathrm{NFR}_c(\theta) \leq \epsilon} \mathrm{CGR}_c(\theta). \tag{6}$$

Empirical runs approximate the Pareto set over gap recovery, negative flips, and each cost coordinate.

Non-backportable capabilities. Backporting cannot remove a hard deployment-interface limit. Let $\mathcal{Y}_0(x)$ be all outputs admissible under the legacy context length, modality, tool protocol, and runtime. If $V_x(y) = 0$ for every $y \in \mathcal{Y}_0(x)$, no parameter update that retains that envelope can solve x. Our deployment-interface obstruction above gives the support argument. Primary experiments focus on capabilities expressible through the legacy text and tool-call interface. New modalities and contexts beyond the legacy hard limit form a registered non-backportable control.

B Data split and verifier protocol

Every source is downloaded at a recorded revision and hashed before use. Official test examples are read only by evaluation jobs. Training and development partitions come from source-provided training data with seed 2027. Near-duplicate checks operate before rollout collection and compare normalized problem statements across train, development, and test. A detected overlap is removed from training and recorded, never silently rewritten.

GSM8K answers are parsed from the declared final-answer field when present and otherwise from the last parseable quantity, then compared after numeric normalization. MATH extracts the last boxed expression and compares after the standard normalization of Hendrycks et al. (2021). MBPP executes the generated program together with the task's own asserts in a resource-limited subprocess with a wall-time limit, and a non-zero exit or a timeout counts as a failure. Model text never receives reference answers, unit tests, evaluator traces, or hidden environment state.

C Collection and optimization details

At round zero, $\pi_{\theta_0} = \pi_0$. Collection samples without replacement inside a round. A failed current-policy rollout receives up to K successor samples, and every reported collection uses $K = 1$ over a single round. A first verified sample becomes y^+, the current-policy failure becomes y^-, and all attempts remain in the cost ledger. A task with no repair stays in the yield denominator and can be revisited only after a later policy update.

Arms with the registered preservation anchor draw equal numbers of delta pairs and legacy-success anchor trajectories in each optimization round, and every other arm trains on delta pairs alone. Prefixes for Equation 7 are sampled uniformly over assistant tokens. Reference logits are cached once with a model, tokenizer, and prompt hash. Multiplier updates use a disjoint development set. Successor collection stops after three consecutive yield checks below $\tau = 0.02$, each over 256 queries. A budget of 1.5M successor output tokens always takes precedence. No reported collection met the yield rule. GSM8K and MBPP queried every failure once, and MATH stopped on the token budget.

D Hyperparameter and budget registry

Generation uses nucleus sampling (Holtzman et al., 2020) at temperature 0.7 with $p = 0.95$ and four samples per task. Responses are capped at $1{,}024$ new tokens on GSM8K and MBPP and $2{,}048$ on MATH. vLLM (Kwon et al., 2023) serves them in bfloat16. Adapters train for one epoch with AdamW (Loshchilov & Hutter, 2019). Warmup is linear over 3% of steps. Training clips gradients at 1.0 and uses micro-batch one with gradient accumulation eight. Adapter scale is tied to rank as $2r$. Verifiable-reward training samples groups of four from the current policy over three rounds on 440 prompts under a reference-KL coefficient of 0.04. Seeds are 17, 29, 43, 71, and 101 throughout. Table 5 lists every value swept, and Appendix H reports their results (Table 11).

All methods receive the same maximum sequence length and training-token budget within a comparison. Gradient accumulation can change with hardware, but the effective batch and optimizer update

Table 5: Values swept, with the pre-specified and the development-selected settings marked. Every sweep runs at seed 17, and the two marked settings run at all five seeds.

Item	Values run	Setting
Adapter rank	4, 8, 16, 32	16 registered, 8 selected
Adapter scale	8, 16, 32, 64	tied to rank as $2r$
Learning rate	10^{-6}, 3×10^{-6}, 10^{-5}, 3×10^{-5}	10^{-5}
Preference temperature β	0.03, 0.1, 0.3, 1.0	0.1
Epochs	1, 2, 3	1
Successor query fraction	0.25, 0.5, 1.0	1.0
Preservation multiplier step	0 fixed, 0.05 adaptive	not retained
Evaluation samples	$K_{\text{eval}} = 4$	–
Seeds	17, 29, 43, 71, 101	all five

Table 6: Additions to FB-S, each at seed 17 on GSM8K against the unconstrained patch at the same adapter rank. Every one of them costs recovery, and none lowers the negative-flip ratio below what the unconstrained update at that rank already reaches. The rank-8 reference is the seed-17 run of the five-seed report. Table 11 evaluates the same configuration in a separate run, hence 1.127 here against 1.151 there.

Update	Successor access	Own failure	Score	CGR ↑		NFR/churn ↓
No addition, rank 8	yes	yes	0.834	1.127	[1.01, 1.28]	0.51
+ fixed forward-KL anchor	yes	yes	0.749	0.353	[0.26, 0.44]	0.82
+ adaptive anchor, first round	yes	yes	0.746	0.320	[0.22, 0.41]	0.87
+ adaptive anchor, after dual update	yes	yes	0.722	0.101	[−0.00, 0.20]	0.97
No addition, rank 16	yes	yes	0.763	0.482	[0.36, 0.60]	0.90
+ predecessor-success replay	yes	yes	0.729	0.166	[0.05, 0.28]	0.94
+ a second epoch	yes	yes	0.239	−4.330	[−5.17, −3.70]	4.03

count cannot. Every patch reported here is a LoRA adapter, and we do not vary full-parameter or cross-tokenizer training.

E Registered preservation constraint

A deployment may require an explicit regression bound, and the constraint below is available for that purpose. Our recommended procedure excludes the anchor. Explicit anchoring was registered as a hypothesis and tested. Development evidence did not support including it (Table 6). Our final procedure therefore has three parts: version-relative gain identification, budget-aware failure-gated acquisition, and length-normalized preference optimization over verified failure–repair pairs. Full-response replay of predecessor successes likewise reduces recovery rather than protecting it.

Legacy-success trajectories provide prefixes h for a local behavior anchor:

$$\mathcal{L}_{\text{keep}}(\theta) = \mathbb{E}_{h\sim\mathcal{D}_{\text{keep}}}\mathrm{KL}(\pi_0(\cdot \mid h) \,\|\, \pi_\theta(\cdot \mid h)) \,. \tag{7}$$

Verifiers certify a final answer rather than a reasoning trace, so a churn ratio below one rules out excess flipping rather than capability loss. Forward KL is a preservation surrogate rather than a universal choice for distillation. Since this anchor already costs recovery without lowering measured flipping, we do not pursue other divergences. Training at round t minimizes

$$\mathcal{L}_t(\theta) = \mathcal{L}_\Delta(\theta) + \lambda_t \mathcal{L}_{\text{keep}}(\theta), \qquad \lambda_{t+1} = \Big[\lambda_t + \eta_\lambda(\widehat{\mathrm{NFR}}_t - \epsilon)\Big]_+ . \tag{8}$$

Here λ_t is the anchor multiplier, η_λ its step size and $\widehat{\mathrm{NFR}}_t$ the development negative-flip rate after round t. Development evaluation, rather than the KL value, updates the multiplier. An increase in measured negative flips tightens the next update. LoRA is the patch parameterization throughout (Hu et al., 2022). Whether a low frontier would lift under full-parameter training remains an open question.

Table 7: Every audited scope, grouped by lineage. Rows with a negative gap carry no usable version delta and no recovery ratio is defined for them. Phi-3-mini's MBPP gain is resolvable and its 24 successor and 33 self-repair pairs fall below the 50 a pairwise arm needs.

Pair	Benchmark	Predecessor	Successor	Gap (95% CI)
Qwen2-1.5B → Qwen2.5-1.5B	GSM8K	0.7109	0.8197	0.1089 [0.0935, 0.1242]
Qwen2-1.5B → Qwen2.5-1.5B	MATH	0.3681	0.5602	0.1920 [0.1743, 0.2094]
Qwen2-1.5B → Qwen2.5-1.5B	MBPP	0.3858	0.4798	0.0941 [0.0524, 0.1358]
Qwen2-7B → Qwen2.5-7B	GSM8K	0.9465	0.9436	−0.0028 [−0.0105, 0.0048]
Qwen2-0.5B → Qwen2.5-0.5B	GSM8K	0.4749	0.3139	−0.1610 [−0.1809, −0.1408]
Phi-3-mini → Phi-3.5-mini	MBPP	0.5753	0.6196	0.0444 [0.0134, 0.0766]
Phi-3-mini → Phi-3.5-mini	GSM8K	0.9084	0.8992	−0.0092 [−0.0177, −0.0007]
Phi-3-mini → Phi-3.5-mini	MATH	0.4520	0.4400	−0.0120 [−0.0241, −0.0005]

Table 8: Supervision available to FB-S on each Qwen scope. Events and density are development-split counts at four samples per task. Pairs, update tokens, and mean repair length in predecessor tokens come from the seed-17 ledgers. CGR is FB-S's five-seed recovery (Table 2). MATH receives the densest supervision and the largest update, and recovers least.

		development improvement events		acquisition and update			FB-S
Benchmark	tasks	count	per task	pairs	update tok.	repair tok.	CGR
● GSM8K	1,495	1,086	0.73	879	596,235	288	**1.159**
■ MATH	1,496	1,496	**1.00**	847	1,004,958	528	−0.893
♦ MBPP	186	122	0.66	60	29,199	71	0.009

F SCOPE AUDITS AND THE RESOLVABLE-GAP GATE

A scope enters the recovery analysis only when its paired bootstrap interval gives a successor advantage with a point estimate of at least $\delta_{\min} = 0.03$ and a lower bound above zero. Table 7 lists every scope we audited, including those the gate refused.

Both 7B models are saturated. At 0.5B, the reversal is not a parsing artefact. An extraction-insensitive final-line proxy gives 0.4923 against 0.4023, and accuracy restricted to explicitly marked answers gives 0.5917 against 0.3022. Mean response lengths match at 212 and 214 tokens. Our development splits derive from dataset training data, so differential memorisation between generations is a live explanation for both.

Table 8 lists the supervision FB-S receives on each resolvable Qwen scope. MATH has the most improvement events per task and receives the largest update, yet FB-S recovers least there, so supervision volume does not explain its shortfall.

Table 9: Same seed-17 adapters, same rollouts, three parsers. Arm ranking inverts across parsers. Under the strict parser, the version gap itself is unresolvable at 0.0067 [−0.0104, 0.0232], so no recovery ratio is defined in that column.

Arm	Semantic	Marker-equivalent	Strict
Predecessor	0.7109	0.3452	0.2082
Successor	0.8197	0.6072	0.2149
FB-S (ours)	**0.7634**	0.2542	0.0003
Repair SFT	0.6863	0.4764	0.1657
Uniform sequence KD	0.6943	**0.4814**	0.2000
Self-improvement SFT	0.6831	0.4259	**0.3124**

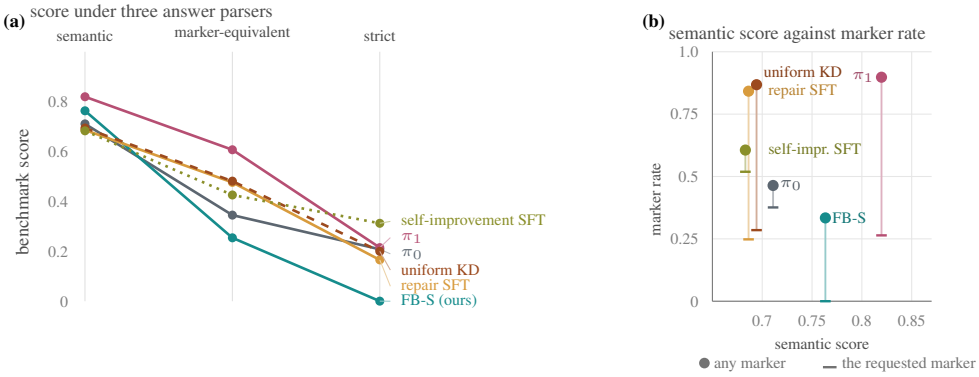

Figure 3: (a) Six GSM8K arms scored under three parsers on identical seed-17 rollouts. (b) Same arms by semantic score against the rate of emitting any final-answer marker (dot) and the requested marker (bar).

Table 10: Semantic correctness against output-interface compliance, seed-17 adapters on GSM8K. The last column conditions on responses carrying a declared marker.

Arm	Semantic	Requested marker	Any marker	Semantic | marker
Predecessor	0.7109	0.376	0.464	0.745
Successor	0.8197	0.264	0.898	0.836
FB-S (ours)	0.7634	0.000	0.334	**0.818**
Repair SFT	0.6863	0.248	0.842	0.708
Uniform sequence KD	0.6943	0.285	0.868	0.710
Self-improvement SFT	0.6831	0.519	0.606	0.704

G Answer parsers and interface compliance

Three parsers score every response. *Strict* accepts only the marker the prompt requests. *Marker-equivalent* additionally accepts a single trailing boxed answer. *Semantic* accepts any stated final answer, following the community flexible-extraction protocol of LM Evaluation Harness. Semantic acceptance is a strict superset of the other two, since no response passes a stricter parser and fails it. Table 9 scores the same seed-17 adapters under all three.

Table 10 separates the two coordinates. Strict scores track marker imitation rather than reasoning. Self-improvement leads it while having the lowest semantic accuracy of any arm, and our strongest arm adopts a different marker and scores near zero. On responses that carry a declared marker, extraction is unambiguous and the free-form fallback never fires. Arm ranking is unchanged on that subset. Figure 3 shows the same reversal. FB-S falls from the best semantic score among the patched arms to nearly zero under the strict parser, because FB-S adopts a boxed answer instead of the requested marker. Compliance and capability are close to orthogonal coordinates here.

Interface mismatch, not reasoning, drives the effect. On MATH both generations use the same marker natively, and the strict and semantic version gaps agree to within 0.006 (0.1982 against 0.1920). FB-P shares this sensitivity with FB-S. At the selected capacity and seed 17, the predecessor ends 0.376 of its responses with the requested #### marker. FB-S does so on 0.000 and FB-P on 0.003. A strict-marker parser therefore scores either patch at zero, while the semantic parser scores them at 0.834 and 0.854. Drawing the repair from the predecessor changes where the positive comes from, not whether the patch keeps the marker.

H Capacity, optimisation, and budget sweeps

Table 11 and Figure 4 report FB-S at seed 17 on GSM8K. Recovery falls monotonically with adapter capacity above rank 8. At the registered rank 16, recovery rises with preference temperature and peaks at half the acquisition budget. Larger learning rates or more epochs collapse it. FB-P does not share

Table 11: Development sweeps, FB-S at seed 17 on GSM8K. The registered point is rank 16, and the selected point is rank 8. Temperature, learning-rate, epoch and budget rows vary the rank-16 point. Query fractions take a leading prefix of the successor query stream, which is what a smaller acquisition budget would have collected.

Setting	Score	CGR	NFR
Rank 4	0.8343	1.134	0.084
Rank 8 (selected)	0.8361	1.151	0.089
Rank 16 (registered)	0.7634	0.482	0.164
Rank 32	0.6627	−0.442	0.267
$\beta = 0.03$	0.7569	0.422	0.168
$\beta = 0.3$	0.7774	0.611	0.150
$\beta = 1.0$	0.8237	1.037	0.099
Learning rate 3×10^{-6}	0.8249	1.048	0.083
Learning rate 3×10^{-5}	0.1522	−5.132	0.828
Three epochs	0.0174	−6.370	0.984
25% of the query stream	0.8099	0.909	0.086
50% of the query stream	0.8480	1.260	0.076

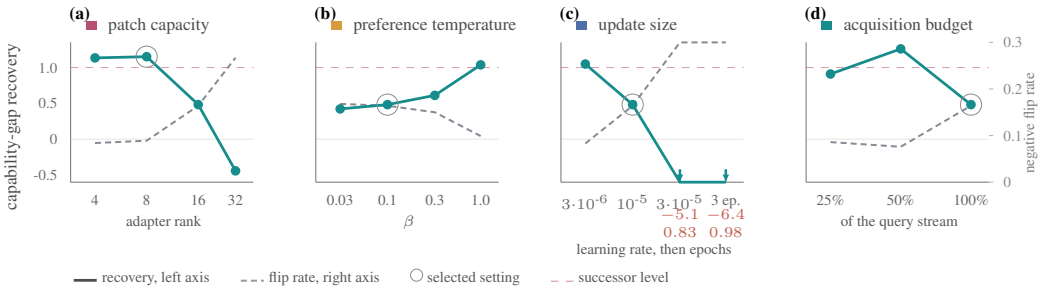

Figure 4: Development sweeps of FB-S at seed 17 on GSM8K. Each panel plots gap recovery (left axis) and negative-flip rate (right axis) against one setting. Panels (b) to (d) vary the registered rank-16 point. Two collapsed points in (c) are clipped at the axis floor, with their values printed under the ticks.

Table 12: Every GSM8K arm at the selected and the pre-specified capacity: patched-model score, gap recovery with its paired task-and-seed bootstrap interval over five seeds, and negative flips as a ratio to the predecessor's churn floor. Triangles mark ratios above the floor, and bold marks the best value in each column. Each table's bootstrap pools its own set of arms, so an interval end can differ from Table 3 by 0.01.

Objective	Successor access	Own failure	Score	CGR ↑		NFR/churn ↓
Predecessor π_0	–	–	0.711	0.000		1.00
Successor π_1	–	–	0.820	1.000		0.59
`development-selected capacity, rank 8, five seeds`						
Self-improvement SFT	no	no	0.674	−0.339	[−0.53, −0.17]	1.25 ▲
Repair SFT	yes	no	0.692	−0.177	[−0.34, −0.04]	1.19 ▲
Uniform sequence KD	yes	no	0.694	−0.155	[−0.32, −0.02]	1.18 ▲
Direct preference optimization	yes	yes	0.745	0.311	[0.19, 0.42]	0.93
SimPO, $\beta = 0.1$	yes	yes	0.837	1.158	[1.03, 1.31]	0.49
SimPO, released $\beta = 2.5, \gamma/\beta = 0.55$	yes	yes	0.838	1.170	[1.05, 1.31]	0.48
FB-S (ours)	yes	yes	0.837	1.159	[1.03, 1.31]	0.49
reference-free variant (ours)	yes	yes	0.839	1.177	[1.05, 1.32]	0.48
Verifier-only GRPO	no	yes	0.845	1.231	[1.07, 1.43]	0.44
FB-P (ours)	no	yes	**0.855**	**1.322**	[1.19, 1.48]	**0.40**
`pre-specified capacity, rank 16, five seeds`						
FB-S (ours)	yes	yes	0.770	0.540	[0.40, 0.67]	0.86
FB-P (ours)	no	yes	**0.865**	**1.418**	[1.28, 1.59]	**0.35**
Repair SFT	yes	no	0.688	−0.209	[−0.39, −0.07]	1.24 ▲

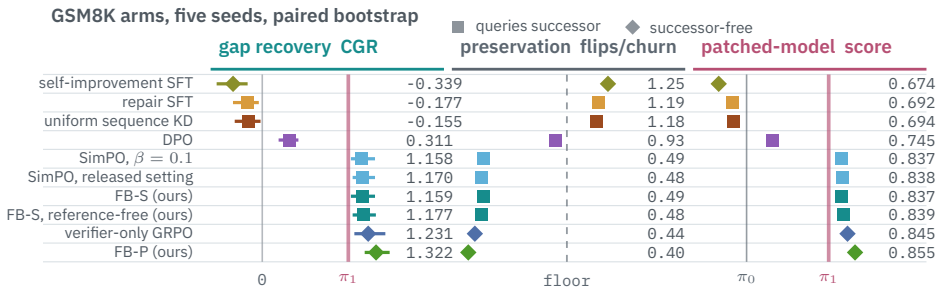

Figure 5: Every GSM8K arm at the selected capacity on gap recovery, flips over churn, and patched-model score, five seeds with paired bootstrap intervals (Table 12).

FB-S's capacity sensitivity. At seed 17, FB-P recovers 1.318 at rank 8, 1.318 at the pre-specified rank 16 and 1.072 at rank 32. FB-S recovers 1.151, 0.482 and −0.442 at the same ranks. Over five seeds at rank 16, FB-P recovers 1.418 [1.275, 1.594] against 0.540 for FB-S, a paired difference of 0.878 [0.748, 1.008]. FB-S's collapse above rank 8 therefore belongs to successor repairs rather than to the objective.

I Preference objectives on identical pairs

Table 12 lists every GSM8K arm at both capacities, the development counterpart of the sealed split in Table 3. Figure 5 plots the selected-capacity rows.

FB-S, its reference-free form, DPO, SimPO and repair SFT train on the same 879 verified failure–repair pairs. FB-P, uniform sequence KD and self-improvement SFT train on 879 examples from their own sources. Verifier-only GRPO trains on 440 prompts without pairs. Every arm runs at the selected operating point over five seeds. Table 13 varies reference subtraction and length normalisation on those identical pairs. Dropping length normalisation cuts recovery from about 1.16 to 0.311, while dropping the reference changes nothing measurable.

Table 13: The effect survives the choice of preference objective but not the removal of length normalisation. Names match Table 12, and intervals are paired task-and-seed bootstraps over the same five seeds. Bold marks each column's best.

Objective	Reference subtraction	Length-normalised	CGR		NFR/churn
FB-S, reference-free (ours)	no	yes	**1.177**	[1.054, 1.321]	**0.48**
FB-S (ours)	yes	yes	1.159	[1.032, 1.310]	0.49
SimPO, $\beta = 0.1$	no	yes	1.158	[1.032, 1.305]	0.49
Direct preference optimization	yes	no	0.311	[0.189, 0.418]	0.93

J PROTOCOL REVISIONS

Three components of the pre-registered protocol were revised before the confirmatory runs that depend on them, and three additions were registered during the study. Original values are reported alongside the final ones.

Model pair. The pre-registered primary pair was 7B. Our 7B gap audit found no resolvable gap, and the 0.5B pair reversed. Our primary pair therefore became 1.5B, and both audits are reported as negative controls for the gate.

Preservation tolerance. The pre-registered tolerance was an absolute negative-flip bound. Under stochastic decoding that coordinate has a non-zero null. Pairing independent samples of the predecessor with itself gives a churn floor of 0.183, and the successor itself sits at 0.108. A fixed bound below that floor would reject even the unchanged predecessor under independent sampling. Our tolerance is therefore a ratio to that floor. Floors are 0.183 on GSM8K, 0.295 on MATH, 0.254 on MBPP and 0.118 on Phi-3 MBPP. Every reported ratio therefore converts to an absolute rate.

Code benchmark. The pre-registered code family was CodeContests (Li et al., 2022), whose sandbox and release pin were unavailable in budget while a gap audit had to precede any training. MBPP substitutes for it under the official split roles, with items 11–510 sealed. No claim is made about CodeContests or LiveCodeBench (Jain et al., 2025).

A paired difference stated in recovery units is a difference in score divided by its scope's point gap, so its width carries the sampling error of the difference and not of that gap. Where a gap is small beside its own interval, as MBPP's 0.094 is, a difference is reported in score instead.

Privileged-access comparators. The registered arm set was black box by design. Two comparators that read successor logits were added to answer whether FB-S rediscovers distillation, with the same six interpretation branches fixed before either ran. On-policy GKD runs through TRL's[3] `GKDTrainer` unmodified at $\lambda = 1$, under the contract every arm shares and with prompts matched to the successor calls failure-gated collection spent on that scope. Logit access is also the most expensive supervision we run, at 437,087 successor token positions over 1,311 sequences for one GKD seed on GSM8K. GKD's β was swept over $\{0.1, 0.5, 0.9\}$ at seed 17 on GSM8K and selected by development recovery, subject to a negative-flip ratio at or below one. No value met that constraint, at 1.49, 1.38 and 1.21 of churn. A fallback registered before the last point finished therefore applies. We freeze the highest recovery of the three and report the arm as failing retention. Every swept value regressed its predecessor, at -0.777, -0.527 and -0.280, so the result is not an artefact of one setting. DistiLLM-2 runs from its official repository at a pinned commit under its released Qwen2.5-1.5B configuration, whose LoRA rank is 16 against the shared rank of 8. We report that capacity difference beside our own rank-16 result. DistiLLM-2's recipe trains a DoRA adapter, which our frozen evaluation backend cannot serve. DistiLLM-2's own merge utility therefore folds the adapter into the base weights before evaluation. GKD runs five seeds on every scope that passes the gate, and DistiLLM-2 on every Qwen scope that passes it.

Second model lineage. The registered protocol fixed one lineage. We added Phi-3-mini-4k-instruct to Phi-3.5-mini-instruct as a second lineage. Phi-3 tests whether a gate refusal reflects one vendor's release practice rather than a property of version differences. Phi-3 is audited on the same three scopes at the capacity frozen on Qwen GSM8K. Prompts, verifiers, decoding and evaluation code

[3] `https://github.com/huggingface/trl`

Table 14: Share of seed-17 GSM8K pairs solved by each patched predecessor in each paired verifier cell, from the frozen version-event record. The regression cell holds tasks the successor failed and the predecessor solved. Verifier-only GRPO and FB-P were not part of this decomposition. Bold marks each column's best.

	improvement	preserved success	regression	joint failure	score
share of pairs	0.182	0.638	0.073	0.108	
FB-S (ours), rank 8	**0.774**	**0.926**	**0.770**	**0.429**	**0.834**
FB-S (ours), rank 16	0.715	0.856	0.657	0.364	0.763
Repair SFT	0.575	0.822	0.556	0.247	0.696
Uniform sequence KD	0.592	0.813	0.529	0.266	0.693
Self-improvement SFT	0.415	0.785	0.747	0.235	0.656

are shared. No setting was searched on the Phi pair, and no Qwen number depends on it. Our gate refused GSM8K and MATH. MBPP resolved, and its 24 successor and 33 self-repair pairs stopped every pairwise arm before training.

Pre-registered follow-up on the two resistant scopes. Every update first tested on MATH and MBPP was successor-guided, so a not-installed verdict there could not separate scope from supervision route. Before running anything, we registered verifier-only GRPO on both scopes at the frozen operating point, with four interpretation branches fixed in advance. (i) If both scopes stay unrecovered, the barrier is the scope. (ii) If MATH recovers, the MATH barrier is the pairwise update rather than the scope. (iii) If MBPP recovers, its verdict is a route effect rather than a data-volume bound. (iv) If either scope degrades, the result is reported like the positive-only arms on GSM8K. Our registered statement was that neither scope recovers, and its falsifier was a materially positive recovery on either. Both scopes recovered, so that statement is rejected and the second and third branches apply. No hyperparameter was searched on either scope, and no GSM8K arm changed.

K Safeguards, capacity and answer parsing

Three further results bound the main findings. No safeguard lowers flips below an unconstrained update, and each costs recovery (Table 6). FB-S's recovery falls from 1.151 at rank 8 to -0.442 at rank 32 (Figure 4), a sensitivity FB-P does not share (Section 5). Answer interfaces also differ across generations. A strict-marker parser inverts every arm's ranking and leaves GSM8K's gap unresolvable at 0.0067 $[-0.0104, 0.0232]$ (Appendix G). Version comparisons must therefore score answer and interface separately.

L Route rule

CBack names the route a scope installs under by a rule that reads development reports only. A route is eligible when the paired-bootstrap lower bound of its capability-gap recovery exceeds zero and its negative-flip ratio is below one. An eligible route thus flips fewer predecessor successes than resampling does. Among eligible routes, the rule takes the highest lower bound and breaks a tie by the lower flip ratio. With no eligible route, the rule returns not-installed. Phi-3 MBPP admits no recovery interval, so its bound is taken on net gain, the scale of its reported verdict. Candidates are every registered route with a five-seed development report on the scope, published comparators included. Objective ablations of FB-S (DPO, SimPO at $\beta = 0.1$ and the reference-free form) are not routes. Qwen GSM8K has nine, MATH eight, MBPP four and Phi-3 MBPP two. FB-P is a candidate where it trains and is absent from both MBPP scopes, where its self-repair collection returns 46 verified pairs from 138 failures and 33 from 80 against a gate of 50. Our rule returns FB-P on Qwen GSM8K at a bound of 1.192. On Qwen MATH, Qwen MBPP and Phi-3 MBPP, the rule returns verifier-only GRPO at 0.092, 0.033 and a net gain of 0.003. No route it selects queries a successor.

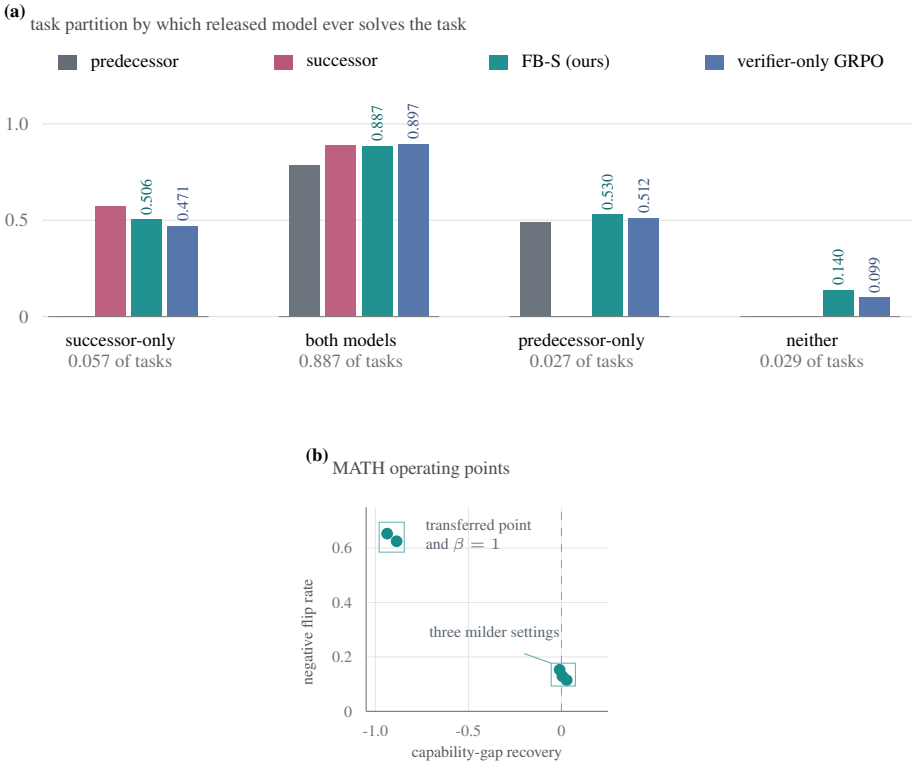

Figure 6: (a) Score on GSM8K development tasks partitioned by which released model ever solves them, seed 17, for the released models and two patched procedures. The successor scores zero on the predecessor-only partition by construction. (b) Five MATH operating points of FB-S at seed 17, gap recovery against negative flips.

Table 15: Cost of each GSM8K procedure at seed 17. Self-generated tokens are predecessor rollouts, and FB-P adds repair sampling. Recovery is a five-seed estimate except for uniform querying, a seed-17 sweep value.

	successor		self-generated	update	all-in	admissible	
Procedure	calls	tokens	tokens	tokens	tokens	events	CGR
FB-S (ours), failure-gated	1,311	403,658	997,633	596,235	1,997,526	879	1.159
FB-S objective, uniform querying	1,311	361,658	997,633	164,672	1,523,963	245	0.399
Verifier-only GRPO	0	0	943,002	483,221	**1,426,223**	–	1.231
FB-P (ours)	0	0	2,070,054	513,967	2,584,021	1,043	**1.322**

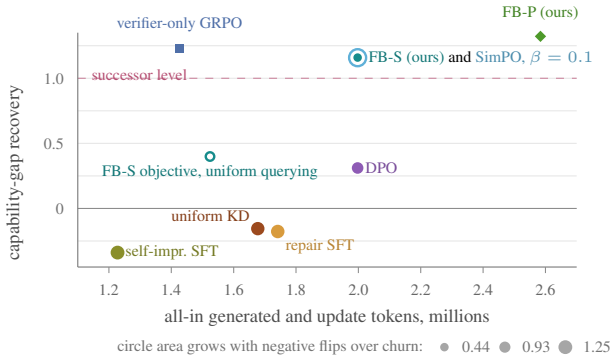

Figure 7: Gap recovery against all-in tokens for each GSM8K arm, counting predecessor rollouts, successor output, and update tokens at seed 17. Recovery is the five-seed estimate except for uniform querying, a seed-17 value. Circle area grows with negative flips over churn.

M Additional analyses

Figure 6a partitions the GSM8K evaluation set by which released model ever solves each task across four samples. Partitioning by solver is what makes a recovery ratio above one interpretable. Table 14 partitions the same rollouts by the paired verifier outcome that defines a version event. Both are computed on the seed-17 rollouts so that every arm is compared on one fixed partition. We report the five-seed aggregates in Table 12 as results. Both patched procedures in Figure 6a recover most of the successor-only capability. Both also hold the predecessor-only successes the successor lost, and the verifier-only procedure needs no successor call. Figure 6b shows FB-S's MATH operating points. At the transferred GSM8K point and at a higher preference temperature, recovery falls to -0.9 and two thirds of the predecessor's successes flip. Three milder settings stay within 0.03 of zero recovery, so nothing in the swept grid separates the two outcomes. Changing the repair's source does: FB-P at the transferred point recovers 0.041 $[0.006, 0.076]$ over five seeds at 0.94 of churn (Table 2).

Figure 2 plots what a successor budget buys and where each arm lands on the recovery–retention plane. Panel (b) plots the five-seed point estimate of every interval in Table 12. Table 15 then keeps each procedure's cost in separate coordinates. Successor calls, self-generated tokens and update tokens are never summed into one price.

Figure 7 charges the successor-guided arms a shared pool of $997{,}633$ predecessor rollout tokens plus their successor output. GRPO is charged its own $943{,}002$ generated tokens. FB-P adds $1{,}072{,}421$ tokens of repair sampling to the shared pool. Ordering by total cost differs from ordering by successor cost, with GRPO the cheapest recovering arm and FB-P the most expensive. MATH widens that difference. FB-P's self-repair sampling there spends $6{,}248{,}901$ predecessor tokens for $1{,}131$ verified pairs, about $5{,}500$ a pair. FB-S spends about $1{,}770$ successor tokens a pair. FB-P buys reachability on MATH with the predecessor's own compute, not cheaply.

N WHAT THE PATCHED MODEL RETAINS

Our union argument in Section 4 predicts that a successful patch lands closer to $S_0 \cup S_1$ than to the successor. FB-S does. Against the successor, the patched predecessor gains 0.0145 on tasks only the predecessor solves and 0.0040 on tasks neither solves. FB-S gives up 0.0037 on successor-only tasks (Figure 6a). FB-S also leads every positive-only arm in all four paired verifier cells (Table 14). FB-S's development score above the successor therefore reflects retained predecessor successes, and no imitation of the successor is needed to produce it.

O VERSION EVENTS IN THE MODEL'S OWN WORDS

Examples 1–5 show five development cases, one per outcome the analysis turns on. Responses are trimmed to the deciding step, and the verifier saw the full text. FB-S is the development-selected patch at seed 17.

Example 1: GSM8K, an improvement installed

π_0 stops at an intermediate quantity, and FB-S adds the missing step.

π_0 ✗	"…he has $6 \times 40 = 240$ shirts. #### 240"
π_1 ✓	"…the total is the sum of his shirts and pants, $240 + 40 = 280$. #### 280"
FB-S ✓	"…we add the shirts and pants: $240 + 40 = 280$. The answer is: 280"

Under a strict final-answer parser this success would score as a failure, since FB-S answers in prose.

Example 2: GSM8K, a successor regression not inherited

π_1 runs past the token cap, and FB-S keeps the predecessor's answer.

π_0 ✓	"…$20 - 12 = 8$ sandwiches left. #### 8"
π_1 ✗	"…To find out how many sandwiches she has left after keeping some for" *(truncated, no answer)*
FB-S ✓	"…subtract what she gave away: $20 - 12 = 8$."

Example 3: MATH at the transferred operating point

Arithmetic stays intact while the answer interface is lost.

π_0 ✓	"$f(-1) = (-1)^2 - 1 = 0$. Therefore the value is $\boxed{0}$."
FB-S ✗	"$f(-1) = (-1)^2 - 1 = 0$. So, the value of $f(-1)$ is 0." *(no box)*

Example 4: MBPP, a plausible program the task's asserts reject

FB-S ✗

```
def remove_spaces(s):  return ''.join(s.split())
```

Example 5: GSM8K, a successor error that distillation copies

Casey applies a base coat, paint and glitter. Each coat takes 20 *minutes to apply and* 20 *minutes to dry. How many minutes does it take? (answer* 120*)*

π_0 ✓	"…$40 + 40 + 40 = 120$ minutes."	$4/4$ samples correct
π_1 ✗	"…Total time $= 20 + 20 + 20 = 60$ minutes." *(drying left out)*	$1/4$
DistiLLM-2 ✗	"…Total time $= 20 + 20 + 20 = 60$ minutes"	$1/4$
GKD ✗	"…Total time $= 20$ minutes $+ 20$ minutes $+ 20$ minutes $= 60$ minutes"	$1/4$
FB-S ✓	"…$20 + 20 = 40$ minutes … $40 \times 3 = 120$ minutes."	$4/4$

Three of four DistiLLM-2 samples and one GKD sample repeat the successor's dropped drying time. FB-S never sees that error as a positive and keeps the predecessor's answer on every sample.

Example 3 is not isolated. Across the MATH development split the predecessor emits a boxed answer on 0.888 of responses and the successor on 0.975. FB-S at the transferred point emits one on only 0.621 of responses, and at raised temperature on 0.603. Milder settings that avoid the collapse keep the marker at 0.881 to 0.887, indistinguishable from the predecessor. FB-P at the transferred point keeps it at 0.900 while its recovery is positive. Interface drift on MATH therefore travels with the collapse rather than causing it, and scoring those rollouts under a corrected stated-answer fallback moves recovery by at most 0.018.

P THE SEALED TEST SPLIT

A split manifest built before any collection recorded the official GSM8K test split as sealed. Our manifest stored the item count and an identity hash over the normalized statements of all $1{,}319$ items. No process could read those rows. Our split loader raises on any file whose records carry a sealed label unless the caller passes an explicit override. No run before this evaluation passed it.

We opened the split once, after every operating point had been selected and every adapter trained. Our opening script rebuilds the rows from the pinned dataset revision and recomputes the identity hash. Our script refuses to write anything unless that hash matches the manifest, and the hash matched. Frozen for the evaluation were the model identities and revisions, the adapter rank, and every objective. So were the verifier and its source hash, the decoding protocol, and the five training seeds. Nothing was retrained and no hyperparameter was reconsidered. Each rollout and cost record carries the evaluation split it came from, so sealed and development rollouts cannot be pooled by accident. Scoring uses the same paired task-and-seed bootstrap as the development tables, and Table 3 reports the result. SimPO at its released setting joined the comparison after the opening. SimPO's setting had been frozen on development by its registered rule, and its five adapters were already trained. SimPO was therefore evaluated under the same frozen protocol, with nothing selected on the split. Every arm is pooled again with SimPO under one bootstrap.

Where a recovered gain comes from. Table 16 attributes each sealed arm's net gain to the tasks it comes from, partitioning tasks by which released model ever solves them in four samples. Our version gap is itself mostly a reliability gap. Tasks both releases can solve hold 0.65 of it, and tasks only the successor solves hold 0.40, less a small regression. FB-P recovers that same shape, with 0.67 and 0.30 of its gain in those partitions. FB-P gains $+0.005$ on tasks neither release solves, so its recovery sits where the version gain does rather than being improvement at large. Every arm but self-improvement SFT takes nearly the same amount from successor-only tasks, between 0.037 and 0.043. Arms separate by reliability on tasks a predecessor could already solve. FB-P gains $+0.093$ there and FB-S $+0.042$. Self-improvement SFT changes by -0.141, which is the whole of its regression.

Our churn floor is higher on this partition, 0.330 against 0.183. A predecessor is simply less reliable on tasks it has not effectively seen. All preservation ratios in Table 3 are computed against the sealed floor.

Table 16: Net gain over the predecessor on the sealed split, attributed to task partitions by which released model ever solves a task in four samples. A cell is the partition's share of tasks times the arm's success rate there less the predecessor's, five seeds, so each row sums to its net gain and the first row decomposes the version gap itself. Bold marks the best arm on the both partition and on net gain.

	successor only	both	predecessor only	neither	net
share of tasks	0.169	0.710	0.034	0.087	
Version gap (successor)	+0.097	+0.156	−0.013	+0.000	+0.241
FB-P (ours)	+0.042	**+0.093**	−0.002	+0.005	**+0.139**
SimPO, released setting	+0.043	+0.085	−0.002	+0.004	+0.131
Verifier-only GRPO	+0.037	+0.083	−0.003	+0.005	+0.121
FB-S (ours)	+0.042	+0.042	−0.003	+0.005	+0.086
Repair SFT	+0.037	+0.023	−0.004	+0.005	+0.060
Uniform sequence KD	+0.037	+0.022	−0.005	+0.004	+0.058
Self-improvement SFT	+0.016	−0.141	−0.006	+0.003	−0.128

Q Reproducibility and artifact discipline

Each run records its source commit, configuration hash and start time, together with model and tokenizer revisions, chat templates and dataset revisions. Evaluator versions, random seeds, package versions and hardware are recorded too. Query logs keep input and output token counts. We report a result only when it passes split, budget, verifier and seed checks, and unsuccessful runs are counted rather than dropped.

R Pre-registered claims

Two hypotheses were pre-registered before the confirmatory runs. H1 predicted that FB-S would outperform repair SFT and verifier-only GRPO on at least two resolved families. Confirmatory results reject H1. Verifier-only GRPO recovers every resolvable scope, whereas FB-S recovers GSM8K only. H2 is supported: failure-gated querying increases admissible repairs per successor token at matched successor calls. Together these outcomes separate backport reachability from the acquisition efficiency that successor access provides.

Development estimates come from the fixed development partitions, and the sealed official test split was opened once for the confirmatory evaluation of Appendix P. Every number we report traces to a logged run under a single evidence freeze.