CINDER: TWIN-REFERENCED TRAINING FOR RELEARNING-RESISTANT UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM unlearning methods that pass their own evaluations are routinely undone by a few steps of fine-tuning on part of the forget set. We argue the right reference for resistance is not "unrecoverable" but a retain-only twin — a model trained without the forget data — and train against it directly. CINDER simulates the attacker's own attack class inside training (full-parameter fine-tuning on a random half of the forget facts, first-order meta-gradient) and penalizes recovery of the held-out half through a softplus penalty anchored at the twin's post-attack floor. We evaluate with the hypotheses the right way round: method-level non-inferiority to the twin at a pre-specified margin — the spread between two independent retrains — across corpora, with every axis required. On sixteen corpus seeds at 0.5B, CINDER's post-attack recovery exceeds the twin's by 0.08 nats (one-sided 95% bound 0.21, margin 0.36), 1.17 nats below NPO (95% CI 0.94–1.39); on sixteen fresh corpora generated after the analysis was fixed, +0.16 nats (bound 0.29), 1.16 below NPO. It is non-inferior on six of eight axes; on the circuit criterion and utility non-inferiority is not shown. A frozen, preregistered eight-axis protocol, used as a screen, agrees: relearning is not rejected on 15 of 16 seeds (16 of 16 fresh) against at most 4 for five baseline families, and no axis on 8 of 16 (3 of 16 fresh, C1 rejecting eleven) against at most 2. Ablations over eight to sixteen seeds show that the floor, the full-parameter inner attack and the fact-level split are each needed, that a floor taken from the pretrained base — no twin at training time — works as well as the twin's, and that adding our floor to MUDMAN brings it within the margin. At 1.5B and 3B relearning is not rejected on any of twelve seeds, but utility is paid for. We report what this does not show: a pass of a finite battery bounds nothing, the objective overshoots its floor, and a twin, though not needed for training at 0.5B, is still needed to evaluate.

1 INTRODUCTION

Machine unlearning for LLMs — removing specific knowledge from a model whose training cannot be repeated — has a credibility problem: methods are evaluated by the loss they were trained to minimize, and a model that looks unlearned under those metrics is routinely not. A few steps of fine-tuning on a fraction of the forget set resurrects "removed" knowledge; quantization, paraphrase and in-context re-injection recover it without touching a weight; the field's own assessment is that irreversible, non-catastrophic forgetting remains exceptionally challenging (Xu et al., 2026). Certification frameworks answer with a battery of attacks calibrated against a *retain-only twin* — a model trained identically without the forget data, which fixes both the floor of legitimate ignorance and the tell of over-suppression. The certificate we build on (Anonymous, 2026) adds the decisive observation: a strong suppression adversary fools its circuit and representation criteria at once, yet budgeted relearning still rejects it. **Relearning resistance is the binding constraint.** Objectives that train against relearning exist (MUDMAN Sondej et al. (2025), TAR Tamirisa et al. (2025)); what none does is aim at the recovery of a model that never saw the data, rather than at making recovery as hard as possible.

If the certificate is the exam, can one teach to the test without Goodharting it? CINDER puts the binding criterion inside the objective: each meta step simulates the certificate's attack class — full-parameter fine-tuning on a random half of the forget facts — and penalizes post-attack recovery of the *held-out* half, hinged at the twin's floor. The target is not "cannot relearn" but "relearns no more than a model that never saw the forget data" — under the attacks and statistics we test, which is all any finite battery can say.

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.

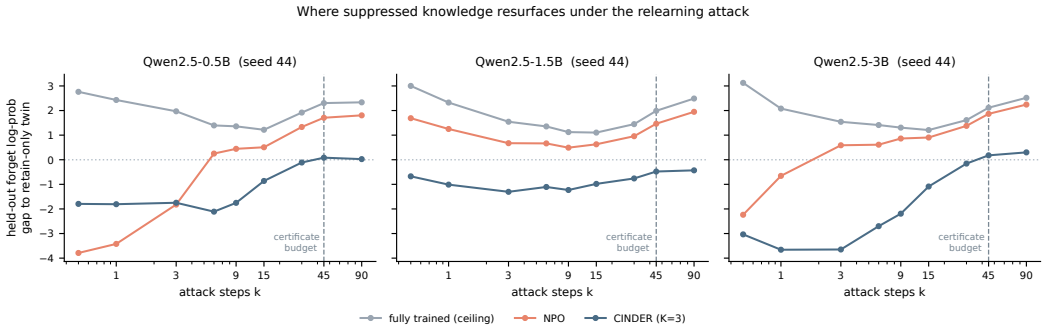

Figure 1: *Where suppressed knowledge resurfaces.* Held-out forget log-probability relative to a retain-only twin, against attack fine-tuning steps, for the fully trained model, NPO, and CINDER at 0.5B, 1.5B and 3B (headline-recipe models). The dashed line is the certificate's 45-step budget. NPO crosses the twin within six steps and sits 1.7–1.9 nats above it at the budget; CINDER stays at or below the twin through twice the budget at every scale.

Training against one's own evaluator invites Goodhart. We answer it three ways: the protocol is frozen upstream and never modified; the claims rest on a separate test with the hypotheses the right way round, fixed in writing before sixteen fresh corpus seeds were generated; and resistance is checked against attacks outside both the protocol and the training simulation.

Contributions.

1. **A twin-referenced relearning objective**, stated exactly (Section 4.1), including what its normalization does to the floor — it re-weights the push rather than stopping it, and the push is load-bearing (Appendix B.12).
2. **Method-level non-inferiority to the twin on relearning**, at the spread between two retrains, on sixteen headline and sixteen fresh seeds; NPO, MUDMAN and the untouched model are not non-inferior. On all eight axes together no method is: CINDER fails the circuit and utility criteria.
3. **Ablations over sixteen seeds**, not one: floor source (twin, split-matched twin, pretrained base), hinge shape and coefficient, a factorial over floor $\times$ attack class $\times$ fact split, and MUDMAN with and without our twin calibration (Section 5.5).
4. **What the evaluation cannot say**: a pass of the frozen protocol is a non-rejection, and four defects of that protocol, among them a blind spot for over-suppression that our own objective produces.

2 Related Work

Unlearning objectives and their reversibility. The dominant families — gradient ascent with retain regularization (GradDiff), negative preference optimization (NPO; Zhang et al. (2024)), representation misdirection (RMU; Li et al. (2024)) and localization-based editing (Guo et al., 2025) — optimize proxies measured at evaluation time, and those proxies decouple from erasure: minimal fine-tuning resurrects removed knowledge (Fan et al., 2025; Xu et al., 2026), quantization alone restores it (Zhang et al., 2025b), and pre-attack diagnostics predict the recovery (Goel et al., 2026); the same pair of failures — collateral over-unlearning and a cheap relearning attack — is formalized outside language models for class-level vision unlearning (Ha et al., 2026). Benchmark critiques (Thaker et al., 2025) blame methods and metrics co-evolving. Table 4 reproduces the landscape under one harness: every family fails, and the suppression family fails *only* relearning.

Robust-by-design unlearning. *RepSelect* (Sondej et al., 2026) collapses top principal components of forget-set weight gradients at a LoRA-adversary vantage point; we run a port of the authors' code (Section 5.4). Its battery is relearning, few-shot prompting and an adaptive attack on its own collapse mechanism — no quantization, extraction, probing, twin calibration or certificate; under our protocol it fails quantization, paraphrase and C1. *MUDMAN* (Sondej et al., 2025) is the closest relative of our inner loop — a persistent full-copy adversary, disruption masking and gradient normalization, evaluated at $\leq$1B against fine-tuning relearning only — and already reports that a LoRA adversary

is weaker than a full one. *TAR* (Tamirisa et al., 2025) trains safeguards against fine-tuning attacks by adversarial meta-learning. Neither references a retain-only twin; we port MUDMAN, add our twin calibration to it, and compare all three under one protocol (Section 5.5). Our contribution is not the meta-gradient but the twin-referenced target and its evaluation. The broader lineage — meta-unlearning for diffusion (Gao et al., 2025), sharpness-aware (Fan et al., 2025) and dual-space smoothness (Yan et al., 2026) objectives, invariance training (Wang et al., 2025), JensUn (Singh et al., 2025) and CIR (Sondej & Yang, 2025) — is robustness-motivated; none is evaluated against a multi-axis battery calibrated to a retain-only twin.

Certification and evaluation. We build on the certificate of (Anonymous, 2026): twin-null construction, a preregistered battery with per-axis conformal nulls and Holm-corrected decisions. Related infrastructure includes TOFU (Maini et al. (2024); our external arm), MUSE (Shi et al., 2025), RULER (Cosma & Finke, 2026) and a verification survey (Xue et al., 2026). MCU (Xiao et al., 2026) is an unlearning method targeting minor representation components that reports improved relearning resistance; we use it as a strong-suppression adversary here, where it fools the circuit and probe criteria yet is still rejected by budgeted relearning — the observation that motivates our objective. An interpretability study across six methods finds unlearning either insufficient or overly destructive (Zhang et al., 2025a). The certificate is frozen here; improving it is upstream work.

3 Preliminaries: the frozen certificate

Twin-null construction. For a training corpus split into forget and retain sets, a *full* model is fine-tuned on both and a *retain-only twin* on the retain set alone, from the same base with the same recipe. The twin operationalizes "never knew": any signal the twin exhibits on forget queries — under any attack — comes from priors and correlations, not the forget data. The twin is therefore both the calibration floor (an unlearned model should not be *more* recoverable than the twin) and the over-suppression null (it should not be conspicuously *less* decodable either).

Attack axes. The protocol scores seven forget axes, each producing a per-fact statistic with worst-case reduction over a preregistered budget grid where applicable: (i) *relearn* — full-parameter fine-tuning on half the forget subjects, scored on the other half, worst case over 3,6,10 epochs $\times$ 2e-5, 5e-5; (ii) *quant* — round-to-nearest quantization, worst case over 8,4,2 bits; (iii) *paraphrase* and (iv) *in-context re-injection* — evaluation-time attacks; (v) *UMA* extraction over attacker-knowledge fractions; (vi) *C1* — layer-wise activation-patching attribution of the recall circuit; (vii) *C2* — probe decodability of forget answers from question representations, worst case over three layers and linear, MLP probe families. A reversed-direction *utility* axis scores retain-set knowledge.

Why this protocol can be judged from this paper alone. The certificate (Anonymous, 2026) is concurrent work under review, so we do not ask the reader to take it on trust: it is frozen (vendored at a pinned commit, shipped with the artifact), falsifiable in both directions on evidence we report (the untouched model fails almost every axis; we document three places it misbehaves, Section 6), reproducible (our rerun of the upstream reference matches all twenty accept/reject flags), and the headline (Section 5.2) is an ordinary noninferiority comparison that needs none of it.

Decision rule. Sixteen retain-only twins (distinct seeds) are scored on every axis; the 240 ordered twin-pair differences form each axis's null. A method's statistic (mean shift against a reference twin) gets a conformal p-value per axis and Holm at $\alpha = 0.05$ decides jointly; the protocol calls a model *certified* iff no axis is rejected. We vendor the harness at a pinned upstream commit and never modify it; the shipped v1 utility axis is used for all decisions.

What a pass means, and what it does not. A small p-value says the method is more extreme than twin–twin differences; Holm controls how often a model that *is* twin-like gets rejected. A pass is therefore a failure to reject: it does not bound leakage, and a more conservative correction makes passing easier, not harder. The 240 pairs reuse sixteen twins and reverse pairs are exact negatives, so they are not 240 exchangeable draws and we claim no finite-sample validity for the p-value. We use the protocol as a frozen *screen* and report its verdicts as non-rejections (Section 5.3); the claims of this paper rest on a test with the hypotheses the right way round (Section 5.2).

4 METHOD

4.1 OBJECTIVE

Let θ be the model, F the forget texts grouped by fact, R the retain texts, ℓ_R the retain loss (per-example summed NLL) and $\mathrm{lp}(x)$ the summed log-probability of a whole forget training text x, question and answer (only padding is masked). The evaluation instead scores the mean log-probability of the answer tokens given the question, so training-side and evaluation-side numbers are in different units and are never compared directly. Each outer step on a retain batch minimizes $\ell_R(\theta) + c\,P(\theta)/(\mathrm{sg}|P(\theta)| + 10^{-8})$ with $c = 0.3\,\mathrm{sg}(\ell_R)$ and sg the stop-gradient: the λ rule that fixes the meta term at 30% of the task loss. P is computed on every second step:

1. **Split.** Draw half the forget *facts* as the attack set A; the rest is the evaluation set E (fact-level, so no fact straddles the split).
2. **Attack.** $\theta' = \theta$ after K = 3 AdamW steps (lr 2e-5, batch 8) of fine-tuning *all* parameters on A — the certificate's attack class, far shorter than its budget.
3. **Penalty.** $P = \mathrm{mean}_{x \in E}\,\mathrm{softplus}(\mathrm{lp}_{\theta'}(x) - f)$ over two batches of E.
4. **First order.** $\nabla_{\theta'}$ of the normalized term is added to θ's gradient (FOMAML), θ is restored, and one AdamW step (lr 2e-5) is taken. There is no forget-ascent term.

The floor f is one constant per corpus: the mean forget log-probability of a retain-only twin after the same K = 3 attack, estimated once before training, attacking and scoring the full forget list. A split-matched floor (attack half, score the other half, eight splits) differs by 0.08 nats on the development seed; Section 5.5 compares the two.

What the penalty does not do. $\mathrm{softplus}(0) = \log 2$, so the penalty is not zero at the floor, and the λ rule changes more than its scale. With $z_i = \mathrm{lp}_i - f$, example i receives gradient weight $c\,\sigma(z_i)/\sum_j \mathrm{softplus}(z_j)$: a push of total size near c wherever the batch sits, concentrated on examples still above the floor. The floor therefore *re-weights* the push; it does not stop it, and training runs past the twin (1.08 nats on average). Freezing the coefficient lets the gradient vanish below the floor and removes most of the resistance (Appendix B.12); a true hinge $\mathrm{relu}(z)$ with a frozen coefficient is compared in Section 5.5.

4.2 DESIGN CHOICES, AND THEIR SCOPE

1. **Anchor to a floor, not to zero** (the twin's, or — Section 5.5 — the pretrained base's). A raw log-probability penalty weakens as the model is suppressed (its normalized gradient scales as $1/|\mathrm{lp}|$), and heavier weights and forget ascent hurt relearning on the development seed.
2. **Simulate the evaluated attack class.** With a LoRA inner attack the penalty plateaus while full-parameter relearning still recovers; MUDMAN reports the same direction (a LoRA adversary is weaker than a full one, as its appendix on LoRA adversaries reports), so this is a confirmation in a new setting, not a discovery.
3. **Full-parameter outer updates.** Adapter-only outer training never lowered the penalty against a full-parameter attack in our runs. In our implementation the recipe costs 28 bytes per parameter (weights, gradients and AdamW state of both loops, plus activations), measured 0.5B–7B; cheaper optimizer routes we tried (pure bf16, SGD, Adafactor, 8-bit Adam; Appendix B.3) lost the relearning pass. This is an implementation cost, not a lower bound for the problem.

Section 5.5 re-tests these over eight to sixteen seeds instead of one.

5 EXPERIMENTS

5.1 SETUP

The primary testbed is the harness's synthetic entity-QA corpus: 80 subjects, 30 of them to forget, with paraphrase templates and a relation schema that enable the full battery. Each corpus seed gets a full model, a reference twin and sixteen protocol twins. Base model Qwen2.5-0.5B (fp32); 1.5B and 3B in Section 5.6. Methods: GA, NPO, RMU, mechanistic localization, a suppression red-team ("hide"), RepSelect (the authors' code, Section 5.4), MUDMAN and MUDMAN with our floor as its unlearning loss (our port, Appendix B.13), and CINDER. **Seed discipline.** Seed 44 is the *development* seed (the ablation ladder, memory routes and cost attempts of Appendix B ran on it) and seed 45

the *protocol-tuning* seed (seven logged judgments of the $\lambda 2/\lambda 3$ variants). Seeds 43 and 46–48 were judged once after the method was frozen and 49–58 under a written preregistration; together these are the sixteen *headline* seeds. Seeds 59–74 are *fresh*: generated after the analysis of Section 5.2 was fixed and written down, and used only to confirm it.

5.2 The central test, with the hypotheses the right way round

The claim is that CINDER relearns no more than a model that never saw the forget data. We test it as non-inferiority. For each corpus seed and axis, let E be the method's excess over the *mean* of the sixteen twins (for utility, its shortfall) and s the twins' standard deviation. The margin is $\delta = \sqrt{2}\, s$, the standard deviation of the difference between two twins: a method may exceed the average twin by no more than what separates two twins (on relearning, 0.36 nats on average). Each twin is attacked on a subject split drawn from its own seed and each method on one drawn from the corpus seed, so s contains retraining *and* attack-split variation, and the estimand is the method's mean excess over the twin pipeline as a whole. With the corpus seed as the unit, we test $H_0 : \mathbb{E}[E/s] \geq \sqrt{2}$ against $H_1 : \mathbb{E}[E/s] < \sqrt{2}$ by a one-sided t-test at $\alpha = 0.05$. A method is non-inferior overall only if every axis rejects H_0; by the intersection–union principle that joint claim needs no multiplicity correction. The margin and the test were fixed before the analysis first ran (LOG 2026-09-23); the headline seeds had been seen, the fresh seeds had not been generated. The analysis as first run clipped E/s at ±50, a numerical guard not in the written plan; it touches one CINDER utility value and no verdict, and both versions are released.

Table 1: Method-level non-inferiority to the twin (one-sided 95% upper bounds). Relearn columns are in nats of held-out log-probability after the certificate's relearning attack, excess over the twin mean; the margin is 0.36 nats. *Axes NI* counts the eight axes whose upper bound on E/s is below $\sqrt{2}$.

seeds	method	n	relearn excess [UCB]	NI	axes NI	not shown NI
43–58	CINDER	16	+0.08 [+0.21]	yes	6/8	c1, utility
43–58	NPO	16	+1.24 [+1.37]	no	2/8	relearn, quant, paraphrase, incontext, c1, c2
43–58	untouched	16	+1.70 [+1.74]	no	1/8	relearn, quant, paraphrase, incontext, uma, c1, c2
fresh	CINDER	16	+0.16 [+0.29]	yes	6/8	c1, utility
fresh	NPO	16	+1.32 [+1.42]	no	0/8	relearn, quant, paraphrase, incontext, uma, c1, c2, utility
fresh	untouched	16	+1.72 [+1.76]	no	1/8	relearn, quant, paraphrase, incontext, uma, c1, c2

Reading. On the headline seeds CINDER's relearning excess is 0.08 nats with a one-sided upper bound of 0.21, inside the 0.36 margin; NPO's is 1.24 (bound 1.37) and the untouched model's 1.70. The paired difference NPO − CINDER is 1.17 nats (95% CI 0.94 to 1.39; CINDER lower on 15 of 16 seeds, one-sided sign test p = 0.00026), and 1.12 [0.88, 1.36] without the two non-naive seeds. On the sixteen fresh seeds, generated after the test was fixed and never used for anything else, the result replicates: CINDER +0.16 nats (bound 0.29), NPO 1.32, paired difference 1.16 [0.99, 1.34], CINDER lower on all sixteen (sign test p = 1.5×10^{-5}), and the same two axes, C1 and utility, not shown non-inferior. Neither method is non-inferior on every axis: CINDER fails C1 and utility — the same two axes the frozen protocol flags (Section 5.3) — and NPO fails six. So the test in the right direction supports the paper's narrower claim, recovery at the twin's level under relearning, and does not establish the broader one: on C1 and utility non-inferiority is not shown, which is not the same as inferiority being shown.

The claim is about the method, not a model. Table 1 averages over corpora and over each method's own training noise; it says nothing about whether one particular unlearned model is within the margin, and we make no such claim. Figure 1 shows the same effect as recovery curves.

5.3 The frozen protocol as a screen

The per-seed protocol of Section 3 answers a different question — is the model flagged? — and we report it because it is frozen, preregistered and multi-axis, not because a pass certifies. Under a fixed-margin ε-rehearsal every baseline family fails at 0.5B and 1.5B (Table 4; a 24-point NPO sweep finds no passing configuration, Appendix B.9), and the suppression family fails *only* relearning. All then face the sixteen-seed protocol at those hyperparameters:

CINDER's rejections are C1 on 7 seeds, utility on 5, relearn and quantization on one each. On the sixteen fresh seeds relearning is never rejected, but C1 is on eleven, so no axis is rejected on only

Table 2: Frozen-protocol outcomes, sixteen headline seeds (Holm, $\alpha = 0.05$). A pass is a non-rejection. Wilson 95% intervals in brackets. RMU and localized editing (0/16 on both) barely forget here; RepSelect (two seeds) is in Section 5.4.

method	no axis rejected	relearn not rejected	mean rejected axes / seed
CINDER	**8/16** [0.28, 0.72]	**15/16** [0.72, 0.99]	**0.88**
NPO	2/16 [0.04, 0.36]	3/16	3.19
GA / GradDiff	0/16	3/16	2.38
suppression red-team	2/16	4/16	1.62
MUDMAN (port)	5/16	14/16	0.81
MUDMAN, our floor	5/16	14/16	1.06
untouched model	0/16	0/16	5.62

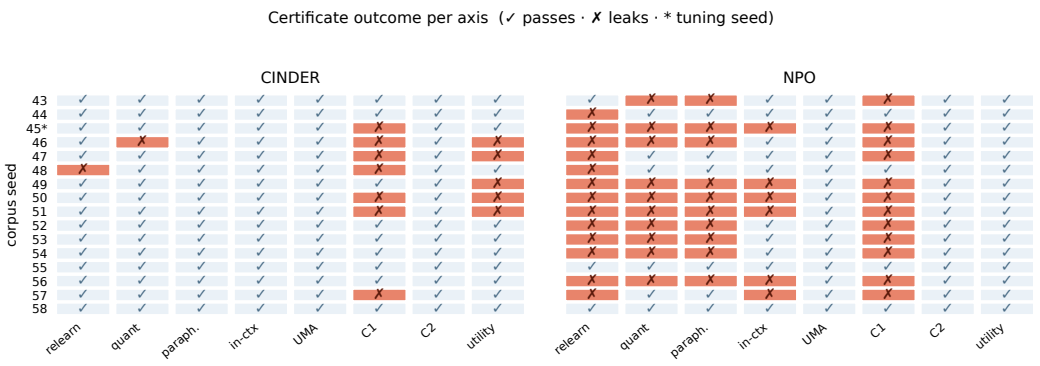

Figure 2: *Frozen-protocol outcome per axis and seed, headline seeds.* CINDER and NPO, sixteen corpus seeds × eight axes; a cross marks a Holm-rejected axis. Seed 44 is the development seed and 45 the protocol-tuning seed.

3 — the all-axis count is fragile across corpora, which is another reason Table 1 carries the claim. NPO's two all-axis passes are on seeds where CINDER also passes, so the paired comparison is 6 discordant seeds to 0 (exact McNemar p = 0.031; 5 to 0 and p = 0.063 without the two non-naive seeds) — suggestive, not decisive, and the reason Table 1 rather than this count carries the claim. The counts also move with the analysis choices a pass depends on: under a pooled null CINDER passes on 10, and under twin-level resampling its eight passes become probabilities of 0.54–0.95 (expected 6.4; Appendix D.3).

Why several axes. GA is not rejected on relearning on 3 of 16 seeds and passes all axes on none: it resists by destroying the model (forget log-probability -67 to -167 nats against the twin's ≈ -3), and utility rejects it on 13 of 16, including all three seeds where it survives relearning (Appendix B.11). On one axis GA would look like a peer of CINDER.

The circuit axis. CINDER was never trained against C1, and C1 is where it is most often rejected. On the span from twin (0) to trained model (1) by C1's own patching statistic CINDER sits at 0.18 on average, and the position ranks the verdict (exact permutation p = 0.00017, Appendix D.1). Both the position and the verdict come from the same statistic, so this shows the verdict is not noise; it is not independent evidence about a circuit, and 0.18 is a reduction of that statistic, not a fraction of a circuit removed. The circuit term $\lambda 3$ repairs C1 on the tuning seed but at utility's expense at every weight tried (Appendix B.4). A preregistered retain guard (skip the meta step while retain loss exceeds 5× its initial value) raises the all-axis count to 9/16 and cuts utility rejections from 5 to 2 (Table T2b); plain CINDER stays the headline.

5.4 Generalization and External Validity

- **Evaluation is harder than training.** Training simulates K = 3 steps on a *fact* split; the relearn axis takes the worst of 45–150 steps on a *subject* split.

- **Held-out attacks.** Six configurations outside the protocol grid and the training simulation (20 epochs; lr 1e-4; fractions .25/.75; an unseen split seed; a LoRA-r16 attack), judged by the ε-rehearsal (one twin, Welch over facts; $\varepsilon = 0.75$ at 0.5B, 0.5 at 1.5B): CINDER passes 6/6 at 0.5B against NPO's 2/6, and 14/24 over four seeds at 1.5B against 3/24. Graded, CINDER sits below the twin on 20 of 24 (mean -0.35 nats) and NPO on none (+1.78). These 24 are repeated measures on four models, not 24 independent tests.
- **A 5% forget set** (4 of 80 subjects, one seed): no axis is rejected, C1 included, while NPO is rejected on six — but CINDER sits 6.7 nats *below* the twin before any attack, over-suppression no axis registers (Appendix D.4).
- **RepSelect** (the authors' code, their retain-budget rule), seeds 43 and 44: rejected on quantization, paraphrase and C1 on both. Within its retain budget it forgets little here (forget EM 0.62–0.85 vs a twin's $\approx$0.10) and its published scale is 8–16B, so we read this as non-transfer, not defeat.
- **Two real corpora**, one seed each, ε-rehearsal on the behavioral axes (relearn, quantization, in-context, utility; $\varepsilon = 0.75$): CINDER passes on TOFU and on MUSE-News knowmem, NPO and the untouched model fail every forget axis. MUSE's retain split is ~9$\times$ smaller, so outer epochs were raised to match step count. By TOFU's own metric CINDER's Forget Quality is the highest of the three (p = 0.045; NPO 0.010) but still distinguishable from the retain model at 0.05, and its mean truth ratio is 25.7 against the retain model's 1.04 — a heavy over-forgetting tail the median hides (Appendix B.10). These are one-seed checks on QA subsets, not the benchmarks' full evaluations.

5.5 What the objective needs, over sixteen seeds

The design choices of Section 4.2 were found on one development seed. Here each is re-tested on the sixteen headline seeds (the factorial on eight), same twins, one judgment per run, all preregistered (LOG 2026-09-23). Relearn excess is Table 1's quantity.

Table 3: Ablations of the objective and the closest prior method. Relearn excess is Table 1's quantity, nats over the twin mean with its one-sided 95% upper bound; *NI* is relearning non-inferiority at the 0.36-nat margin, *axes NI* the axes non-inferior of eight, *screen pass* the seeds on which the frozen protocol rejects no axis.

variant	n	relearn excess [UCB]	NI	axes NI	screen pass
CINDER (headline)	16	+0.08 [+0.21]	yes	6/8	8/16
floor, held-out split	16	-0.11 [+0.05]	yes	6/8	7/16
floor from base (no twin)	16	+0.11 [+0.32]	yes	6/8	4/16
floor − 5 nats	8	-0.14 [+0.10]	yes	6/8	2/8
floor + 5 nats	8	+0.26 [+0.48]	no	4/8	3/8
numerator frozen	8	-0.06 [+0.22]	yes	5/8	4/8
numerator capped (2×)	8	+0.15 [+0.47]	no	4/8	2/8
coefficient frozen	16	+0.78 [+0.87]	no	2/8	1/16
relu, coeff. frozen	7	+0.74 [+0.88]	no	2/8	0/7
no floor	8	+0.80 [+0.94]	no	2/8	1/8
LoRA inner attack	8	+0.44 [+0.60]	no	2/8	2/8
no fact split	8	+0.29 [+0.42]	no	4/8	1/8
LoRA, no floor	8	+0.94 [+1.03]	no	1/8	0/8
no split, no floor	8	+0.67 [+0.79]	no	1/8	0/8
LoRA, no split	8	+0.35 [+0.67]	no	2/8	2/8
LoRA, no split, no floor	8	+0.87 [+1.03]	no	2/8	0/8
MUDMAN (port)	16	+0.27 [+0.39]	no	6/8	5/16
MUDMAN, our floor	16	+0.18 [+0.36]	yes	6/8	5/16
MUDMAN + retain guard	8	+1.62 [+1.74]	no	0/8	0/8
MUDMAN, our floor, + guard	8	+1.68 [+1.76]	no	1/8	0/8

Each component is needed. Removing the floor, replacing the full-parameter inner attack by LoRA, or dropping the fact-level split each loses relearning non-inferiority (excess 0.29–0.80 nats, bounds 0.42–0.94), and no combination of removals recovers it; this replaces the one-seed ladder of Appendix B.1 with eight seeds per cell. **Where the floor comes from matters less than that it is there.** A floor estimated on the meta step's own held-out split (-0.11, bound 0.05) or from the pretrained base with no twin at all (+0.11, bound 0.32) gives the same verdict as the twin's floor

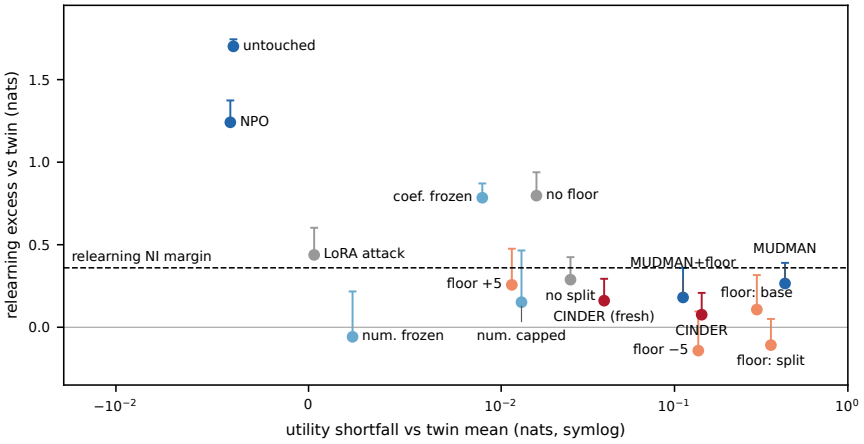

Figure 3: *What each variant pays for its resistance.* Relearning excess over the twin (with one-sided 95% upper bound) against utility shortfall — retain answer log-probability lost relative to the twin mean, nats, symlog axis — averaged over corpus seeds. The dashed line is the relearning NI margin.

over sixteen seeds, at a larger utility cost (Figure 3). Moving the floor matters in one direction only: -5 nats keeps non-inferiority (-0.14), +5 nats loses it (+0.26, bound 0.48) — a floor set too high stops the push early, which the re-weighting analysis of Section 4.1 predicts. **The push must persist, but need not grow.** Freezing the whole λ coefficient loses the resistance (+0.78); freezing only its numerator at the initial retain loss, keeping the adaptive denominator, keeps it (-0.06, bound 0.22) and cuts the mean utility shortfall on the same eight seeds from 0.025 to 0.002 nats — though utility is still not shown non-inferior on the standardized scale, and these eight seeds include the two development seeds, so this is a lead, not a replacement for the headline. Capping the numerator at twice its initial value does not keep it. **MUDMAN.** Our port at its best tuned configuration is not non-inferior on relearning (+0.27, bound 0.39) and loses utility at every configuration we tried (Appendix B.13); with our floor as its unlearning loss it reaches the margin (+0.18, bound 0.36) and loses less utility (0.11 vs 0.43 nats). MUDMAN's own retain rule (pause unlearning when retain loss rises by 0.05) stops it unlearning at all on this corpus (+1.62). So the floor transfers to another meta-unlearning method and improves it; CINDER's own advantage over MUDMAN here is 0.19 nats of relearning excess and a smaller utility loss, not a different verdict class.

5.6 Cost and scale

Cost, per request and per evaluation. At 0.5B CINDER trains in 153–174 s against NPO's 16 s (6× after a bit-identical change to where the inner loop parks its backup weights). The method itself needs one retain-only twin for the floor — 25 s here, which is also exact retraining, so on this testbed retraining is cheaper; unlearning earns its cost only where training cannot be repeated, the setting the corpus stands in for. A twin is not required even for that: a floor from the pretrained base gives the same relearning result (Section 5.5). The sixteen protocol twins (≈7 min to train, ≈35 min to score at 0.5B) are the *evaluation's* cost, paid once per corpus, not the method's. Memory is 28 bytes per parameter in our implementation (both loops' weights, gradients and AdamW state plus activations): 12.9, 40.3, 80.6 and 199.4 GB at 0.5B–7B, the 7B figure predicted from the smaller three to 0.2%. Cheaper routes we tried each lost the relearning pass (Appendix B.3); sharding runs the recipe unchanged at 1.5B and 3B.

Scale. One hyperparameter changes per scale, not monotonically: outer lr 2e-5, 1e-4 and 5e-5 at 0.5B, 1.5B and 3B, chosen on seed 44 at each scale. Relearning is not rejected on any of the twelve seeds above 0.5B — six at 1.5B and six at 3B — where the untouched model is rejected at the p-floor on every one. All-axis non-rejection is rarer: none of six at 1.5B, where utility rejects every seed (retain EM 0.890–0.965 against twins at 0.995–1.000, real damage, not the axis defect of Appendix D.4), and two of six at 3B: the pilot seed, whose pass depends on the correction (C1 at $p = 0.037$), and seed 49, preregistered and judged once, whose smallest axis p is 0.166. The other four each leak one or two axes (C1, utility, or both). The 1.5B learning rate that buys forgetting costs retained knowledge;

at 3B the meta gradient can spike the retain loss and whether training recovers depends on the draw (seed 45: retain EM 0.085). We have no schedule that preserves utility reliably above 0.5B, and 7B is a memory estimate, not a result. Per-axis p-values: Tables T6 and T6b.

6 Limitations and honest accounting

- **What is and is not shown.** Method-level non-inferiority to the twin on relearning, one-sided, at a margin of one twin-to-twin difference, on a synthetic corpus at 0.5B and replicated on sixteen fresh seeds. Not shown: anything about a single model (Section 5.2), that the knowledge is deleted rather than unrecoverable by these attacks, or non-inferiority on C1 and utility. The frozen protocol's passes are non-rejections and bound nothing.
- **Evaluation design.** Methods and twins are attacked on different subject splits (each drawn from the scored model's seed), so method–twin differences mix model and split variation; paired comparisons between methods (NPO − CINDER) share the split and are the cleaner evidence. Training and evaluation use different log-probability units (Section 4.1).
- **The twin must exist at training time.** The floor comes from a retain-only twin after a simulated attack; a floor from the pretrained base works as well at 0.5B (Table 3), so the twin is needed for evaluation but not for training — untested above 0.5B. And "never knew" is relative to the fine-tuning stage: twin and model share a pretrained base, so this is not yet a method for knowledge acquired in pretraining.
- **The objective overshoots.** The λ rule turns the floor into a re-weighting and keeps pushing past it (1.08 nats below the twin before attack, on the evaluation's answer log-probability). Freezing the coefficient removes the overshoot and the resistance with it (B.12); on a 5% forget set the overshoot reaches 6.7 nats and no axis sees it.
- **Scale and utility.** Sixteen headline and sixteen fresh seeds at 0.5B, six at 1.5B, six at 3B. Above 0.5B relearning transfers but utility is paid for, and all-axis passes are none of six at 1.5B and two of six at 3B.
- **Statistics.** The protocol's 240 ordered pairs are not exchangeable draws and we claim no finite-sample validity for its p-values; the method-level test assumes the per-seed standardized excess is roughly normal across sixteen seeds; facts within a subject are dependent, so fact-level tests (the ε-rehearsal) are optimistic in unknown degree.
- **The certificate is concurrent, unpublished work (Anonymous, 2026).** It is frozen, vendored and shipped, and we report four defects in it (Appendix D.4) — but a frozen evaluator does not absolve its interpretation, which is why Section 5.2 does not use it.
- **Baselines and tuning.** NPO runs at its paper settings, and a 24-point sweep finds no configuration that passes relearning (Appendix B.9); MUDMAN received an eight-point grid on two seeds and no configuration met utility (Appendix B.13); RepSelect was run on two seeds only. A larger tuning budget for any baseline could narrow the gaps we report, and a utility-constrained MUDMAN is the comparison we could not produce.
- **Withdrawn claims.** Ten claims were withdrawn or narrowed during the work (Appendix B.7–B.8), including a 3B result set produced by a configuration error.

7 Conclusion

Training against a simulated relearning attack, anchored to a floor rather than to zero, brings post-attack recovery down to the twin's level: 1.17 nats below NPO on sixteen corpora, non-inferior to the twin at one retrain-to-retrain difference, replicated on sixteen fresh corpora and matched by a floor that needs no twin. It does not make a model indistinguishable from a twin — C1 and utility still separate them — and the part of the design that matters most, the continued push past the floor, is also the source of its over-suppression. The most promising lead is also the cheapest: keeping the normalization's adaptive denominator but freezing its numerator kept relearning non-inferior and cut retained-knowledge loss tenfold on eight seeds, which makes a schedule that stops at the floor, rather than past it, the natural next step. What remains open besides is utility above 0.5B and knowledge that was acquired in pretraining rather than fine-tuned in.

—

REPRODUCIBILITY STATEMENT

Every number in this paper is generated from the released run artifacts rather than transcribed: `scripts/make_tables.py` rebuilds Tables T1–T8 from `runs/`, `scripts/diag_noninferiority.py` produces Tables 1 and 3's right-direction columns, and the figures are produced by the four `scripts/fig_*.py` entry points and, for Figure 5, `scripts/diag_power.py`. Appendix C gives the exact driver command for every table, including the three environment variables that distinguish the method from the objective it is ablated against, and states that the authoritative record of what a run used is the `config` block written into its own `results.json`. The certification harness is vendored at a pinned commit and never modified; the sixteen protocol twins, their per-axis scores, the conformal nulls and the Holm decision are all regenerated by `scripts/m4_full_protocol.py` from those commands. Seeds are fixed throughout and the seed discipline is stated in Section 5.1; two early seeds were re-run under the final code and reproduce their stored metrics to under 1e-9. Appendix B is a ledger of every configuration that did not work, and Appendix D holds the analyses moved out of the main text for length.

ETHICS STATEMENT

This work studies how to remove specific knowledge from a language model so that it cannot be recovered, and how to verify that removal under attack. The intended use is compliance with deletion requests and the removal of hazardous capability; the symmetric risk is that a method which *appears* to remove knowledge without doing so provides false assurance to a data subject or a deployer. Our central finding — that suppression-style baselines pass evaluation-time criteria while remaining recoverable by a few steps of fine-tuning, and that a pass of a finite attack battery bounds nothing — is a statement about that risk, and we report the axes our own method fails as prominently as those it passes. The experiments use a synthetic entity-QA corpus and two public benchmarks of fictitious or public-news facts; no personal data and no human subjects are involved. The attack code in this work is the evaluation battery of a certification protocol and reproduces published attacks rather than introducing new ones. We identify four defects in the certification protocol we build on and report them upstream rather than exploiting them.

AI USE STATEMENT

Following the ICLR policy's categories. **Required-disclosure uses that occurred.** Generative AI tools (Claude, via Claude Code) were used to implement methods (the CINDER objective, drivers, protocol scripts and diagnostics); to design and give feedback on experiments, including the preregistered seed expansions and the forget-fraction run; to propose and refine hypotheses, several of which the same tools later refuted (Appendix B.7); to interpret results; and to develop parts of the statistical framework and formulate the claims that rest on it — specifically the method-level non-inferiority test of Section 5.2 (its margin and intersection–union structure), the plug-in power model and twin-level bootstrap of Appendix D.3, the disjoint-pair observation, and the pooled-null re-judging of Section 5.3; and to port MUDMAN from its authors' code. Those constructions were checked and are owned by the authors, but they were not authored independently of AI and we say so. **Required-disclosure uses that did not occur.** The paper states and proves no theorem, so no proof was written with or without AI. No synthetic data set was generated by AI: the corpus is produced by a seeded, deterministic generator in the vendored harness, and the external arms use public TOFU and MUSE splits unchanged. No dataset was cleaned or reformatted by AI; no translation or qualitative analysis was involved. **Recommended disclosure.** AI tools drafted and revised parts of this manuscript, created the figures, wrote and edited code, and searched and summarized related work. **Human responsibility and verification.** Every experimental decision, preregistration, hyperparameter choice and claim was made and is owned by the authors. All code was reviewed and is covered by the released test suite; every reported number is regenerated from run artifacts by released scripts; every cited reference was retrieved from its own record. The released experiment ledger documents the workflow in full, including two configuration errors the AI-assisted workflow introduced — one of which produced an entire set of 3B results before our audit caught it (Appendix B.8) — and the guards and regression tests added in response. The authors take responsibility for the final content of this work.

REFERENCES

Anonymous. Anonymized certificate paper, under review, 2026. Concurrent work; reference withheld for double-blind review.

Georgina Cosma and Axel Finke. RULER: Representation-level verification of machine unlearning. arXiv preprint arXiv:2605.27569v3, 2026.

Chongyu Fan, Jinghan Jia, Yihua Zhang, Anil Ramakrishna, Mingyi Hong, and Sijia Liu. Towards LLM unlearning resilient to relearning attacks: A sharpness-aware minimization perspective and beyond. In *International Conference on Machine Learning (ICML)*, 2025.

Hongcheng Gao, Tianyu Pang, Chao Du, Taihang Hu, Zhijie Deng, and Min Lin. Meta-unlearning on diffusion models: Preventing relearning unlearned concepts. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2131–2141. IEEE, 2025. doi: 10.1109/ICCV51701.2025.00206.

Anmol Goel, Alan Ritter, and Iryna Gurevych. Auditing language model unlearning via information decomposition. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 808–826, Rabat, Morocco, 2026.

Phillip Guo, Aaquib Syed, Abhay Sheshadri, Aidan Ewart, and Gintare Karolina Dziugaite. Mechanistic unlearning: Robust knowledge unlearning and editing via mechanistic localization. In *International Conference on Machine Learning (ICML)*, 2025.

SeungBum Ha, Saerom Park, and Sung Whan Yoon. Unlearning's blind spots: Over-unlearning and prototypical relearning attack. In *International Conference on Machine Learning (ICML)*, 2026.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew B. Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Adam Khoja, Zhenqi Zhao, Ariel Herbert-Voss, Cort B. Breuer, Samuel Marks, Oam Patel, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam A. Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Russell Kaplan, Ian Steneker, David Campbell, Brad Jokubaitis, Alex Levinson, Jean Wang, William Qian, Kallol Krishna Karmakar, Steven Basart, Stephen Fitz, Mindy Levine, Ponnurangam Kumaraguru, Uday Tupakula, Vijay Varadharajan, Ruoyu Wang, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The WMDP benchmark: Measuring and reducing malicious use with unlearning. In *International Conference on Machine Learning (ICML)*, 2024.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. TOFU: A task of fictitious unlearning for LLMs. In *Conference on Language Modeling (COLM)*, 2024.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A. Smith, and Chiyuan Zhang. MUSE: Machine unlearning six-way evaluation for language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Naman Deep Singh, Maximilian Müller, Francesco Croce, and Matthias Hein. Unlearning that lasts: Utility-preserving, robust, and almost irreversible forgetting in LLMs. arXiv preprint arXiv:2509.02820v1, 2025.

Filip Sondej and Yushi Yang. Collapse of irrelevant representations (CIR) ensures robust and non-disruptive LLM unlearning. arXiv preprint arXiv:2509.11816v2, 2025.

Filip Sondej, Yushi Yang, Mikołaj Kniejski, and Marcel Windys. Robust LLM unlearning with MUDMAN: Meta-unlearning with disruption masking and normalization. arXiv preprint arXiv:2506.12484v5, 2025.

Filip Sondej, Yushi Yang, and Adam Mahdi. RepSelect: Robust LLM unlearning via representation selectivity. arXiv preprint arXiv:2606.17168v3, 2026.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. Tamper-resistant safeguards for open-weight LLMs. In *International Conference on Learning Representations (ICLR)*, 2025.

Pratiksha Thaker, Shengyuan Hu, Neil Kale, Yash Maurya, Zhiwei Steven Wu, and Virginia Smith. Position: LLM unlearning benchmarks are weak measures of progress. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 520–533. IEEE, 2025. doi: 10.1109/SaTML64287.2025.00035.

Changsheng Wang, Yihua Zhang, Jinghan Jia, Parikshit Ram, Dennis Wei, Yuguang Yao, Soumyadeep Pal, Nathalie Baracaldo, and Sijia Liu. Invariance makes LLM unlearning resilient even to unanticipated downstream fine-tuning. In *International Conference on Machine Learning (ICML)*, 2025.

Zeguan Xiao, Xuanzhe Xu, Yun Chen, Yong Wang, Jian Yang, Yanqing Hu, and Guanhua Chen. Robust LLM unlearning against relearning attacks: The minor components in representations matter. arXiv preprint arXiv:2605.11685v1, 2026.

Xiaoyu Xu, Xiang Yue, Yang Liu, Qingqing Ye, Huadi Zheng, Peizhao Hu, Minxin Du, and Haibo Hu. Unlearning isn't deletion: Investigating reversibility of machine unlearning in LLMs. In *International Conference on Machine Learning (ICML)*, 2026.

Lulu Xue, Shengshan Hu, Wei Lu, Yan Shen, Dongxu Li, Peijin Guo, Ziqi Zhou, Minghui Li, Yanjun Zhang, and Leo Yu Zhang. Towards reliable forgetting: A survey on machine unlearning verification. *ACM Computing Surveys*, 58(12):1–35, 2026. doi: 10.1145/3807451.

Han Yan, Zheyuan Liu, and Meng Jiang. Dual-space smoothness for robust and balanced LLM unlearning. In *International Conference on Learning Representations (ICLR)*, 2026.

Qingjie Zhang, Haoting Qian, Zhicong Huang, Cheng Hong, Minlie Huang, Ke Xu, Chao Zhang, and Han Qiu. Understanding the dilemma of unlearning for large language models. arXiv preprint arXiv:2509.24675v1, 2025a.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *Conference on Language Modeling (COLM)*, 2024.

Zhiwei Zhang, Fali Wang, Xiaomin Li, Zongyu Wu, Xianfeng Tang, Hui Liu, Qi He, Wenpeng Yin, and Suhang Wang. Catastrophic failure of LLM unlearning via quantization. In *International Conference on Learning Representations (ICLR)*, 2025b.

A GENERATED TABLES

What is in the appendices. A is the table pack: every per-seed number the main text summarises, generated from `runs/` by `scripts/make_tables.py` rather than transcribed, so the counts in Section 5.3 can be recounted from the rows. B is the negative-result ledger: every configuration that did not work, why we ran it, and what its failure rules out — the design laws are inferences from these runs, so the ledger is the evidence for Section 4.2 and not an afterthought. C gives the exact command behind each table and figure. D holds analyses that were main-text paragraphs in an earlier draft and were moved here for length; the main text keeps a pointer to each. **The table pack.** 4 is the fixed-margin rehearsal every baseline family fails. 5 is the headline: sixteen corpus seeds × eight axes × three methods under the frozen protocol, with 6 the preregistered retain-guard ablation beside it. 8 is the RepSelect head-to-head, 11 the held-out attack sweep, 12 the component-necessity ladder in its judged form, 13 the 1.5B scale row and 15 where each passing statistic sits inside its own twin–twin null. A blank cell is a run we did not make, never a run we dropped.

Table 4: Baseline certificate rehearsal (ε-test, all methods fail)

model	certified	forget EM	retain EM	failing axes
full	✗	0.98	0.98	relearn, quant4, paraphrase, incontext
ga	✗	0.00	0.94	relearn
npo	✗	0.19	1.00	relearn
rmu	✗	0.97	0.99	relearn, quant4, paraphrase, incontext
localized	✗	0.97	0.98	relearn, quant4, paraphrase, incontext
hide	✗	0.01	0.97	relearn

Qwen2.5-1.5B (ε=0.5)

model	certified	forget EM	retain EM	failing axes
full	✗	0.99	0.98	relearn, quant4, paraphrase, incontext
ga	✗	0.01	0.82	relearn, utility
npo	✗	0.71	1.00	relearn, quant4, paraphrase, incontext
rmu	✗	0.93	0.97	relearn, quant4, paraphrase, incontext
localized	✗	0.93	0.98	relearn, quant4, paraphrase, incontext
hide	✗	0.00	1.00	relearn

Per-method summary (certified = every axis passes Holm at α=0.05):
Axes leaked, count over the same seeds:
Per-method summary (certified = every axis passes Holm at α=0.05):
Axes leaked, count over the same seeds:
Per-method summary (certified = every axis passes Holm at α=0.05):
Axes leaked, count over the same seeds:
Per-method summary (certified = every axis passes Holm at α=0.05):
Axes leaked, count over the same seeds:

B NEGATIVE-RESULT LEDGER

Every configuration we ran that did **not** work, and what each one rules out. The chronological source is `runs/LOG.md`; this is the distilled version.

One name is used throughout, because the released code uses it. *v0* is the driver's default objective: a raw log-probability penalty with no twin-floor hinge, a LoRA inner attack instead of a full-parameter one, and no fact-level split. It is row 1 of the ladder below, it never certifies, and it is the configuration that silently produced a whole set of 3B results we had to withdraw (B.8). It keeps its name here because `CINDER_ALLOW_V0=1` and `tests/test_recipe_guard.py` refer to it by that name; elsewhere the method is simply CINDER, and the preregistered variant that adds a retain guard is CINDER + retain guard.

We report the ledger in full because the design choices of Section 4.2 were inferred from these failures — mostly on one development seed, which is why Section 5.5 re-tests the two that matter over sixteen

Table 5: Full preregistered protocol (16-twin conformal + Holm, α=0.05)

seed	cinder	npo	full
43	✓ —	✗ quant, paraphrase, c1	✗ quant, paraphrase, c1, c2, relearn
44	✓ —	✗ relearn	✗ relearn, quant, paraphrase, incontext, c1, uma
45 *(tuning)*	✗ c1	✗ relearn, quant, paraphrase, incontext, c1	✗ relearn, quant, paraphrase, incontext, c1, c2, uma
46	✗ quant, c1, utility	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, c1
47	✗ c1, utility	✗ relearn, c1	✗ relearn, quant, paraphrase, incontext, uma, c1
48	✗ relearn, c1	✗ relearn	✗ relearn, quant, paraphrase, incontext, c1
49	✗ utility	✗ relearn, quant, paraphrase, c1, incontext	✗ relearn, quant, paraphrase, incontext, c1
50	✗ c1, utility	✗ relearn, quant, paraphrase, incontext, c1	✗ relearn, quant, paraphrase, incontext, uma, c1
51	✗ c1, utility	✗ relearn, quant, paraphrase, incontext, c1	✗ relearn, quant, paraphrase, incontext, c1
52	✓ —	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, c1, uma
53	✓ —	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, c1, c2
54	✓ —	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, uma, c1
55	✓ —	✓ —	✗ relearn, quant, paraphrase, incontext, uma, c1
56	✓ —	✗ relearn, quant, paraphrase, c1, incontext	✗ relearn, quant, paraphrase, incontext, c1
57	✗ c1	✗ relearn, incontext, c1	✗ relearn, quant, paraphrase, incontext, c1
58	✓ —	✓ —	✗ relearn, quant, paraphrase, incontext, uma, c1

method	fully certified	passes relearn	mean leaky axes / seed
cinder	8/16 (43, 44, 52, 53, 54, 55, 56, 58)	15/16	0.88
npo	2/16 (55, 58)	3/16	3.19
full	0/16	0/16	5.62

method	relearn	quant	paraphrase	incontext	uma	c1	c2	utility
cinder	1	1	0	0	0	7	0	5
npo	13	10	10	6	0	12	0	0
full	16	16	16	15	8	16	3	0

Table 6: Retain-guard ablation (k=5) — beside T2, same twins

seed	cinder	full
43	✓ —	✗ quant, paraphrase, c1, c2, relearn
44	✓ —	✗ relearn, quant, paraphrase, incontext, c1, uma
45 *(tuning)*	✗ c1	✗ relearn, quant, paraphrase, incontext, c1, c2, uma
46	✗ quant, c1, utility	✗ relearn, quant, paraphrase, incontext, c1
47	✗ relearn, c1	✗ relearn, quant, paraphrase, incontext, uma, c1
48	✓ —	✗ relearn, quant, paraphrase, incontext, c1
49	✗ c1, utility	✗ relearn, quant, paraphrase, incontext, c1
50	✗ c1	✗ relearn, quant, paraphrase, incontext, uma, c1
51	✗ c1	✗ relearn, quant, paraphrase, incontext, c1
52	✓ —	✗ relearn, quant, paraphrase, incontext, c1, uma
53	✓ —	✗ relearn, quant, paraphrase, incontext, c1, c2
54	✓ —	✗ relearn, quant, paraphrase, incontext, uma, c1
55	✓ —	✗ relearn, quant, paraphrase, incontext, uma, c1
56	✓ —	✗ relearn, quant, paraphrase, incontext, c1
57	✗ c1	✗ relearn, quant, paraphrase, incontext, c1
58	✓ —	✗ relearn, quant, paraphrase, incontext, uma, c1

method	fully certified	passes relearn	mean leaky axes / seed
cinder	9/16 (43, 44, 48, 52, 53, 54, 55, 56, 58)	15/16	0.69
full	0/16	0/16	5.62

method	relearn	quant	paraphrase	incontext	uma	c1	c2	utility
cinder	1	1	0	0	0	7	0	2
full	16	16	16	15	8	16	3	0

Table 7: Fixed-coefficient hinge (Appendix B.12) — beside T2, same twins

seed	cinder	full
43	✗ c1	✗ quant, paraphrase, c1, c2, relearn
44	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, c1, uma
45 *(tuning)*	✗ relearn, quant, paraphrase, c1, incontext	✗ relearn, quant, paraphrase, incontext, c1, c2, uma
46	✗ quant, paraphrase, incontext, c1, utility	✗ relearn, quant, paraphrase, incontext, c1
47	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, uma, c1
48	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, c1
49	✗ quant, paraphrase, c1, utility	✗ relearn, quant, paraphrase, incontext, c1
50	✗ c1	✗ relearn, quant, paraphrase, incontext, uma, c1
51	✗ c1	✗ relearn, quant, paraphrase, incontext, c1
52	✗ c1	✗ relearn, quant, paraphrase, incontext, c1, uma
53	✗ relearn, quant, paraphrase, c1	✗ relearn, quant, paraphrase, incontext, c1, c2
54	✗ quant, paraphrase, c1, relearn	✗ relearn, quant, paraphrase, incontext, uma, c1
55	✓ —	✗ relearn, quant, paraphrase, incontext, uma, c1
56	✗ c1	✗ relearn, quant, paraphrase, incontext, c1
57	✗ quant, paraphrase, c1, utility	✗ relearn, quant, paraphrase, incontext, c1
58	✗ quant, uma, c1	✗ relearn, quant, paraphrase, incontext, uma, c1

method	fully certified	passes relearn	mean leaky axes / seed
cinder	1/16 (55)	10/16	2.88
full	0/16	0/16	5.62

method	relearn	quant	paraphrase	incontext	uma	c1	c2	utility
cinder	6	10	9	2	1	15	0	3
full	16	16	16	15	8	16	3	0

Table 8: RepSelect head-to-head (same twins, same nulls)

seed	cinder	npo	repselect	full
43	✓ —	✗ quant, paraphrase, c1	✗ quant, paraphrase, c1	✗ quant, paraphrase, c1, c2, relearn
44	✓ —	✗ relearn	✗ relearn, quant, paraphrase, c1, incontext	✗ relearn, quant, paraphrase, incontext, c1, uma

method	fully certified	passes relearn	mean leaky axes / seed
cinder	2/2 (43, 44)	2/2	0.00
npo	0/2	1/2	2.00
repselect	0/2	1/2	4.00
full	0/2	0/2	5.50

method	relearn	quant	paraphrase	incontext	uma	c1	c2	utility
cinder	0	0	0	0	0	0	0	0
npo	1	1	1	0	0	1	0	0
repselect	1	2	2	1	0	2	0	0
full	2	2	2	1	1	2	1	0

Table 9: Every baseline family under the full protocol

method	seeds	certified	passes relearn	leaky axes / seed	most-rejected axes
CINDER	16	8/16	15/16	0.88	c1 7, utility 5, quant 1
NPO	16	2/16	3/16	3.19	relearn 13, c1 12, quant 10
GA (GradDiff)	16	0/16	3/16	2.38	utility 13, relearn 13, c1 7
suppression	16	2/16	4/16	1.62	relearn 12, c1 9, utility 4
RMU	16	0/16	0/16	4.44	relearn 16, c1 11, quant 10
localization	16	0/16	0/16	5.56	quant 16, paraphrase 16, c1 16
untouched	16	0/16	0/16	5.62	quant 16, paraphrase 16, c1 16

Table 10: Method-level non-inferiority to the twin (Section 5.2)

condition	method	n	all axes not rejected	relearn not rejected	relearn excess, nats [UCB]	axes NI / 8	failing NI axes
0.5B headline (seeds 43-58)	cinder	16	8/16	15/16	+0.08 [+0.21]	6	c1, utility
0.5B headline (seeds 43-58)	full	16	0/16	0/16	+1.70 [+1.74]	1	relearn, quant, paraphrase, incontext, uma, c1, c2
0.5B headline (seeds 43-58)	npo	16	2/16	3/16	+1.24 [+1.37]	2	relearn, quant, paraphrase, incontext, c1, c2
0.5B fresh (seeds 59-74)	cinder	16	3/16	16/16	+0.16 [+0.29]	6	c1, utility
0.5B fresh (seeds 59-74)	full	16	0/16	0/16	+1.72 [+1.76]	1	relearn, quant, paraphrase, incontext, uma, c1, c2
0.5B fresh (seeds 59-74)	npo	16	0/16	1/16	+1.32 [+1.42]	0	relearn, quant, paraphrase, incontext, uma, c1, c2, utility
0.5B floorsplit	cinder	16	7/16	16/16	-0.11 [+0.05]	6	c1, utility
0.5B basefloor	cinder	16	4/16	14/16	+0.11 [+0.32]	6	c1, utility
0.5B relufixed	cinder	7	0/7	4/7	+0.74 [+0.88]	2	relearn, quant, paraphrase, incontext, c1, utility
0.5B mudman	mudman	16	5/16	14/16	+0.27 [+0.39]	6	relearn, utility
0.5B mudmantwin	mudman_twin	16	5/16	14/16	+0.18 [+0.36]	6	c1, utility
0.5B numfrozen	cinder	8	4/8	8/8	-0.06 [+0.22]	5	c1, c2, utility
0.5B numcapped	cinder	8	2/8	6/8	+0.15 [+0.47]	4	relearn, c1, c2, utility
0.5B floorshiftm5	cinder	8	2/8	8/8	-0.14 [+0.10]	6	c1, utility
0.5B floorshiftp5	cinder	8	3/8	6/8	+0.26 [+0.48]	4	relearn, c1, c2, utility
0.5B mudguard	mudman	8	0/8	0/8	+1.62 [+1.74]	0	relearn, quant, paraphrase, incontext, uma, c1, c2, utility
0.5B mudtwinguard	mudman_twin	8	0/8	0/8	+1.68 [+1.76]	1	relearn, quant, paraphrase, incontext, uma, c1, c2
0.5B fixedstats	cinder	16	1/16	10/16	+0.78 [+0.87]	2	relearn, quant, paraphrase, incontext, c1, utility
0.5B nofloor	cinder	8	1/8	3/8	+0.80 [+0.94]	2	relearn, quant, paraphrase, c1, c2, utility
0.5B lora	cinder	8	2/8	6/8	+0.44 [+0.60]	2	relearn, quant, paraphrase, c1, c2, utility
0.5B nosplit	cinder	8	1/8	8/8	+0.29 [+0.42]	4	relearn, c1, c2, utility
0.5B lora_nofloor	cinder	8	0/8	3/8	+0.94 [+1.03]	1	relearn, quant, paraphrase, incontext, c1, c2, utility
0.5B nosplit_nofloor	cinder	8	0/8	4/8	+0.67 [+0.79]	1	relearn, quant, paraphrase, incontext, c1, c2, utility
0.5B lora_nosplit	cinder	8	2/8	6/8	+0.35 [+0.67]	2	relearn, quant, paraphrase, c1, c2, utility
0.5B lora_nosplit_nofloor	cinder	8	0/8	4/8	+0.87 [+1.03]	2	relearn, quant, paraphrase, incontext, c1, utility

Table 11: Held-out attack sweep (outside the grid AND the training sim)

attack config	cinder	npo
`ep20_lr2e-5_f50`	pass (-0.19)	**FAIL** (+1.72)
`ep6_lr1e-4_f50`	pass (-0.17)	**FAIL** (+0.36)
`ep6_lr2e-5_f25`	pass (-0.81)	pass (-0.12)
`ep6_lr2e-5_f75`	pass (-0.42)	**FAIL** (+1.18)
`ep6_lr2e-5_f50_seed123`	pass (-0.82)	**FAIL** (+0.92)
`lora_r16_s100_lr1e-3`	pass (-0.08)	pass (+0.14)

Table 12: Component necessity (RQ4, seed 44 rehearsal)

variant	relearn mean diff	p	certified
full-FT inner, no hinge	+1.240	0.955	✗
hinge only (LoRA inner)	+0.870	0.640	✗
both (CINDER)	+0.053	0.037	✓

Table 13: Qwen2.5-1.5B under the full protocol (per-axis conformal p)

seed	method	relearn	quant	paraphrase	incontext	uma	c1	c2	utility	leaky
44	cinder	0.602	0.884	0.979	0.871	0.772	0.004	0.618	0.004	c1, utility
44	full	0.004	0.004	0.004	0.004	0.004	0.004	0.531	0.004	relearn, quant, paraphrase, incontext, uma, c1, utility
45	cinder	0.685	1.000	1.000	0.801	0.871	0.004	0.166	0.004	c1, utility
45	full	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.676	relearn, quant, paraphrase, incontext, uma, c1, c2
46	cinder	0.411	0.166	0.826	0.909	0.780	0.004	0.950	0.004	c1, utility
46	full	0.004	0.004	0.004	0.004	0.199	0.004	0.004	0.448	relearn, quant, paraphrase, incontext, c1, c2
46	npo	0.004	0.004	0.004	0.012	0.780	0.004	0.008	0.224	relearn, quant, paraphrase, c1, c2, incontext
47	cinder	0.369	0.643	0.954	0.382	0.672	0.004	0.992	0.004	c1, utility
47	full	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.357	relearn, quant, paraphrase, incontext, uma, c1, c2
47	npo	0.004	0.004	0.004	0.004	0.017	0.004	0.004	0.178	relearn, quant, paraphrase, incontext, c1, c2, uma
48	cinder	0.502	1.000	1.000	1.000	0.971	0.046	0.801	0.004	utility
48	full	0.004	0.004	0.004	0.008	0.593	0.004	0.328	0.004	relearn, quant, paraphrase, c1, utility, incontext
48	npo	0.004	0.353	0.755	1.000	0.983	0.004	0.145	0.776	relearn, c1
49	cinder	1.000	1.000	1.000	1.000	0.714	0.004	0.245	0.004	c1, utility
49	full	0.004	0.004	0.004	0.004	0.066	0.004	0.029	0.739	relearn, quant, paraphrase, incontext, c1
49	npo	0.004	0.004	0.004	0.004	0.071	0.004	0.004	0.178	relearn, quant, paraphrase, incontext, c1, c2

Table 14: Qwen2.5-3B under the full protocol (per-axis conformal p)

seed	method	relearn	quant	paraphrase	incontext	uma	c1	c2	utility	leaky
44	cinder	0.133	1.000	1.000	1.000	0.876	0.037	0.610	0.120	—
44	full	0.004	0.004	0.004	0.008	0.004	0.004	0.531	0.178	relearn, quant, paraphrase, uma, c1, incontext
45	cinder	0.635	0.772	0.714	0.378	0.934	0.004	0.921	0.004	c1, utility
45	full	0.004	0.004	0.004	0.656	0.427	0.004	0.651	0.004	relearn, quant, paraphrase, c1, utility
46	cinder	0.515	0.776	0.921	0.934	1.000	0.004	0.423	0.112	c1
46	full	0.004	0.004	0.004	0.004	0.004	0.004	0.519	0.307	relearn, quant, paraphrase, incontext, uma, c1
46	npo	0.004	0.004	0.137	0.017	0.373	0.004	0.535	0.228	relearn, quant, c1
47	cinder	0.037	0.979	1.000	0.838	0.639	0.481	0.751	0.004	utility
47	full	0.004	0.004	0.004	0.008	0.004	0.004	0.004	0.925	relearn, quant, paraphrase, uma, c1, c2, incontext
47	npo	0.004	0.195	0.129	0.876	0.071	0.004	0.004	0.884	relearn, c1, c2
48	cinder	0.635	1.000	1.000	1.000	0.801	0.494	0.739	0.004	utility
48	full	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.461	relearn, quant, paraphrase, incontext, uma, c1, c2
48	npo	0.029	0.021	0.137	0.656	0.598	0.004	0.021	0.004	c1, utility
49	cinder	0.967	1.000	1.000	0.963	0.257	0.274	0.328	0.166	—
49	full	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.398	relearn, quant, paraphrase, incontext, uma, c1, c2
49	npo	0.004	0.004	0.004	0.004	0.012	0.004	0.004	0.178	relearn, quant, paraphrase, incontext, c1, c2, uma

Table 15: Where each result sits inside the twin-twin null

method	relearn	quant	paraphrase	incontext	uma	c1	c2	utility
cinder	61	0	0	0	43	95	53	94
npo	100	100	100	98	48	100	80	88
full	100	100	100	100	99	100	92	56

Table 16: Minimum detectable effect per axis (what a pass does NOT rule out)

axis	MDE (median)	range
relearn	0.878	0.623 – 1.349
quant	0.663	0.462 – 1.120
paraphrase	0.678	0.455 – 1.189
incontext	1.226	0.656 – 1.821
uma	0.178	0.111 – 0.235
c1	1.323	0.843 – 1.716
c2	0.115	0.096 – 0.200
utility	0.056	0.001 – 0.312

— and because a reader cannot judge a "designed-to-pass" method without seeing what the design rejected.

B.1 Objective shape — why the penalty is a twin-calibrated hinge

The objective was built one decision at a time, and each row below is a run that was judged before the next was written. **How to read it.** *Relearn margin* is the held-out forget log-probability after the certificate's relearning attack, minus the retain-only twin's under the same attack: 0 means the unlearned model recovers exactly as much as a model that never saw the data, and positive means it recovers more, i.e. the knowledge was still there. The certificate's relearn axis passes only when the margin is inside twin–twin variation, which at this seed is about +0.1; ✗ marks a run that failed it. The ladder is the evidence for design choices 1 and 2 of Section 4.2 (re-tested over eight seeds per cell in Section 5.5), and it is reported in full, including the four rows that pushed the wrong way.

Table 17: Objective-shape ladder, 0.5B corpus seed 44. Each row is one training run judged once under the relearning attack; the margin is nats above the retain-only twin's post-attack recovery, so lower is better and the hinge target is zero.

#	Variant	Result (relearn margin, 0.5B seed 44)	Rules out
1	Raw log-prob penalty, weight 0.3 (v0)	+0.88 ✗	— (baseline for the ladder)
2	Same, weight 0.6, K=5, every step	+1.09 ✗	"more pressure helps"
3	Same, weight 1.0	+1.18 ✗	same, harder
4	Weight 0.6 + forget-ascent 0.5, 10 epochs	+1.20 ✗	ascent as a booster
5	Twin-floor hinge + forget-ascent 0.3	+1.04 ✗	ascent alongside the hinge
6	Twin-floor hinge, meta-only	+0.87 ✗	hinge alone suffices
7	Full-FT inner attack, **no** hinge	+1.24 ✗	attack-class alone suffices
8	**Hinge + full-FT inner + fact split**	**+0.05** ✓	— (the method)

Rows 2–4 are the counter-intuitive core: increasing the pressure on the forget set monotonically *worsened* resistance to relearning. Deeper suppression digs a hole that a few attack steps refill faster — the training-time mirror of what we observe when the MCU method is run against this protocol (Section 2), the observation that motivated the project. Rows 6–8 isolate the interaction: neither the hinge nor the matching attack class certifies alone.

B.2 Attack-class asymmetry — both directions

Design choice 2 says the simulated attack must match the threat model's parameter class. This table shows the claim is symmetric: simulating a weaker class than the attacker uses fails, and *defending* in a weaker class than the attacker uses also fails. The second direction is what drives the memory cost of Section 5.6: in our runs an adapter-only defence never lowered the penalty.

Table 18: Attack-class asymmetry at 0.5B seed 44. Rows vary the parameter class of the inner attack and of the outer (defence) update independently; only matching the attacker's class in both places certifies.

Variant	Result	Reading
LoRA-subspace inner attack	plateau $\approx$ +0.63, never certifies	Resistance to adapter-space attacks does not transfer to full fine-tuning.
LoRA-only **outer** loop (adapters trained, base frozen)	meta penalty flat across all 400 steps at both 1e-4 and 1e-3; forget EM stays $\geq$0.99	Adapter-space *defenses* cannot satisfy the hinge against full-parameter attacks.

The two rows together give the bidirectional statement in Section 4.2; they are single runs with one optimizer and one LoRA configuration each, not a proof that no adapter defence can work.

B.3 Memory-reduction routes — five failures, one conclusion

Each was an attempt to fit the certified recipe above 0.5B on a 32 GB card. All five preserve the objective and change only the numerics, so each is a direct test of whether full-parameter fp32

AdamW is actually required or merely convenient. All five lose the relearning pass. **Tuning range: none.** Each route was run once, on seed 44, at the recipe's learning rates with the substitute's defaults (Adafactor with relative step and parameter scaling off; SGD at lr 5e-3, its only setting), and closed on that single failure (LOG 2026-07-22). The table therefore says these routes did not work as drop-in replacements, not that they cannot be made to work.

Table 19: Five memory-reduction routes, 0.5B seed 44. Every route keeps the objective and the attack class and changes only precision or optimizer state; none retains the relearning certification.

Route	Outcome
Pure bf16 training (weights+grads+Adam)	forget EM 0.63 vs 0.09 in fp32 — 2e-5-scale updates underflow; certificate lost
SGD inner attack	diverges (twin floor estimate -2.4×10^{6} at 0.5B; -6.4×10^{3} at 1.5B)
Adafactor, both loops	forget EM 0.23, relearn axis fails
Adafactor, inner only	forget EM 0.12, relearn axis fails
8-bit AdamW (bitsandbytes), both loops	numerical explosion (log-prob $\approx -3\times10^{11}$)

Conclusion: in our implementation the recipe needed fp32 AdamW in **both** loops, roughly 4× model memory plus a full inner attack — the 28 B/param of Section 5.6, and the reason unsharded runs cap at 0.5B on a 32 GB card.

B.4 Curriculum terms that did not earn their place

- **λ2, adversarial non-decodability.** Added after C2 was hypothesised to bind. It does not: CINDER alone already sits at the twin's C2 floor (worst-case probe accuracy 0.167 vs the twin's 0.233, chance 0.083), and adding λ2 pushed the model *below* the twin floor — over-suppression despite the hinge — while degrading relearn resistance on both tested seeds (+0.05 → +0.53; −0.49 → +0.39). Dropped.
- **λ3, circuit repair.** This one *is* earned by the protocol (C1 leaks on 4/6 seeds) but cannot be paid for: four settings — weight 0.3, weight 0.1 every 4 steps, weight 0.3 with the corrupt branch detached, and weight 0.05 on the late layer only — all repaired C1 (p 0.004 → 0.012–0.31) and all exceeded the conformal utility envelope (p = 0.004; retain-damage statistic 0.023 against a twin-null maximum of 0.014). Reported as the knife-edge, not as a component.

B.5 Cost amortisation

`meta_every=4` reaches 5.8× NPO wall-clock but retain EM falls to 0.64; `K=1` reaches 131 s but retain EM 0.73. Both lose certification *and* utility. The certified configuration stays at 10.6× NPO. We report the overhead as an open problem rather than claiming a practical recipe.

B.6 Protocol-side findings (reported upstream, not used for our decisions)

- The certificate's own v1.1 general-text utility axis — defined in the harness but never used in the shipped protocol — rejects **every** method we tested, including NPO and the untouched control (p = 0.004). Its twin–twin null contains only seed variation and no benign method drift, so any unlearning at all exceeds it. All our decisions therefore use the frozen v1 protocol; we report this as a certificate-design note for the harness authors.

B.7 Claims we retracted during the work

Kept because they show which conclusions were fragile, and to what.

Fragile to seed count.

- *"All baselines fail the relearn axis (0/4)."* False: NPO survives Holm on seed 43 at p = 0.0498. Corrected to NPO 3/16 across the final sixteen seeds.
- *"Certified seeds concede retain EM; leaking seeds preserve it."* Inferred from four seeds, refuted by the sixth: seed 43 certifies at retain EM 0.990 while seed 47 fails utility at 0.905. No seed-level correlation survives.

- *"The 1.5B utility axis is degenerate — it rejects the untouched model too."* True on seed 44 and false on seed 45, where the untouched model passes at p = 0.676 while CINDER fails. The tight null was a property of one twin population, not of scale.
- *"Residual circuit position separates passing from leaking seeds cleanly."* An empty gap at six seeds; at sixteen the ranges interleave (seed 51 leaks at 0.180, below seed 49's passing 0.208). The association survives and strengthens by two orders of magnitude; the threshold form does not.
- *"Baselines never fully certify."* False: NPO certifies on 2 of 16 seeds (55, 58), both of them seeds CINDER also certifies. Stated in Section 5.3 rather than buried here, since it weakens our own headline.

Fragile to a configuration error of ours.

- *"The 1.5B recipe does not transfer to 3B."* Withdrawn: the run described as the recipe "applied verbatim" carried `forget_ascent_frac = 0.3` from a script default, where every certified run records 0.0 — a term Section 5.5 already shows to be harmful. The conclusion was drawn from an objective the paper does not propose. Caught by diffing the stored configs of the certified runs against the pilot's.
- *"The simulated attack's budget must grow with model size" (an earlier "design law 3", the 3B K sweep, its preregistered confirmation, and the resistance-horizon mechanism).* Withdrawn in full. Every 3B run from 2026-09-05 to 09-09 was launched with the driver's defaults for `CINDER_INNER_MODE` (LoRA, not full-parameter), `CINDER_META_TARGET` (no twin-floor hinge) and `CINDER_ATTACK_FRACTION` (no fact-level split) — the v0 objective of B.1, row 1. The launch scripts set only the variables they were varying; nothing failed, nothing warned; `results.json` did not record the three. Caught by an audit that compared Section 4 against the stored configs and the training logs (no `[floor]` line; a meta term diverging to −884 where the hinged runs sit at +10). Fixed by recording every setting, refusing those defaults without an explicit ablation flag, and rerunning 3B with the certified recipe (B.8). The previous "configuration error" entry above had caught one wrong setting in the same runs and missed these three.

Fragile to mistaking correlation for cause.

- *"3B fails relearning because the objective over-suppresses."* The pilot showed both at once and we read one as causing the other. Reducing the overshoot from 17.7× to 2.26× left the verdict bit-identical, at the conformal floor on both axes; four arms spanning 15× of suppression depth all failed the same way. B.8 records what the cause turned out to be.
- *"Over-suppression is itself a detectable tell on the decodability axis."* Predicted before looking; C2 returns p = 0.884 at a forget log-prob 17.7× below the twin floor. Design choice 1's cost is paid on the relearning axis alone, and we narrowed the choice's statement rather than keeping the broader one.

Understated rather than false.

- *"Certification costs ~4× model memory."* That figure covered the outer loop only (weights, gradients, AdamW = 16 B/param) with the inner attack mentioned separately, and read as the whole bill. The measured total is 28 B/param, 7× the fp32 weights.

The seed-count retractions were caught by regenerating tables from `runs/` programmatically (`scripts/make_tables.py`); the configuration error was caught by comparing stored configs, not prose. Both habits exist because prose is where stale numbers survive — twice in this project a corrected tally sat beside an uncorrected sentence for weeks.

B.8 THE 3B LAUNCHER ERROR, AND THE RERUN

What ran. `scripts/m2_cinder_v0.py` is configured by environment variables whose defaults are the v0 objective. The certified recipe existed only as a list in Appendix D.1. Five 3B launch scripts each set the four variables they were changing and inherited defaults for the rest — including the three that make the method CINDER. The resulting objective was: LoRA-subspace inner attack, raw mean log-prob penalty (no hinge), penalty evaluated on the attack set itself. Under it, "K" was the LoRA inner-step count, and the monotone relearn recovery we reported (p 0.004 → 0.021 → 0.228 for K = 3, 9, 15) was real — for that objective.

How it was caught. Not by any test or check that existed. An audit of methodology against implementation compared Section 4.1 with the stored configurations, found three of the ten defining

settings unrecorded, then found the training-log signature: the hinged runs at 0.5B and 1.5B log a meta term near +10 at step 0 (a softplus above the twin floor); every 3B run logged −3.9 at step 0 and diverged to between −33 and −884. No 3B log contains the `[floor]` line the driver prints when the hinge is on.

What changed. The driver now writes all thirteen effective settings into `results.json`, prints a `[recipe]` line, and exits before training if any of the three is at its v0 value unless `CINDER_ALLOW_V0=1` declares an ablation. A test pins both the guard and the hazardous defaults. The recipe lives in one file that every launch script sources.

The rerun. Certified recipe, K = 3, seed 44 as pilot, seed 45 frozen and judged once (preregistered 2026-09-10, with one declared deviation: the plan anticipated raising K, not choosing a learning rate).

Table 20: 3B learning-rate bracket after the launcher error was fixed, judged under the certified recipe. The window is bounded on both sides: the 1.5B rate destroys the model and the 0.5B rate under-forgets, and only the middle rate lands on the twin.

outer lr	seed	forget lp (twin −3.1)	retain EM	verdict
1e-4 (the 1.5B value)	44	−4.71	0.000	retain collapse
2e-5 (the 0.5B value)	44	−2.57	0.865	under-forgets; relearn at the floor
5e-5	**44**	**−6.73**	**0.975**	**certified** (C1 p = 0.037)
5e-5, frozen	45	−3.28	0.085	relearn passes (p = 0.63); retain collapse

The binding constraint transfers on both seeds; full certification on one. K = 3 is the certified value at every scale, and the "budget law" does not exist.

B.9 NPO IS NOT A STRAW MAN — A 24-CONFIGURATION SWEEP

The main text runs NPO at the harness default (lr 2e-5, β 0.1, six epochs) on every seed. On the tuning seed we swept lr $\in$ 1e-5, 2e-5, 5e-5, 1e-4 $\times$ $\beta \in 0.1, 0.5, 1.0 \times$ epochs $\in 6, 12$ through the ε-rehearsal (`scripts/local_npo_sweep.sh`).

0 of 24 pass the relearning axis. The bracket is design choice 1 in miniature:

region	example	forget lp (twin −2.6)	forget EM (twin 0.08)	retain EM
over-suppression	β 0.1, lr 1e-5	−12 to −15	0.05	0.99
near the twin	β 0.1, lr 1e-4, 12 ep	−3.2	0.12	0.88
under-forgetting	$\beta \geq 0.5$, lr $\leq$ 2e-5	−0.3 to −1.5	0.63–0.78	0.95–0.99
collapse	β 1.0, lr 1e-4, 6 ep	−1.3	0.12	0.50

No configuration lands on the twin and resists relearning at once; the ones that reach the twin's forget log-prob do so by damaging retain. The default is representative of the family. Per-run `results.json` in `runs/npo_sweep/`.

B.10 TOFU'S STANDARD METRICS

Setup of both external arms. TOFU (`locuslab/TOFU`): the first 8 authors of `forget10` as the forget set and the first 40 of `retain90` as the retain set, 20 QA pairs per author; MUSE-News knowmem: 100 forget and 100 retain QA pairs. Qwen2.5-0.5B, one seed (42), the headline recipe (MUSE with outer epochs raised to 40 to match step count), judged by the ε-rehearsal on relearn, quantization, in-context and utility (ε = 0.75; paraphrase and the circuit/probe axes need the synthetic schema). Result files: `runs/m4_tofu_seed42`, `runs/m4_muse_seed42_ep40`. The p-values below are TOFU's own KS statistic and, like every non-rejection in this paper, are not evidence of equivalence.

`scripts/tofu_standard.py` computes TOFU's truth ratio on the `forget10_perturbed` and `retain_perturbed` splits — length-normalised probability of the perturbed answers over the paraphrased one — and Forget Quality, the two-sample KS p-value of the forget-set truth-ratio distribution against the retain-only model's. (TOFU's Model Utility additionally aggregates ROUGE over three splits; we report the retain truth ratio and answer log-probability as the utility side.)

CINDER's median truth ratio is on the retain model's; its mean is inflated by a tail of questions pushed far below the paraphrased answer — the over-forgetting side. Its Forget Quality is the highest

model	forget truth ratio, median (mean)	retain truth ratio	retain lp / token	Forget Quality
retain-only	0.845 (1.04)	0.519	−0.021	—
CINDER	**0.850** (25.7)	0.479	−0.022	**0.045**
NPO	0.769 (0.94)	0.510	−0.032	0.010
fully trained	0.695 (0.85)	0.525	−0.022	5×10^{-5}

of the three and still below 0.05. NPO and the fully trained model are distinguishable from retain in the other direction.

B.11 Gradient ascent resists relearning by destroying the model

GA is the one baseline that passes the relearning axis about as often as NPO (3/16 against 3/16) while certifying on none, and the reason is worth stating because it is the clearest evidence in this paper that a single-axis comparison is not enough.

GA does not forget; it breaks. Its forget log-probability runs to **−67 to −167 nats** against the retain-only twin's ≈ −3, and retain exact match falls as low as **0.070**. The utility axis rejects it on **13 of 16 seeds**. On all three seeds where GA survives relearning (43, 53, 55) it leaks utility, and on the three seeds where its retain quality is highest (retain EM 0.915–0.945; seeds 44, 54, 56) it fails relearning instead. Judged on the relearning axis alone GA would look like a near-peer of CINDER; judged against the twin on every axis it is not an unlearning method at all.

This also bounds design choice 1 rather than contradicting it. Within a calibrated objective, raising the penalty monotonically *worsens* relearning resistance (B.1, rows 2–4). Unbounded ascent is outside that regime: it can resist by leaving nothing coherent to relearn, and pays for it where the certificate looks. The choice is about the calibrated family the method lives in, and we state it that way.

B.12 The λ rule cancels the hinge's tail — why the method overshoots its own floor

The objective's two hard-won rules interact in a way we did not intend, and the paper's over-suppression is the result. We found it in a self-audit before submission rather than from a reviewer, and state it here rather than in a footnote.

The mechanism. The penalty is `softplus(lp - floor)`, which is designed to vanish once the model's post-attack forget log-probability `lp` reaches the retain-only twin's floor. Every term is then scaled by the λ rule of Section 4.1, `coeff = frac * |L_retain| / (|term| + 1e-8)`, so that its contribution is a fixed fraction of the task loss whatever its raw scale. Below the floor the softplus tail is `exp(lp - floor)` and so is its derivative, so the coefficient grows at exactly the rate the penalty shrinks and the product does not vanish:

lp − floor	penalty	coefficient	gradient w.r.t. lp
+2.0	2.13	0.28	0.249
0.0	0.693	0.87	0.433
−3.0	0.0486	12.4	0.586
−6.0	0.00248	242	0.599
−10.0	4.5e−5	1.3e4	0.600

The gradient does not decay towards the floor; it *rises* to `frac * |L_retain|` = 0.6 and stays there. The hinge stops contributing value and never stops contributing force.

The signature in the runs. Across the sixteen headline 0.5B runs (seeds 43–58, the runs of Table 5) the forget log-probability sits **1.08 nats below** the twin on average and below it on **14 of 16**, with the extremes at −2.66 (seed 56) and +0.44 (seeds 46, 49). An earlier draft quoted −1.20 on 18 of 20 over a set that also counted development re-runs; the sixteen-seed figure is the one every other number here uses. At 1.5B the same gap is −0.4 to −1.4, and on the 5%-forget corpus of Section 5.4 it reaches −6.7. This is the same quantity D.4(iv) reports as a certificate defect. Both statements are true and they are about different things: the certificate cannot *see* over-suppression, and our objective *produces* it. We had previously reported only the first.

The experiment, run. We froze the λ-rule coefficient at its first-meta-step value so the hinge tail can vanish (`CINDER_META_COEFF=fixed`), preregistered it on 2026-09-23 with the prediction that

the overshoot would shrink toward zero, utility rejections would fall, and full certification would *not* exceed the headline's 8/16 because the penalty stops early — and ran all sixteen seeds against the same twins (Table T2d). All three predictions held, the last one by a wide margin:

	headline (normalized)	fixed coefficient
forget log-prob vs twin, mean	−**1.08 nats** (below on 14/16)	**+0.90 nats** (below on 1/16)
utility rejections	5/16	3/16
relearning passes	15/16	10/16
certified	8/16	**1/16**
leaked axes per seed	0.88	2.88
quantization / paraphrase leaks	1 / 0	10 / 9

Removing the tail-cancellation removes the overshoot — it flips sign, the model now stopping *above* the twin — and with it most of the resistance: seven of the eight certified seeds lose certification, and the quantization and paraphrase axes, which the headline never leaks, open on more than half the seeds. Utility does improve, as the mechanism predicts, but by two seeds.

What this settles. The continued push below the floor is not an accident to be engineered away; it is what carries the model from "at the twin" to "resistant to relearning", and it is also what produces the over-suppression of Section 5.4 and the 1.5B utility leaks. One mechanism, one benefit and one cost, now measured on both sides. Design choice 1 stands as a statement about the *anchor* — the ablation ladder of B.1 still shows a zero-anchored penalty is strictly worse — and the normalized hinge is the version of it that works. A coefficient schedule that lands *on* the floor rather than past or short of it is the obvious next experiment, and it is not in this paper.

B.13 The MUDMAN port, and how it was tuned

What is ported. `cinder/mudman.py` follows the authors' paper-era code (github.com/filyp/MUDMAN at commit 36241eb): one flat loop; a full copy of the intervened parameters as the adversary, re-forked from the model every 32 loops and otherwise persistent, taking one SGD step (lr 1e-3) per loop on forget cross-entropy; a momentum accumulator (0.9) of retain gradients that also steps the model (momentum SGD); the unlearning gradient taken at the adversary's weights from the same forward pass, before the adversary's own update; one global L2 normalization computed before masking; disruption masking against the sign of the retain accumulator; only `gate_proj` intervened; negative token-mean cross-entropy as the unlearning loss. **Where it differs, and why.** (i) The step is scaled by $\sqrt{\#\text{params}}/\|g\|$ as in their Optuna implementation (`surgical_irreversible_unlearning.py`), so the unlearning rate is a per-element RMS step; their minimal listing uses $1/\|g\|$, a reparametrization the tuning grid absorbs. (ii) No adversary decay (their final Algorithm 1 also drops it). (iii) The loop count is matched to CINDER's (six passes over the retain batches). (iv) Their retain-loss rule (skip the unlearning update while the retain loss exceeds its initial value by 0.05) is reported as a separate variant, *MUDMAN + guard*, because the first tuning round ran without it.

Tuning budget and selection, fixed before tuning (LOG 2026-09-23). Unlearning rate 3e-5, 1e-4, 3e-4, 1e-3 × retaining rate 1e-4, 1e-3 on development seeds 44 and 45, sixteen judged runs; select the most non-rejected axes summed over both seeds, ties to fewer utility rejections, then (written down before computing it) the higher mean relearn p. Selected: 1e-4 / 1e-3. **Every one of the eight configurations was rejected on utility on both seeds**; the lowest rate was also rejected on relearning on both. So no configuration in this budget met the utility criterion, and the comparison in Section 5.5 is against the best configuration by the stated rule, not against a utility-feasible MUDMAN; we did not find one.

MUDMAN with our floor. The same loop with CINDER's calibrated penalty, softplus(lp − f), as the unlearning loss at the adversary's weights. Its floor f comes from CINDER's own K = 3 full-parameter AdamW attack on the twin, not from MUDMAN's persistent SGD adversary, so it is a scalar transferred from another attack, not a floor matched to MUDMAN's; same hyperparameters as MUDMAN, no re-tuning.

C REPRODUCIBILITY

Every number in the paper comes from a run directory under `runs/`, and every table is regenerated from those directories by `scripts/make_tables.py` — none is transcribed by hand. This appendix gives the exact command for each.

C.1 ENVIRONMENT

```
uv sync --extra dev --extra gpu     # Python >=3.11; torch cu128 (Blackwell sm_120)
uv run pytest -m "not slow"         # 99 fast tests
uv run pytest -m slow               # 22 GPU tests (vendored harness)
uv run ruff check .
```

The certification harness is vendored **frozen** at `vendor/certif` (upstream commit recorded in `vendor/VERSION`) and is never modified; `vendor/tests_upstream` is its own test suite, run unchanged.

C.2 COMMON CONFIGURATION

The driver `scripts/m2_cinder_v0.py` is configured entirely by environment variables. The certified CINDER recipe is:

```
export PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True
export CINDER_META_TARGET=twin        # twin-calibrated hinge (never zero)
export CINDER_ASCENT_FRAC=0.0         # no forget-ascent term
export CINDER_ATTACK_FRACTION=0.5     # fact-level attack/eval split
export CINDER_INNER_MODE=full         # full-parameter inner attack
export CINDER_INNER_K=3               # inner steps
export CINDER_META_EVERY=2            # meta step cadence
```

C.3 TABLE 4 — BASELINE REHEARSAL

```
# 0.5B (eps=0.75)
CINDER_SEED=44 CINDER_OUT=runs/m1_baselines_seed44 \
  uv run python scripts/m1_baselines.py
# 1.5B (eps=0.5)
CINDER_MODEL=Qwen/Qwen2.5-1.5B CINDER_EPS=0.5 CINDER_AMP=1 CINDER_GRAD_CKPT=1 \
CINDER_BATCH=8 CINDER_SEED=44 CINDER_OUT=runs/m1_baselines_1.5b_seed44 \
  uv run python scripts/m1_baselines.py
```

C.4 TABLE 5 — FULL PREREGISTERED PROTOCOL (PER SEED)

Four phases; each is resumable and skips work already on disk.

```
SEED=43   # repeat for 44,45,46,47,48
# (a) train FULL + TWIN, unlearn with NPO and CINDER, save method checkpoints
CINDER_METHODS=npo,cinder CINDER_SAVE_METHODS=1 CINDER_SEED=$SEED \
CINDER_CKPT_DIR=runs/m2_ckpt_seed$SEED CINDER_OUT=runs/m4_ckpts_seed$SEED \
  uv run python scripts/m2_cinder_v0.py
# (b) build the 16 protocol twins   (c) score the per-axis nulls
for PHASE in build null; do
  CINDER_PHASE=$PHASE CINDER_SEED=$SEED CINDER_TWIN_DIR=runs/m4_twins_seed$SEED \
    uv run python scripts/m4_full_protocol.py
done
# (d) conformal p-values + Holm decision
CINDER_PHASE=judge CINDER_SEED=$SEED CINDER_TWIN_DIR=runs/m4_twins_seed$SEED \
CINDER_METHOD_DIR=runs/m4_ckpts_seed$SEED/models \
CINDER_FULL_CKPT=runs/m2_ckpt_seed$SEED/full.pt \
CINDER_OUT=runs/m4_protocol_seed$SEED \
  uv run python scripts/m4_full_protocol.py
```

Twin seeds are fixed at 70–85 (`CINDER_TWIN_SEEDS`), α=0.05 (`CINDER_ALPHA`).

C.5 TABLE 8 — REPSELECT HEAD-TO-HEAD

```
CINDER_SEED=$SEED uv run python scripts/m4_repselect.py     # lr grid + budget rule
# then re-run D.3(d); the judge picks up repselect.pt from the same method dir
```

C.6 TABLE 11 — HELD-OUT ATTACK SWEEP

```
CINDER_SEED=44 uv run python scripts/m4_heldout.py
```

C.7 External data arms

```
CINDER_DATASET=tofu CINDER_METHODS=npo,cinder CINDER_BATCH=8 CINDER_AMP=1 \
CINDER_SEED=42 CINDER_CKPT_DIR=runs/tofu_ckpt_seed42 CINDER_OUT=runs/m4_tofu_seed42 \
  uv run python scripts/m2_cinder_v0.py
CINDER_DATASET=muse CINDER_METHODS=npo,cinder CINDER_META_TARGET=twin \
CINDER_ASCENT_FRAC=0.0 CINDER_ATTACK_FRACTION=0.5 CINDER_INNER_MODE=full \
CINDER_META_EVERY=2 CINDER_INNER_K=3 CINDER_BATCH=4 CINDER_PENALTY_BATCHES=1 \
CINDER_AMP=1 CINDER_OUTER_EPOCHS=40 CINDER_SEED=42 \
CINDER_CKPT_DIR=runs/muse_ckpt_seed42 CINDER_OUT=runs/m4_muse_seed42_ep40 \
  uv run python scripts/m2_cinder_v0.py          # MUSE-News is the default subset
```

MUSE's retain split is ~9× smaller than the synthetic corpus, so the default epoch count yields far fewer outer steps; `CINDER_OUTER_EPOCHS` is raised to match step count rather than epoch count.

C.8 Diagnostics and figures

```
uv run python scripts/diag_circuit.py          # circuit-position measurement (Sec.5.3)
uv run python scripts/diag_statistics.py       # re-score ckpts -> statistics + nulls
uv run python scripts/diag_null_sensitivity.py # pooled-null re-judging (Sec.5.3)
uv run python scripts/diag_power.py            # power curves, MDE, Fig. 5 (D.3)
uv run python scripts/diag_twin_bootstrap.py   # twin-level bootstrap of every verdict (D.3)
uv run python scripts/diag_attack_budget.py    # gap-vs-attack-steps curves behind Fig. 1
uv run python scripts/fig_attack_budget.py     # Fig. 1
uv run python scripts/fig_protocol_matrix.py   # Fig. 2, seed x axis x method matrix
uv run python scripts/fig_circuit_position.py  # Fig. 3
uv run python scripts/fig_knife_edge.py        # Fig. 4, C1 <-> utility trade-off
uv run python scripts/make_tables.py           # regenerates draft/TABLES.md
```

`diag_power.py` and `diag_twin_bootstrap.py` need no GPU: they read the per-twin score files and the cached statistics. `scripts/check_refs.py` re-verifies the bibliography against the arXiv API — that every cited paper exists, that its title and full author list match in order, that preprints name the latest version, and that every entry is findable; `-offline` runs the key-consistency half, which `tests/test_refs.py` pins. `scripts/local_forget_fraction.sh <gpu> <seed>` runs the 5%-forget corpus of Section 5.4 (`CINDER_NSUBJ=80 CINDER_NFORGET=4`) through the full protocol.

C.9 Scale

Certified training above 0.5B requires model-parallel fp32 AdamW (Appendix B.3):

```
bash scripts/local_15b_seeds.sh 44 45 46 47 48 49   # every setting is in the script:
# Qwen2.5-1.5B, D.1 recipe, CINDER_OUTER_LR=1e-4, CINDER_EPS=0.5, CINDER_BATCH=8,
# CINDER_AMP=1 CINDER_GRAD_CKPT=1 CINDER_SHARD=1 CINDER_PENALTY_BATCHES=1
# (device_map=balanced over both cards), then build/null/judge per seed
```

`CINDER_SHARD=1` splits the method model across all visible GPUs for training and collapses it back to one device for evaluation, so the frozen harness never sees a sharded model.

At 3B the outer learning rate must be rescaled again, and not monotonically (Section 5.6): 2e-5 under-forgets, 1e-4 collapses retain, 5e-5 certifies on the pilot seed. Every other setting is D.1 unchanged, K = 3 included:

```
source scripts/a100_common.env             # the certified recipe, D.1, in one file
CINDER_MODEL=Qwen/Qwen2.5-3B CINDER_OUTER_LR=5e-5 \
  bash scripts/a100_rerun3b.sh             # seeds 44, 45; twins reused if present
bash scripts/a100_new3b.sh                 # seeds 46-49, same recipe, fresh twins
```

C.10 Review-driven experiments (Section 5.2, Section 5.5)

```
# one worker per GPU drains a shared job list; job lines are "fresh <seed>" or
# "var <variant> <seed>", variants defined in the script (floorsplit, relufixed,
# basefloor, the seven factorial cells, mudman, mudmantwin, mudtune_<ulr>_<rlr>)
bash scripts/review_queue.sh runs/queue/local.txt 1
uv run python scripts/diag_noninferiority.py   # right-direction test
uv run python scripts/fill_paper_tables.py     # writes the main-text NI and ablation tables
uv run python scripts/fig_tradeoff.py          # utility vs relearning figure
```

The driver prints `[recipe] inner_mode=full meta_target='twin' attack_fraction=0.5` followed by the remaining settings at the start of CINDER training and writes every setting into `results.json`; if either is missing from a run, that run is not the method. It also refuses to train with any of those three at its default unless `CINDER_ALLOW_V0=1` is set — a guard added after five 3B launch scripts inherited the defaults (Appendix B.8).

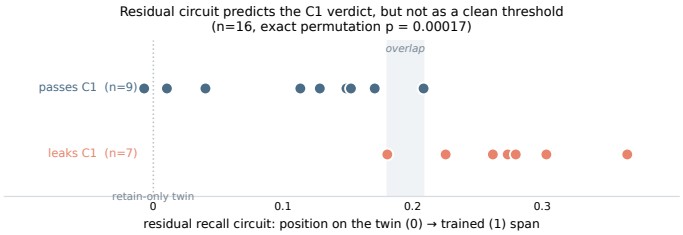

Figure 4: *Position on C1's own statistic ranks the C1 verdict.* Position of each CINDER model on the twin (0) → trained (1) span of the certificate's patching statistic; passing and rejected seeds overlap in [0.180, 0.208]. Exact permutation p = 0.00017, n = 16. Position and verdict share one statistic, so this is a consistency check, not independent evidence about a circuit.

Four settings are easy to lose because the driver's defaults are the v0 objective: `CINDER_INNER_MODE=full`, `CINDER_META_TARGET=twin`, `CINDER_ATTACK_FRACTION=0.5`, `CINDER_ASCENT_FRAC=0.0`. Check the `config.cinder` block of `results.json`, not the command line.

Two operational notes for shared machines, both learned by hitting them. A 3B protocol seed writes ~110 GB of twin checkpoints, and `torch.save` fails mid-write on a full volume rather than up front, so check free space before the build phase. And `CINDER_DEVICE` must name a card with ~48 GiB free: the single-device phases collapse the sharded model onto it, and the default can land on a card another job already holds.

The twin checkpoints are needed only to compute `scores_<seed>.json`; the judge reads the scores alone. Once a seed's sixteen score files exist, its `twin_*.pt` can be deleted and every verdict for that seed remains reproducible.

D Extended results moved from the main text

Each section below was a paragraph of the main text in an earlier draft, moved here for length rather than for weakness; the main text keeps a pointer to each, and none of the four changes a headline number. They answer four questions a careful reader asks in this order: given that an axis passed, *where* inside the twin–twin null did it pass (D.1); could the certificate have been bought more cheaply (D.2); what size of leak would a pass have caught, and how much do the verdicts owe to a null built from overlapping twin pairs (D.3); and what is wrong with the certificate itself (D.4). D.3 and D.4 are the two that work against us, and they are the two we would read first in someone else's paper.

D.1 Where the passes sit inside the twin–twin null (from Section 5.3)

Where the passes sit. A pass means an axis was not flagged, so it is worth asking whether the statistic is merely under the bar or genuinely inside the twin band. We therefore report each result's percentile within its own twin–twin null (Table 15). Two facts emerge. On relearn (median percentile 61), UMA (43) and C2 (53), CINDER is squarely inside twin–twin variation — these are passes in the strong sense. On quantization, paraphrase and in-context, however, CINDER sits at percentile 0: on most seeds its statistic is *lower than all 240 twin pairs*. That is not twin-likeness but over-suppression, which the certificate's one-sided axes do not penalise. Our own first design choice says over-suppression is a tell, so we scope the claim accordingly: under the relearning attacks tested, CINDER's recovery is within twin–twin variation, and on the evaluation-time behavioural axes it is measurably quieter than a twin. Notably C2 — the probe axis built to catch exactly this — remains mid-null, so the over-suppression is visible in raw log-probabilities but not to the prespecified probe family, which we regard as a finding about the certificate rather than about the method.

D.2 Memory-reduction routes and the memory cost of our implementation (from Section 5.6)

Five memory-reduction routes break the certified recipe (bf16 underflow; SGD-inner divergence; Adafactor in either loop; 8-bit Adam explosion), and adapter-only outer training stalls the hinge — in our implementation, full-parameter fp32 AdamW was needed in both loops. We measured the

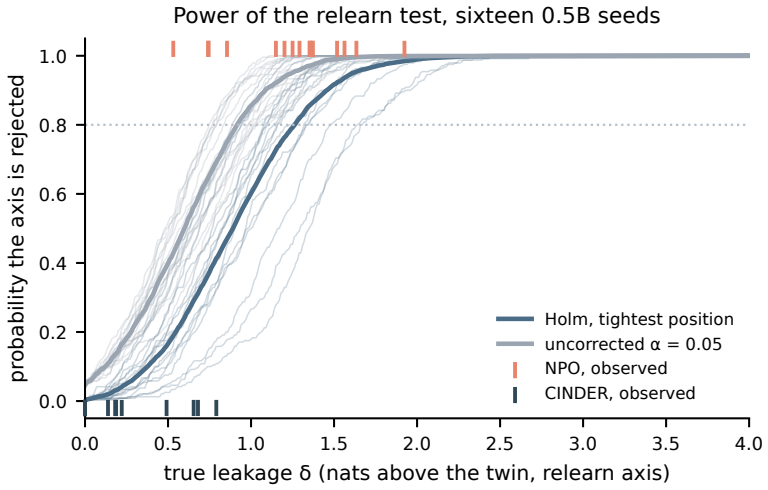

Figure 5: *Power of the relearn test on the sixteen 0.5B seeds.* Probability that the axis is rejected as a function of the true leakage δ above the twin, under the model that a leaking method's statistic is a twin–twin difference shifted by δ. Thin lines are seeds, thick lines their mean; the dark curve is Holm's tightest position, the light one an uncorrected $\alpha = 0.05$. Ticks mark the observed statistics of NPO (top) and CINDER (bottom).

resulting bill directly (`scripts/scale_budget.py`): 12.9 GB at 0.5B, 40.3 GB at 1.5B, 80.6 GB at 3B and 199.4 GB at 7B — 28.1, 28.0, 28.0 and 28.1 bytes per parameter. The constant is stable to 0.1 B/param over a 15.5× size range and across two GPU architectures (RTX 5090 sm_120 and A100 sm_80), with a fitted intercept of 4 MB: the cost is pure per-parameter state, 7× the fp32 weights, which is what exhausts a single 32 GB card above 0.5B. We projected the 7B figure at 199 GiB from the three smaller points before running it and measured 199.4 GiB, a 0.2% error, so the linear fit held to 7B. It is the cost of this implementation — AdamW state in two loops — not a lower bound for the problem, and it is a budget, not a barrier: with training sharded across cards and evaluation collapsed back onto one — so the frozen harness never sees a split model — the same recipe runs unchanged at 1.5B and 3B.

D.3 What a pass cannot mean — the minimum detectable effect (from Section 6)

- **What a pass cannot mean.** The conformal p-value floors at 1/241 and Holm's tightest threshold is $\alpha/8 = 0.00625$, so at the stringent positions an axis is flagged only when its statistic exceeds *every* one of the 240 null values. That null maximum is large: median 0.878 nats on relearn, 1.323 on C1, 0.663 on quantization (Table 16), and the 80%-power effect below is larger still. A pass is a failure to reject and bounds nothing; power describes how often a leak of a given size would be missed, not a confidence bound on the leak of a model that passed. The bound the paper uses is the method-level one-sided interval of Section 5.2, where the hypotheses are the right way round.

- **Power, per axis** (`scripts/diag_power.py`). Leakage at which the per-seed test reaches 80% power, Holm-tightest position, median over sixteen seeds [range]; and the test's power at the effect NPO and CINDER actually show:

The resolution is coarse on the log-probability axes (relearn, quantization, paraphrase, in-context: 0.8–1.6 nats) and fine on the bounded ones (UMA, C2, utility: 0.06–0.25). NPO's relearn effect sits at the resolution, hence 13/16 detections; CINDER's sits well below it on every forget axis. The utility row is the axis of D.4(ii): its 80%-power leakage spans 32× across seeds.

- **Twin-level bootstrap** (`scripts/diag_twin_bootstrap.py`, B = 2,000). Probability of certification when the sixteen twins are resampled with replacement and every null and reference rebuilt; asterisks mark the point verdict:

Expected certifications: CINDER 6.43, NPO 1.43, untouched model 0.00. Seed 57 (0.41) is the one near-miss on CINDER's side; the ordering of the two methods is the same under resampling, the

Table 21: Resolution of the per-axis test, sixteen 0.5B seeds. The 80%-power column is the leakage a pass would have caught four times in five; the last two columns evaluate each seed's own power curve at the effect that method actually shows, so a high value means the test was capable of catching that method and a low one means a pass is weak evidence.

axis	80%-power leakage	power at NPO's statistic	at CINDER's
relearn	1.16 [0.89, 1.66]	0.73	0.09
quant	0.82 [0.61, 1.39]	0.60	0.05
paraphrase	0.90 [0.62, 1.53]	0.54	0.02
in-context	1.58 [0.88, 2.30]	0.26	0.00
UMA	0.25 [0.16, 0.32]	0.03	0.01
C1	1.64 [1.16, 2.26]	0.76	0.39
C2	0.15 [0.13, 0.26]	0.09	0.02
utility	0.06 [0.01, 0.32]	0.03	0.31

Table 22: Certification probability under twin-level resampling, B = 2,000. A cell is the fraction of resamples in which that method certifies on that seed; an asterisk marks the seeds the preregistered point verdict certifies. For CINDER, thresholding at 0.5 selects the same eight seeds as the point verdict; for NPO it does not (seed 48: 0.75, not certified at the point; seed 58: 0.16, certified). Either way the verdicts are probabilities, not facts: CINDER's certified seeds sit at 0.54–0.95 and its expected count is 6.43, not 8.

seed	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58
CINDER	*.82	*.95	.00	.00	.00	.08	.00	.00	.00	*.54	*.64	*.57	*.66	*.92	.41	*.85
NPO	.00	.00	.00	.00	.00	.75	.00	.00	.00	.00	.00	.00	*.51	.00	.00	*.16

counts are not. With the eight disjoint twin pairs as the null, the p-value floor is 1/9 and nothing can be rejected at $\alpha = 0.05$: the certificate has no resolution without the shared pairs, a fact about its design that belongs upstream (D.4).

D.4 Four defects of the certificate, passed upstream (from Section 6)

- Four observations about the certificate itself, passed back to its authors rather than exploited. (i) The v1.1 general-text utility axis, defined in the harness but never shipped in the protocol, is unpassable by ANY method including NPO and the untouched control — its twin-twin null contains only seed variation and no benign method drift. (ii) The shipped v1 utility axis's per-seed resolution is a property of the twin draw: across sixteen 0.5B twin populations its null width spans 288×, where every forget axis spans 2–3× (Section 5.3). A retain-damage statistic of 0.009 is a rejection on one seed and would pass on fourteen others. It needs a variance floor, or a pooled null, before its rejections can be read as the method's. (iii) At 1.5B the same axis rejected the untouched model on one seed, which we briefly took for a scale effect until a second seed showed otherwise — the same defect, seen earlier and read wrong. (iv) The twin is meant to be the over-suppression null as well as the recovery floor (Section 3), but the only axis that can register over-suppression is C2, the probe. On the 5%-forget corpus (Section 5.4) CINDER sits 6.7 nats below the twin on the forget set before any attack and C2 passes at p = 0.92; the certificate certifies a model that is conspicuously *more* ignorant than one that never saw the data, which is itself a tell an adversary could read. A pre-attack forget log-probability axis tested in the reverse direction would close it.

D.5 The C1–utility knife-edge under the circuit term (from Section 5.3)

The circuit term ($\lambda 3$, Appendix B.4) was the one curriculum term the protocol asked for: C1 is CINDER's most frequent rejection. On the protocol-tuning seed every weight tried moved C1 from rejected to passing (p 0.004 → 0.012–0.31) and every one was rejected on utility, so it is reported as a trade-off rather than a component. One seed and four weights; whether a weight exists that passes both is open.

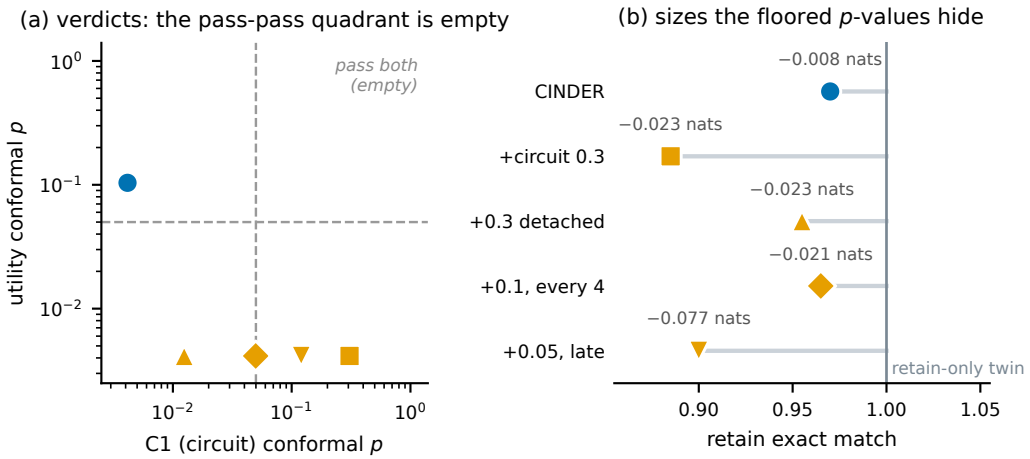

Figure 6: *The C1↔utility knife-edge under the circuit term λ3, seed 45.* (a) Every tested weight repairs C1 and every one is rejected on utility: the pass–pass quadrant is empty. (b) The same runs in model terms, since the utility p-value floors at 1/241 and the four look identical there.