

It Was the Learning Rate: Compute-Matched Baselines Erase the Apparent Gains of a Memristor Crossbar Repair Controller

Prateek Yadav

Abstract—None of the memristor-crossbar repair papers we examined tests against a tuned, equal-compute baseline. We measured what that omission is worth. Our repair controller (D) paces its spare cells and gates retraining on a held-out set. It first looked 34.0 points ahead of a rival with *living spares* (a spare keeps training from the cell’s current weight). Tuning the rival’s learning rate cut the lead to 4.2 points. Giving it D’s number of retraining (burst) steps cut it to 1.06. A pre-specified eight-seed replication cut it to -0.005 points (95% CI -1.7 to $+1.7$). An unfair baseline manufactured a 34-point win, and fair baselines removed it. D still beats retraining-only at equal burst steps by 3.78 points (5 of 5 seeds), so the controller is genuine where it holds. Two more findings: a forward-only BatchNorm refit stays at chance once spares run out (76.06 points below D, 0 of 8 seeds), and living spares match frozen-reference spares at a tuned rate ($+0.03$ points; an earlier $+2.48$ came from a barely working rate). The scope is simulation, ResNet-9 on CIFAR-10. The lesson is a call to the field: tuned, step-matched baselines and a replication are the price of a credible margin. All code, runs and analysis are public.

Index Terms—Memristor crossbar, stuck-at faults, spare cells, fault-aware retraining, in-memory computing, reliability, baseline tuning, replication.

I. INTRODUCTION

RPAIR controllers for memristor crossbars report large wins. In the papers we examined, none faces a rival that was tuned and given the same compute. We built such a controller, saw it lead by 34 points, and then did the control experiment: we tuned the rival’s learning rate, matched its retraining steps, and replicated on pre-specified seeds. The lead fell to zero. This paper reports that fall, step by step, and what survives it.

An analog resistive-memory (RRAM) chip stores neural-network weights as cell conductances. Cells fail while the chip is deployed. A failed cell is stuck at a high or a low conductance and cannot be written again. A chip designer can keep a pool of spare cells and replace failed cells with spares. The pool is finite, and each wave of new faults uses part of it. Spare cells are therefore a budget that the chip spends over its life.

Two repair tools exist. Remapping moves a failed weight onto a spare cell. Retraining adjusts the healthy weights so the network works around the faults. Prior work uses each tool, often once and from a known fault map [1]–[3]. A deployed chip needs both, repeatedly, across several fault waves. In the

papers we examined (Table II), no baseline received both the same compute and a tuned learning rate; Xia et al. report a 1.2x iteration count for their method, and Song et al. match iterations for one baseline [3], [4]. We re-ran none of these methods, so this describes what the papers report, not how the methods would perform. Two questions follow. Do remapping and retraining interact? And how much of a clever repair controller’s gain survives a fair comparison?

They do interact. Remap restored a spare toward a reference that the chip captured once, before any retraining, so each remap overwrote the compensation that earlier retraining had learned. We found this by reading the write paths, then measured it. The fix is simple: a remapped spare takes the cell’s current weight and trains in later bursts. We call these living spares. Living spares first showed a gain of 2.48 points over frozen references, but at an under-tuned burst rate. A pre-specified control at the tuned rate (3 seeds, paired within seed) found $+0.03$ points, which is noise-level, and a lifetime accuracy 0.31 points lower. The $+2.48$ was an artifact. We report the overwrite as a code-level observation, not as an accuracy gain.

Living spares make a second question matter: how fast to spend the pool, and when to retrain. Our controller D spends a per-wave share of the free spares (pacing). It ranks cells by saliency, refits BatchNorm statistics, and retrains in a short burst only when a held-out canary set shows a loss. Importance-ranked remap, BatchNorm refit, plateau stop and canary-triggered retraining each have prior art [1], [5], [6], and we do not claim them. We use D to ask whether this machinery buys accuracy when every rival gets the same number of burst steps.

The ladder. The answer is the paper’s centerpiece. A first version of this work reported a 34-point lead over a living-spares rival. At least 97% of that lead came from an untuned learning rate and unequal compute. Each fix removed a slice of it:

$$34.0 \rightarrow 4.2 \rightarrow 1.06 \rightarrow -0.005 \text{ points}$$

(old rate, unmatched $\rightarrow$ tuned $\rightarrow$ step-matched, 5 seeds $\rightarrow$ 8-seed replication)

That ladder is a quantified decomposition of how an unfair baseline builds a large win. We caught our own 34-point illusion. Any repair-controller paper can now be checked against it.

What we did differently. We specified hypotheses and decision rules before the counted runs (a commitment in the

author’s own timestamped repository, not an independent registry). We gave each rival D’s exact number of burst steps (Section IV-C). We tuned each arm’s learning rate and report both a test-picked and a canary-picked choice. We audited the raw run data and fixed simulator errors that had changed earlier numbers. We replicated our own best margin and ran two pre-specified reviewer baselines.

What D still wins. D beats retraining-only at equal burst steps by 3.78 points, in 5 of 5 seeds, significant after Holm correction. That is a real result where it holds. Against the strongest matched rival (living-spare retraining at D’s burst steps, C’-matched) and a one-shot Liu-style retrain at D’s burst steps (E1), D is not distinguishable on final accuracy. The eight-seed mean against C’-matched is -0.005 (95% CI -1.69 to $+1.68$; 6 of 8 seeds ahead, Holm $p = 1.0$). D is ahead on 6 of 8 seeds by 0.44 to 1.52 points and behind by about 3 points on the 2 seeds where its gate skipped a burst (Section V-E). D serves far better through life than the one-shot retrain (Section V-D). A forward-only BatchNorm repair stays at chance when the spare pool is exhausted. Sections V-I and VI give the other settings and every limit.

Contributions. This paper is a quantified case study, not a new repair method and not a methodological discovery: that baselines should be tuned and matched is established practice (Section II). It is the missing control experiment for a memristor repair controller, run end to end and published in full.

- 1) We **quantify** how an unfair baseline manufactures a win: a 34.0-point lead falls to 4.2 (tuned rate), 1.06 (step-matched) and -0.005 (8-seed replication, 95% CI -1.7 to $+1.7$), and a 44.7-point retraining-only lead falls to 3.78 (Sections V-B and V-C).
- 2) We **show** that D still beats retraining-only at equal burst steps by 3.78 points in 5 of 5 seeds, and that a one-shot Liu-style retrain at D’s steps cannot be told apart from D on final accuracy but serves far worse through life (Section V-D).
- 3) We **trace** a frozen-reference remap that overwrites retrained weights, and we **retire** its apparent payoff: living spares give $+0.03$ points at the tuned rate, against $+2.48$ at a barely working one (Section V-F).
- 4) We **establish** that a BatchNorm refit does not rescue a chip whose spare pool is exhausted: it stays at chance on 8 of 8 seeds (Section V-D). Table VIII gives what each design that reaches the same final accuracy costs in spares, canary data, tuning and service through life.
- 5) We **diagnose** the controller’s gate drift failure mode, **test** D on a probe-estimated fault map, and **catalogue** the simulator pitfalls that would have silently corrupted results, each with a stored measurement (Sections V-E, V-K and IV-D).
- 6) We **release** everything: code, configurations, raw run records and analysis scripts are public, and the numbers re-derive from the stored event logs (Availability).

The rest of the paper is organized as follows. Section II reviews related work, Section III describes the method, Section IV the experimental setup, Section V the results, Sec-

TABLE I
LEGEND: ARM LETTERS AND EXPERIMENT LABELS USED IN THE TEXT.

Label	Meaning
D	our controller (paced remap, saliency ranking, Batch-Norm refit, canary-gated burst)
C’-matched	living-spare retraining at D’s burst steps (strongest fair rival)
B-matched	retraining only, no remap, at D’s burst steps
R0 / R1	no repair / remap only
E1 (-last, -first)	Liu-style one-shot retrain at the last / first wave
E2	BatchNorm refit only, no optimizer step
E3	replication of D versus C’-matched on 3 new seeds

tion VI limitations, and Section VII concludes. Figure 1 gives the system and the story in one view.

II. BACKGROUND AND RELATED WORK

A. Setting and terms

We use one term for one thing. *Spare cell*: a reserve cell that can replace a failed cell. *Wave*: one fault event. Faults are permanent: a stuck cell stays at its high-resistance (HRS) or low-resistance (LRS) conductance. *Burst*: a short retraining run between waves. *Canary set* (a calibration set): 1,024 held-out training images that serve as the only health signal. *Arm*: one policy in a comparison (letters are defined in Table I). *Deployment regime*: we train once to a checkpoint, then run inference through K fault waves. Retraining happens only inside policy bursts. Labelled data for bursts is assumed to be available on the device or at the edge.

B. Related work

Table II positions the closest work. We read each paper listed there; several 2025–2026 items are preprints or recent journal articles, and we checked their identifiers in October 2026. Our claims are limited by the coverage of that reading and of our search.

Retrain once, from a known map. Chen et al. and Song et al. map weights to a crossbar with redundant rows, then retrain [2], [3]. Jin et al. rank weights by gradient significance and retrain after mapping [7]. Liu et al. retrain with masks, then remap only the most significant defects onto a capped pool of redundant columns [1]. All four work once and from a static map. The conflict between remapping and retrained weights cannot arise in that order. Importance-ranked remap is Liu’s contribution, not ours.

Lifetime loops. Xia et al. alternate on-line fault detection with fault-tolerant training; the journal version adds a threshold on the share of newly detected faults [4], [8]. Tung et al. remap whole tasks every epoch from fault-density tests during training [9]. RRAMedy detects a health change on a deployed chip, diagnoses it, and retrains [6], [10]. It rejects spares as too costly. These papers publish the lifetime detect-then-retrain loop. We do not claim it. RRAMedy’s mitigation policy, run under an oracle map, is our arm B (Table III).

Spares and repair budgets. Xia et al. describe a post-fabrication, reconfigurable spare pool that is allocated greedily to the worst columns [11]. Error-correcting pointers allocate

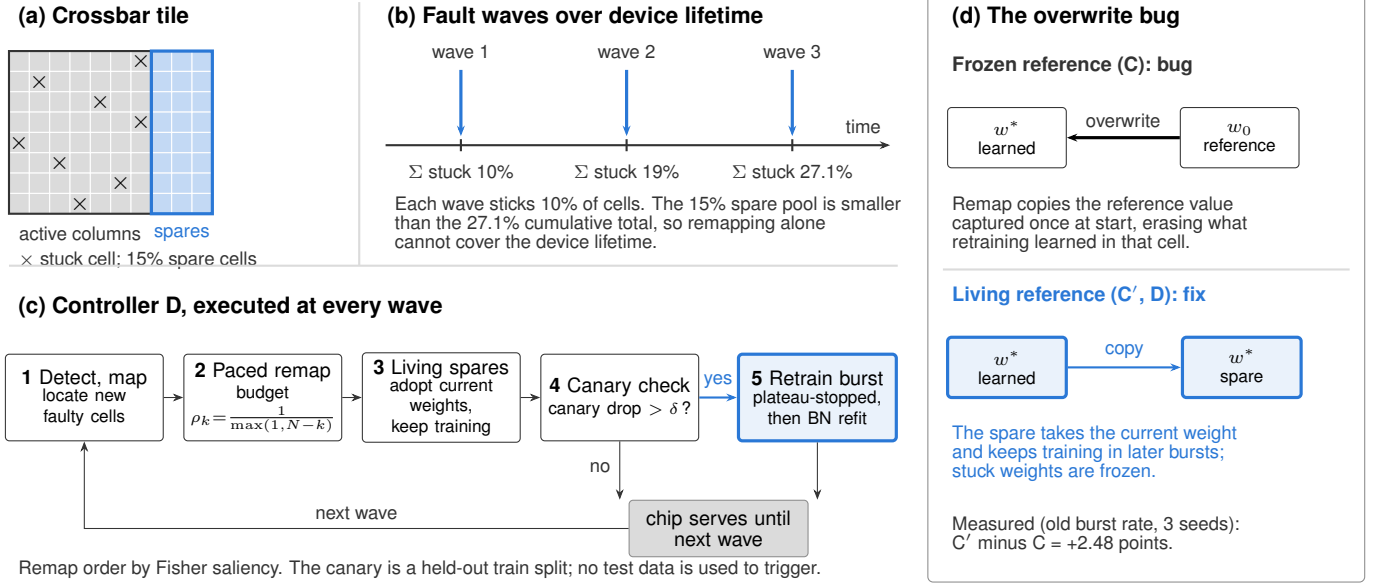


Fig. 1. System and story. A chip loses cells in waves; a finite spare pool is the repair budget. Remap to a frozen reference restores the original value over what retraining learned. A living reference keeps it (no accuracy gain was measured at the tuned rate). Controller D spends the pool at a paced rate and retrains in short, canary-gated bursts. This is a diagram; it shows no measured data.

greedily as cells fail [12]. Khaleghi and Rao choose *which* shared spare to consume, to keep later repairs possible [13]. Hamun defers retirement of worn cells until the estimated accuracy loss passes a limit [14]. ReSAFT sizes redundancy at design time [15]. None of these sets *how many* spares to spend per wave inside a retraining controller. Greedy spending is our baseline for that question.

Retraining-free and compensation schemes. Jung et al. refit BatchNorm statistics without gradient steps [5]. Fault-Free re-encodes weights at compile time and compensates the largest errors [16]. SEC-CiM and SEC-CiM+ use a digital error memory under a fixed budget; the budget is used first come, first served [17], [18]. SEC-CiM+ avoids the overwrite conflict by not retraining. DEAR refreshes a stored error after each weight update during training [19]. Maharjan and Lu re-clamp stuck weights after each optimizer step [20]. These are partial precedents for parts of our fix: a freeze of stuck weights, and a refresh of a stored reference. We claim only their combination with a finite pool, trained spares, and a deployed chip (Section III).

Map-free repair and fair comparison. STAR repairs a network without any fault localization, using Equilibrium Propagation after a one-shot fault injection [21]. Its learning rule does not port to our backpropagation network. Our arm B-matched is the unassisted baseline of that class, not STAR itself. Xia et al. report a 1.2x iteration count for their method, and Song et al. match iterations for one baseline [3], [4]. We give every rival the controller’s burst-step count and tune each arm’s learning rate.

Baseline tuning in machine learning. That weak or untuned baselines inflate reported gains is documented elsewhere. Blalock et al. found that pruning papers were hard to compare for lack of standard baselines and benchmarks [22]. Melis et al. showed that a well-tuned standard LSTM beat more recent

language models [23], and Lucic et al. found that most GAN variants reach similar scores given enough tuning and restarts [24]. Dodge et al. argue for reporting results as a function of the tuning budget [25], and Bouthillier et al. quantify the variance that seeds and tuning add to benchmark comparisons [26]. In memristor inference, Joshi et al. train with injected noise so that phase-change-memory inference is accurate [27]; their setting is analog noise, not stuck-at faults. Our case study applies the same lesson, with a measured decomposition, to a fault-repair controller.

III. METHOD

A. The overwrite problem

A remap replaces a stuck cell by a spare. In a frozen-reference design, the pool stores a reference conductance when it is built. Every later remap restores the spare toward that reference. Between waves, retraining changes the weights of healthy cells so that they compensate for faults. A later remap then writes the old reference over a cell that retraining had adjusted. The compensation can be lost, and nothing reports it. This is a code-level observation; Section V-F shows that we could not measure an accuracy cost at the tuned rate.

B. Living spares

A living spare takes the cell’s current weight when the remap happens. It is not pinned to a reference, and it trains in later bursts. A second change keeps stuck weights fixed during retraining. After each optimizer step, the value of every stuck cell is restored. Otherwise gradient, momentum and weight decay drift the stuck weights, and a later remap copies that drift into a spare. A spare that is remapped can still fail again. It counts against the pool. Freezing stuck weights and refreshing references both have partial precedents (Section II).

TABLE II
CLOSEST PRIOR WORK AND WHAT THIS PAPER DOES WITH IT. “NOT CLAIMED” MEANS WE USE THE ELEMENT AS PUBLISHED.

Work	What it does	Relation to this paper
Liu DAC’17 [1]	One-shot masked retraining, then remap of top defects onto a capped pool; oracle map	Importance-ranked remap: not claimed. Our E1 arm adapts it to waves.
Chen DATE’17, ITT-RNA [2], [3]	Map first, retrain once; trained redundant cell in a differential pair; static	Static, offline. Our setting adds waves and a pool drawn at fault time.
Jin DATE’20 [7]	Gradient-significance ranking, masked re-training	Ranking: not claimed.
Jung DATE’21 [5]	BatchNorm refit, no gradient step	BN refit: not claimed. Our E2 arm tests it alone.
Xia DAC’17, TCAD’19 [4], [8]	Periodic detection and tolerant training; fault-share gate	Lifetime loop: not claimed. No spares, no pacing.
Xia JETCAS’18 [11]	Reconfigurable spare pool, greedy allocation	Nearest spare-pool precedent. Sets where, not when.
Tung DATE’23 [9]	Per-epoch whole-crossbar task remap during training	Online remap; no spares, no retraining bursts.
RRAMedy ICCD’19, TC’22 [6], [10]	Alarm, diagnosis, retraining of a deployed chip; no spares	Generic gate and loop: not claimed. Arm B stands for it.
SEC-CiM, SEC-CiM+ [17], [18]	Digital error compensation under a fixed budget; no retraining	Budget without pacing. Discussion only.
DEAR, Maharjan [19], [20]	Stored-error refresh; stuck-weight re-clamp	Partial precedents for our fix.
Hamun, ECP, Khaleghi, ReSAFT [12]–[15]	Deferred retirement; greedy pointers; which spare; design-time sizing	Greedy spending is our baseline.
STAR [21]	Map-free repair with Equilibrium Propagation	B-matched is the same class; not a run of STAR.

Our contribution is the combination for a finite pool on a deployed chip with several waves.

C. Controller D

D acts after each wave. It uses only quantities that a deployed chip can observe: the fault map (oracle or estimated, Section V-K) and the canary set.

- 1) **Paced remap.** At wave k of N expected waves, each layer may spend a share $\rho_k = 1/\max(1, N - k)$ of its free spares. The last expected wave may spend all that remain. Cells are ranked by a Fisher saliency score.
- 2) **BatchNorm refit.** D refits BatchNorm statistics in a forward-only pass.
- 3) **Canary gate.** D reads the canary set. If the drop against a reference is below a margin, D skips retraining. The reference is the canary reading after the previous wave’s repair (Section V-E).
- 4) **Burst.** Otherwise D retrains in a burst with fresh SGD and a cosine learning-rate decay. The burst stops at a plateau of the canary signal or at a step cap.

D needs the expected number of waves N . A wrong belief costs about 0.9 to 1.0 point at the tuned rate (3 seeds, two priors files). D used 1,375 to 1,500 burst steps per seed on the first five seeds, and 700 to 1,500 over all eight.

D. Arms

Table III defines every arm. B-matched and C’-matched receive the same number of burst steps as D on the same seed. That equal burst steps is the main control of this study. It does not charge D for its extra passes: Fisher-saliency scoring,

canary forward passes and BatchNorm refit passes are not counted as steps, so D does less comparable work than its step count suggests and the matched rivals are not given those passes. The rivals also receive D’s step count after the fact, which a deployed rival could not know in advance. E1 and E2 are the two reviewer baselines that we specified in advance.

IV. EXPERIMENTAL SETUP

A. Setup

The network is a ResNet-9 on CIFAR-10. We run it in the CrossSim simulator (version 3.2.1) with an 8-bit DAC and a calibrated 8-bit ADC. The main runs have no read noise, no programming error and no drift error. Each seed trains one checkpoint on 50,000 images. The fault-free deployed accuracy is 88.92 percent (5 seeds). The default test has three waves of 10% random stuck-at cells. The cumulative stuck fraction is 27.1%. The pool holds 15% spares, one per cell, without area or routing cost. The fault map is an oracle in the main runs. Accuracy is the accuracy after the last wave’s burst on the first 5,000 test images (sampling noise about ± 0.5 points). The main test uses seeds 42, 101, 102, 103 and 104, and E3 adds seeds 105, 106 and 107. Learning rates and D’s settings (gate margin, plateau rule, priors) were chosen on seed 42 and the first seeds, and the ResNet-9 rates on test accuracy; these seeds and images are also evaluation seeds and images, so no result here comes from data untouched by tuning. B and B-matched were not run on seeds 105 to 107. The same seed gives the same checkpoint and the same fault maps to every arm; we check the fault-map hash. Rate and gate settings are in the logs of each run, not in the stored configuration.

TABLE III

ARM DEFINITIONS. ALL ARMS SHARE THE SAME CHECKPOINT AND FAULT MAPS WITHIN A SEED (HASH-CHECKED). B-MATCHED AND C'-MATCHED USE D'S BURST STEPS FOR THE SEED.

Arm	Definition	Role
R0 / R1	No mitigation / remap only, random order, no retraining	Static baselines
B	One fixed burst after every wave; no remap, no spares	Retraining only (RRAMedy-class policy under an oracle map)
B-matched	B with D's total burst steps for the seed	Equal-burst-step retraining-only
C	Remap; burst only when a wave overflows the pool; frozen reference	Overwrite bug
C'	C with living spares	Spares without a controller
C'-matched	C' with D's burst steps for the seed	Strongest fair rival
D	Controller of Section III-C	This paper
E1	One Liu-style burst at the last wave; greedy saliency-ordered remap each wave; no pacing, gate or BN refit; burst length equals D's steps; rate 0.5 not re-swept	Closest prior art, adapted to waves
E2	Random-order remap each wave, then BatchNorm refit; zero optimizer steps	What a forward-only repair recovers

B. Statistics

We pair arms by seed and use a two-sided paired t test as the primary test. Holm correction runs inside families fixed in advance (the members of family H are listed in the appendix): a headline family H of 15 comparisons, a limitation family N of 5, and an exploratory family S of 64. The E1 and E2 comparisons form a family E of 4, and the lifetime (area under the per-wave curve) comparisons form a family of 28. Rows in family S are exploratory: we mark them “significant (exploratory family)”. We call a result *significant* only if the Holm-adjusted p is below 0.05 and the one-sided sign test is below 0.05. That needs $n \geq 5$ and all seeds in one direction. We call a result *consistent* if all seeds agree but it is not significant, *mixed* if seeds disagree, and *noise-level* if the mean is under twice the noise of the mean. Ten identical reruns differ by an RMS of 0.46 points; the noise of a mean of five seeds is about 0.20. With $n = 3$, no result can be significant, by construction of this rule; a three-seed row is therefore never evidence of absence of an effect, only of too little power to show one.

C. Methodology and rigor

Table IV lists each control and its limit. We kept four habits.

We specified before we ran. The plan is pre-specified, not pre-registered: the criteria for E1 to E5 were committed to the author’s own timestamped development repository before any counted run of them, but not to an independent registry, so the timestamp is not independently verifiable. The plan states hypotheses, direction, decision rules and the families for Holm correction. For E3 it says what counts as a replication (Section V-C). Seeds 103 and 104 were added after a first look at seeds 42, 101 and 102. Seeds 105 to 107 (E3) were added after the five-seed analysis, so the +1.06 point result on five seeds is an interim look. E3 (seeds 105 to 107) was specified in commit 018276a before it ran (Availability).

We matched burst steps and tuned every arm. Each rival gets D’s burst steps. Each arm’s burst learning rate is swept. The learning rate was the largest hidden factor we found (Section V-B).

We audited the raw data. An audit pass by the author’s scripts, not an independent party, re-derived the numbers from the stored event logs. It checked 683 complete deployment lifetimes: 562 passed, 121 raised benign warnings, and none failed. It also corrected 53 wording errors in our own earlier report. We list the remaining provenance limits in the table.

We report failures. E3 did not replicate our best margin, and E1 could not be separated from D on final accuracy. A closed-loop extension was killed at 4% noise. We found a failure mode of our own controller’s gate after the verdicts were fixed, and we label that diagnosis post-hoc. A pre-specified variant of D, called D2 (D with an adaptive trigger that stops bursts earlier, about 1,220 against 1,470 steps), failed its main-test criterion, and D stays the controller. D2 passed its calibrated-scenario criterion in 4 of 4 scenarios. Section V reports each outcome as it came.

D. Simulation fidelity

A simulator that gives the wrong answer for the right reasons is dangerous. Our simulator needed many fixes before its numbers meant anything. We publish the catalogue because other groups use similar tools. We report only entries with a stored measurement in Table V. We found further errors in our development log, such as a training step that never reached the analog core and a backend whose forward pass ignored conductance. We do not report effect sizes for those, because a stored probe is still missing.

Three lessons stand out. First, converter details dominate accuracy. A fixed input-clipping range in the DAC collapsed the analog network at the float-to-analog handover. Fixing it moved one run from 45.98 to 89.79 percent and removed handover drops of 58 to 75 points. That comparison is unpaired (one seed against four), so we use it as a scale, not as an effect size. An earlier “model capacity” conclusion was an artifact of this defect. Second, the repair logic had its own bugs. Two remap bugs left a third of the spare pool unused and blocked remap at the last wave. A test suite of 207 tests passed while a mitigation bug was present, because the tests ran each writer alone and never together with an optimizer step. Third,

TABLE IV
METHODOLOGY LEDGER. EACH ROW IS A CONTROL WE USED, WHAT IT PROTECTS AGAINST, AND ITS KNOWN LIMIT.

Control	What we did	Known limit
Pre-specification	E1 to E5 (including E3) committed at 018276a (2026-10-14) before any counted run; decision rules fixed in advance	Author-timestamped, not independently registered; earlier experiments (first-wave spectral work) were specified after nine runs; see companion paper (in preparation)
Sequential design	Seeds 103 and 104 added after a first look at seeds 42, 101 and 102; E3 adds seeds 105 to 107	The eight-seed test is a second look; its p value is optimistic. D versus B-matched (5 seeds) was also extended after a first look. We print new seeds alone and pooled; B-matched has five seeds
Equal burst steps	B-matched and C'-matched get D's exact steps (± 1) per seed	Steps of the burst only: D's saliency, canary and BatchNorm-refit passes are not charged; rivals get D's step count in hindsight; write cycles and endurance are not costed
Learning-rate tuning	Per-arm rate sweeps; D keeps the rate it was tuned at	ResNet-9 rates were picked on seed-42 <i>test</i> accuracy (5,000 images), and seed 42 is an evaluation seed; we print both picks (Table VII). D has more settings (gate margin, plateau rule, priors, rate) than the matched rivals (rate only); matched arms were not re-swept at matched compute
Raw-data audit	Numbers re-derived from event logs; fault-map hashes equal across arms within a seed	No run records its code commit (0 of 683); one snapshot commit, 32aa9ae, holds the code the runs used (Availability)
Fault-map pairing	Same seed, same hash across arms	The same seed gives different maps on different GPU models (9 of 9 multi-GPU buckets differ); seeds 103 and 104 ran on another GPU than 42 to 102
Noise floor	Rerun RMS 0.46 points; largest single difference 1.22	A single-seed gap below 1.22 points is not interpretable
Failed replications and kills reported	E3 failure, E1 not distinguishable, closed-loop 4% extension, D2, many-small-wave and S4 losses, gate drift	The gate-drift diagnosis is post-hoc

comparisons need hygiene. The same seed gives different fault maps on different GPU models. Best-of- N maxima over 156 evaluations inflate a summary table. A device comparison can be vacuous: five of five arms of an STT-MRAM run were identical to the default device.

V. RESULTS

A. Main contrasts

Figure 2 and Table VI give the main test. The test has eight seeds for D, C'-matched, E1, E2 and R1. B and B-matched have five seeds, because E3 did not run them on the three new seeds. Three groups of arms appear. Remap-only (R1) and the forward-only BatchNorm repair (E2) end at chance. B-matched, the equal-burst-step retraining-only arm, ends near 85 percent. D, C'-matched and the one-shot Liu-style arm (E1-last) end within 0.6 points of each other on the mean and are not distinguishable on 8 seeds (no equivalence test; the 95% interval for D minus C'-matched is -1.69 to $+1.68$).

D beats B-matched by 3.78 points (per seed $+4.14$, $+3.52$, $+3.68$, $+4.46$, $+3.10$; 5 seeds). The 95% interval is $+3.1$ to $+4.4$. The Holm-adjusted p is 0.0014 (raw paired- t $p = 9.2 \times 10^{-5}$), so the result is significant. This is a sequential design: we looked at 3 seeds, then extended to 5, and no alpha-spending was planned, so the label is optimistic. As a check, splitting 0.05 evenly over the two looks (0.025 each) leaves

both the raw and the Holm-adjusted p below the threshold; the family itself is listed in the appendix and also fixes how much the Holm adjustment costs. Without seed 42, the seed where rates were picked, the gap is 3.69 in 4 of 4 seeds. On these five seeds D ends at 88.50 percent, B-matched at 84.72, and the fault-free chip at 88.92. This is a five-seed result: B-matched was not run on seeds 105 to 107. Remap alone (R1) ends near chance: D beats it by 78.77 points (5 of 5 seeds, significant, exploratory family S). Retraining is therefore needed at 15% spares. Spares add to retraining: C'-matched beats B-matched by 2.72 points (5 of 5 seeds, Holm 0.004, significant in the exploratory family S). That contrast compares spares and remap with retraining only, so it does not isolate living spares.

B. Tuning and burst steps explain most of the first headline

Our first draft reported D 83.3 against C' 49.3 and B 38.6. The burst learning rate was at the edge of the calibration grid for B and C'. At the tuned rate, B rises from 38.59 to 83.32 ($+44.73$, 3 of 3 seeds, Holm 0.003). C' rises from 49.30 to 84.31 ($+35.01$), and D from 83.33 to 88.52 ($+5.19$). The lead of D over C' shrinks from 34.03 points (old rate, unmatched), to 4.21 (tuned, unmatched), to 0.97 (tuned and step-matched, 3 seeds) and 1.06 (5 seeds). On eight seeds it is -0.005 . Burst-step matching and tuning changed the conclusion at every step.

TABLE V

SIMULATOR PITFALLS WITH A STORED MEASUREMENT. EACH WAS FOUND BY US AND FIXED OR DOCUMENTED. “EFFECT” IS THE MEASURED CHANGE IN A REPORTED NUMBER.

Pitfall	Symptom	Measured effect
DAC input clipping	Analog phase collapses at handover; bit-depth sweeps saturate	One run 45.98 $\rightarrow$ 89.79 percent (unpaired, one seed vs four); handover drop 58–75 points $\rightarrow$ under 1
Capacity misdiagnosis	A ladder of net sizes read as a capacity limit	Conclusion invalidated by the DAC fix
Regime saturation	Density 0.1 every 8 batches	$1 - 0.9^{118} = 100\%$ stuck; every arm at chance
Remap selection stall	Remapped cells stayed in the stuck mask	Pre-fix runs used 656,868 of 985,293 spares (66.7%, with the gate bug); post-fix runs use all
Synthetic health gate	Remap gated on a drifting stress metric	Remap could never run at wave 3
Stale reference (the over-write)	Remap restores a reference captured once	Living minus frozen +2.48 points at the old rate (superseded); +0.03 at the tuned rate, noise-level
Dispatch on healthy cells	Soft mitigation acted on non-stuck cells	About 26 million cells touched per run; a claimed +2.56 gain was confounded
Baseline under-tuning	Burst rate at the edge of the grid	Retraining-only +44.7 points when tuned, 3 of 3 seeds
Fault-map pairing	Classic arms use the global RNG	14 of 223 report pairs unpaired
Reporting	Best-of- N maxima; duplicate directory counted as a seed	Maximum over 156 evaluations is optimistic by construction
Test-suite blind spot	Writers tested without an optimizer step	207 tests passed with a mitigation bug present
Vacuous device comparison	STT-MRAM equals default device	5 of 5 arms identical; no device conclusion

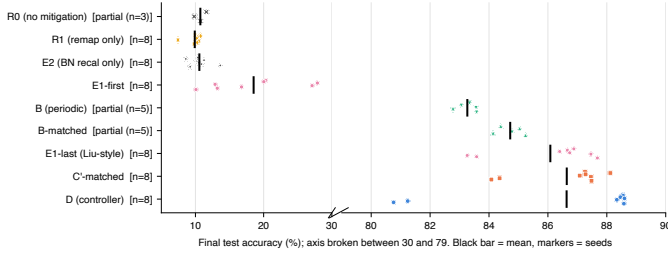


Fig. 2. Final test accuracy per arm after the last wave (ResNet-9, CIFAR-10, three waves of 10%, 15% spares, oracle map). Each marker is one seed; the black bar is the mean. Eight seeds (42, 101–107) for D, C'-matched, E1, E2 and R1; five seeds (42, 101–104) for B and B-matched; three for R0. Means: D 86.64, C'-matched 86.64, E1-last 86.08, B-matched 84.72, E2 10.58. Notice that D and C'-matched have the same mean, and that the two low D seeds (105, 106) are the ones that pull D down. The vertical axis is broken between 30 and 79, so the vertical distance between the low group and the high group is not to scale.

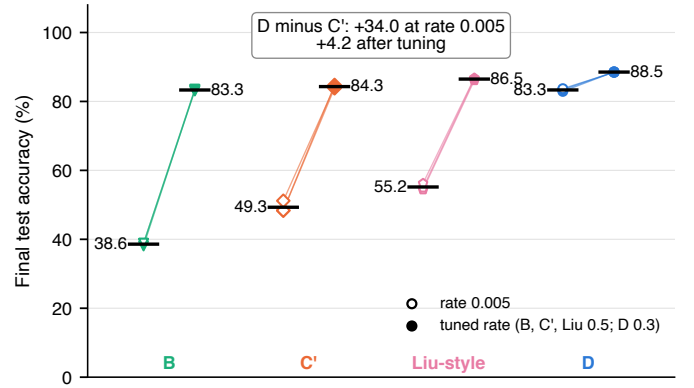


Fig. 3. Untuned versus tuned baselines (ResNet-9, CIFAR-10, $n=3$ seeds: 42, 101, 102). Each point is one seed. The burst learning rate moves from 0.005 (old) to the tuned value. B rises from 38.59 to 83.32 and C' from 49.30 to 84.31 on the mean; D rises from 83.33 to 88.52. Notice that the controller's lead over C' shrinks from 34.03 to 4.21 as the baselines are tuned.

Figure 3 shows the arms before and after tuning. The same pattern holds on VGG6 and ViT-tiny, where tuning gains for B are +44.56 and +24.17 points.

We picked the ResNet-9 rates on seed-42 *test* accuracy, on the same 5,000 images we report. That choice is optimistic. Table VII also gives the choice from the canary set. The largest loss for any arm is 0.45 points, and D's rate does not change. The test pick helped the baselines, not D. D has more settings than the matched rivals, which have only a learning rate: a gate margin, a plateau rule and priors (taken from one seed's checkpoint). This favors D, so the lack of an edge over the rivals is not due to an under-tuned D. VGG6 and ViT-tiny rates were picked on canary accuracy only. Matched arms exist only at rate 0.5, taken from the sweep of the unmatched arms; the rate was not re-swept at matched compute, which may favor or disfavor either side. So the canary-picked matched gaps are projections: D minus B-matched shifts by at most +0.45, and D minus C'-matched by +0.05.

C. The matched rival: pre-specified replication

The strongest fair rival is C'-matched. It has living spares and D's burst steps. On the first five seeds, D was ahead of it by 1.06 points (95% interval +0.5 to +1.6, 5 of 5 seeds). The Holm-adjusted p was 0.056, so we labelled the result consistent, not significant. We also did not quote the sign test, because seeds 103 and 104 were added after our first look. That interim result is the reason we planned E3.

E3 is a pre-specified replication on three new seeds, 105 to 107. The rules were fixed before the runs (Section IV-C). *Success* needs a positive mean, at least 7 of 8 seeds ahead, a Holm p below 0.05, and all three new seeds ahead. *Failure* is a mean below +0.5 or at most 5 of 8 seeds ahead. The outcome is a failure. The per-seed gaps on the new seeds are -3.12, -3.32 and +1.10 points (mean -1.78; D ahead on 1 of 3). Over all eight seeds the mean gap is -0.005, D is ahead on

TABLE VI

MAIN PAIRED CONTRASTS, RESNET-9, CIFAR-10, 15% SPARES (FINAL ACCURACY, POINTS, UNLESS MARKED LIFETIME). PER-SEED VALUES FOLLOW SEED ORDER 42, 101, 102, 103, 104 (AND 105, 106, 107 WHERE $n = 8$). FAM. IS THE HOLM FAMILY: H (HEADLINE, 15), E (REVIEWER BASELINES, 4), S (EXPLORATORY, 64). HOLM p IS WITHIN THAT FAMILY. ROWS IN FAMILY S ARE EXPLORATORY, SO “SIGNIFICANT” THERE IS NOT A HEADLINE CLAIM. D – B-MATCHED IS A SEQUENTIAL DESIGN (SEEDS EXTENDED FROM 3 TO 5 AFTER A FIRST LOOK). THE TWO LIFETIME ROWS ARE EXPLORATORY AND NOT PRE-SPECIFIED; THEIR p VALUES ARE UNADJUSTED. STATUS FOLLOWS SECTION IV. WINS COUNT SEEDS WHERE THE FIRST ARM IS AHEAD. THE 95% INTERVAL (PAIRED t) FOR D – C’-MATCHED, ALL SEEDS, IS -1.69 TO $+1.68$; THE 0.056 FOR THE FIRST FIVE SEEDS IS THE HOLM VALUE IN FAMILY H WITH THAT CONTRAST AT $n = 5$ IN PLACE OF $n = 8$.

Contrast (A minus B)	n	Mean	Per seed	Wins	Fam.	Holm p	Status
D – B-matched	5	+3.78	+4.14, +3.52, +3.68, +4.46, +3.10	5/5	H	0.0014	significant (sequential design)
D – C’-matched, first five seeds	5	+1.06	+1.28, +0.44, +1.18, +1.52, +0.88	5/5	H	0.056	consistent (interim look)
D – C’-matched, new seeds	3	-1.78	-3.12, -3.32, +1.10	1/3	—	—	mixed
D – C’-matched, all seeds (E3)	8	-0.005	+1.28, +0.44, +1.18, +1.52, +0.88, -3.12, -3.32, +1.10	6/8	H	1.000	mixed; failure; not distinguishable (no equivalence test)
E1-last – D	8	-0.55	-1.66, -1.90, -2.06, -1.86, -0.88, +2.02, +2.82, -0.90	2/8 (E1 ahead)	E	0.66	mixed; not distinguishable (no equivalence test)
E2 – D	8	-76.06	-78.20, -79.98, -77.20, -78.42, -77.86, -70.36, -67.12, -79.34	0/8	E	<0.001	negative
E2 – R1	8	+0.67	+0.40, -2.18, +0.94, -0.02, +3.04, +0.36, +3.38, -0.54	5/8	E	0.66	mixed
E1-last – R1	8	+76.18	—	8/8	E	<0.001	significant
C’-matched – B-matched	5	+2.72	+2.86, +3.08, +2.50, +2.94, +2.22	5/5	S	0.004	significant (exploratory family)
D – R1	5	+78.77	—	5/5	S	<0.001	significant (exploratory family)
E1-last – D, lifetime	8	-24.02	-25.37, -25.15, -24.53, -23.21, -24.06, -22.67, -21.79, -25.35	0/8 (E1 ahead)	S	unadj. 3×10^{-10}	exploratory; not pre-specified
E2 – D, lifetime	8	-47.02	-49.56, -48.59, -46.79, -48.21, -46.29, -44.25, -42.46, -50.01	0/8 (E2 ahead)	S	unadj. 3×10^{-10}	exploratory; not pre-specified

TABLE VII

LEARNING-RATE SELECTION, RESNET-9, 3 SEEDS. BOTH PICKS ARE SHOWN. LOSS IS THE 3-SEED ACCURACY LOST BY USING THE CANARY-PICKED RATE INSTEAD OF THE TEST-PICKED RATE.

Arm	Test pick	Canary pick	Loss (pt)
B	0.5	0.3	0.45
C’	0.5	0.3	0.06
D	0.3	0.3	0.00

6 of 8, and the Holm-adjusted p in the recomputed headline family is 1.000. Figure 4 shows all eight seeds.

We draw three conclusions. First, the +1.06 lead is not a claim of this paper, and we do not call it replicated. The five seeds that showed it were the seeds that prompted E3, so the estimate was likely too high. Second, the defensible headline against matched rivals is no detectable difference on 8 seeds, not a proven tie. We ran no equivalence test. The 95% interval for D minus C’-matched is -1.69 to $+1.68$ points (paired t , $n = 8$, SD 2.01; recomputed from the per-seed differences in Table VI). So a deficit or a lead of up to about 1.7 points cannot be excluded. Even the 90% interval used for a two-one-sided

test (± 1.35) is wider than a 1-point margin, so equivalence at 1 point is not shown. The mean of -0.005 mixes two groups: D is ahead on 6 of 8 seeds by 0.44 to 1.52 points, and behind by 3.12 and 3.32 points on the 2 seeds where the gate skipped a burst. That pattern fits “no edge” and fits “D is ahead unless its gate fails” about equally, and we cannot tell them apart. Third, the E3 test is a sequential design with an interim look: three seeds were added after a five-seed analysis. Any p value from the eight-seed set is optimistic, so we print the new seeds alone and the pooled set. The fix for the gate (a fixed pre-deployment reference) is untested, so we can neither claim that D works when fixed nor dismiss it. Recomputing the Holm family with the eight-seed contrast leaves the D versus B-matched result significant ($p = 0.0014$). Seeds 105 to 107 were run on separate hardware batches from seeds 42 to 102, and the same seed gives different fault maps on different GPU models; the GPU model is not recorded in the stored run metadata, so we cannot say whether the two failing seeds share a machine or a fault-map family with each other or differ from the other six.

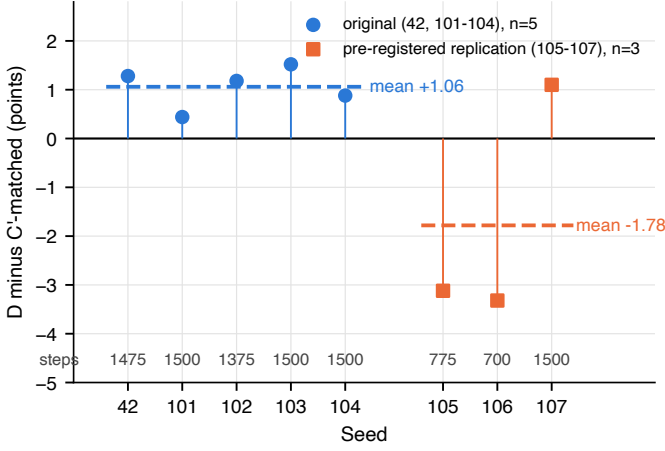


Fig. 4. Per-seed difference D minus C'-matched in final accuracy, all eight seeds. The five original seeds (42, 101–104) gave +1.28, +0.44, +1.18, +1.52, +0.88 (mean +1.06). The three pre-specified new seeds (105, 106, 107; E3) gave -3.12, -3.32, +1.10 (mean -1.78). The eight-seed mean is -0.005, with D ahead on 6 of 8 seeds. The numbers under each seed are D's burst steps (1475, 1500, 1375, 1500, 1500, 775, 700, 1500). Notice that the two strongly negative new seeds are the low-step ones.

D. Reviewer baselines: E1 and E2

Reviewers will ask two questions. Does the closest prior art, adapted to waves, match D? Does a forward-only BatchNorm repair already recover most of the gain? E1 and E2 answer them. Their rules were specified before the runs.

E1 is not distinguishable from D on final accuracy. E1 runs one Liu-style burst at the last wave, with a greedy saliency-ordered remap each wave. Its burst has D's step count and a rate of 0.5 from an earlier plateau; we did not re-sweep it. E1-last minus D is -0.55 points over eight seeds (D ahead on 6 of 8; E1 ahead on seeds 105 and 106), Holm $p = 0.66$. The pre-specified label is mixed: D and E1 cannot be told apart on 8 seeds, and we ran no equivalence test. The result is heterogeneous: D is ahead on 6 seeds by 0.88 to 2.06 points, and E1 is ahead by 2.02 and 2.82 points on seeds 105 and 106, where D's gate skipped a burst. E1 is far above remap-only (+76.18, 8 of 8 seeds, significant), so the sanity check passes. A single burst at the first wave (E1-first) collapses once later waves arrive: finals are 10.08 to 27.88 percent, and the mean is 68.11 points below D (0 of 8 seeds). A burst before the last wave cannot cover later faults. On final accuracy, a simple one-shot retrain at D's compute is not distinguishable from D's paced, gated repair on this test. It does not say that E1 is the better design: E1 has no repair between waves, and it ran with a rate that was not swept.

E1 does not serve through life. Final accuracy is the pre-specified measure for E1, but a deployed chip must serve between waves. We therefore computed lifetime accuracy (mean of the three per-wave accuracies) for E1-last and E2 on the same runs. This analysis is exploratory, not pre-specified, and in family S. E1-last minus D is -24.02 points over 8 seeds (D ahead on 8 of 8). E2 minus D is -47.02 points (D ahead on 8 of 8). The mechanism is plain. E1 repairs wave 0 by greedy remap, then the pool is empty, and with no retraining before the last wave it serves at 10 to 17 percent after wave 1. Its

single last-wave burst brings it back to 83.3 to 87.7 percent. This gap is close to by construction for a one-shot method, and we do not read it as D beating Liu's method. C'-matched also retrains in service, and it is not significantly different from D on lifetime: D minus C'-matched is -0.70 points over 8 seeds (95% interval -1.59 to +0.19; D ahead on 2 of 8, mixed; Section V-G), carried mostly by seeds 105 and 106 (-1.93, -2.77) where D's gate skipped a burst. The lesson is that one terminal retrain matches D on the final number but not on service through life.

E2 stays at chance when the spare pool is exhausted. E2 remaps at random and refits BatchNorm. It takes no optimizer step. Wave 0 is repaired (89.12 and 88.90 percent on seeds 103 and 104). Then the spares run out and accuracy falls. The eight finals range from 8.58 to 13.64 percent, with a mean of 10.58. E2 minus D is -76.06 points (0 of 8 seeds, negative). E2 minus R1 is +0.67 (5 of 8 seeds, Holm $p = 0.66$, mixed): BatchNorm statistics alone do not repair the collapse. The pre-specified rule says that if E2 came within 2 points of D, we must withdraw the claim that bursts are needed. E2 is far outside that range, so the claim stands: burst retraining is required in this scenario. E2 ran at 27.1% cumulative faults and 15% spares, where remap-only is also at chance because the pool runs out. We therefore claim only that BatchNorm refit does not rescue a chip whose spare pool is exhausted (the tested regime). We did not test it at a low fault rate.

E. The gate has a drift failure mode

D lost to C'-matched on seeds 105 and 106 (D 81.24 and 80.76 against 84.36 and 84.08). We read the per-wave logs after the verdicts were fixed. This diagnosis is post-hoc; it changes no verdict, no seed and no gate definition. The canary gate measures the drop against the canary reading after the previous wave's repair. On both seeds the wave-1 burst stopped early at a plateau (275 and 200 steps), and left a weak reference (83.69 and 83.40 percent). At wave 2 the canary dropped by only 1.76 and 0.20 points, below the gate threshold of 2.0, so D skipped the burst and ended near 81 percent. On a typical seed (42) the drop at that wave was 7.62 points and D ran all three bursts (500, 500 and 475 steps). The reference moves down after a weak burst, so the bar moves down with it. We read this as a reference-drift failure mode, seen in 2 of 8 seeds, and report it as a limitation. C'-matched (fixed burst length, no gate) and E1 (one fixed burst) are not exposed to it. D's under-spending also gave the step-matched rivals fewer steps on those seeds, yet they finished 2.0 to 3.3 points higher. The causal reading is a judgment from one trace per seed, not a pre-specified test. A gate that measures against the pre-deployment canary would be a new design, and it needs its own pre-specified test. We did not run it, so the gate fix is untested.

F. The overwrite effect

At the old burst rate (0.005), living spares gained 2.48 points over frozen references (46.82 against 49.30 percent; 3 of 3 seeds; Figure 5). Both arms ended at 47 to 49 percent there, so that rate barely worked, and we report the number only

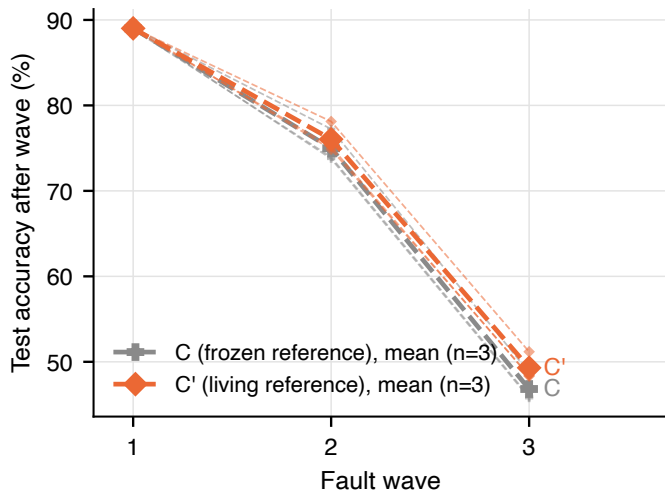


Fig. 5. The overwrite bug. Per-wave accuracy for frozen-reference spares (C) and living spares (C') at the old burst rate ($n=3$ seeds: 42, 101, 102). Final accuracy is 46.82 for C and 49.30 for C' (+2.48; per seed +2.38, +2.64, +2.42; paired t test $p = 0.0011$). This old-rate effect is superseded: at the tuned rate the difference is +0.03 points (noise-level; Section V-F).

as an old-rate result that is superseded. We then ran a pre-specified control at the tuned rate (0.5; 3 seeds; on a different GPU, with fault maps identical across arms within a seed). Living minus frozen final accuracy is +0.03 points (per seed +0.08, 0.00, 0.00), which is noise-level, and the pre-specified decision was NOT SUPPORTED. A third arm that starts the spare from its original value also lies within noise. The lifetime mean accuracy is 0.31 points lower for living spares, on 3 of 3 seeds. Frozen ran 0.42 to 0.96 points above living after the first wave, and the last burst brought the arms together. The mechanism (a remap restores the original reference over a retrained cell) is a code-level observation, and we did not show that it costs accuracy at the tuned rate. The design also cannot test the case it is about. In the capacity policy, every remap happens before the first retraining burst, so the value at fault of each remapped cell equals its original programmed value. The C versus C' contrast (old and new rate) therefore compares only a pinned spare with a trainable one; it never tests overwriting learned compensation. Only D remaps after retraining, and it has no isolated ablation of that step. This result fits the paper's main thesis: a tuned comparison removed an apparent gain, here the living-spares gain, as it removed D's lead over the matched rival. The stuck-weight freeze has no isolated measurement for quantized devices.

G. Spares and lifetime

Remap alone is a capacity cliff. At 2, 5, 10 and 15% spares it ends at 9.9, 11.8, 10.0 and 10.3 percent; at 30% it equals the fault-free chip at 89.02. The cliff sits at the cumulative stuck fraction of 27.1%, by construction. We did not run spare counts between 15 and 30%. On the three seeds with a 30% R1 run, D at 15% spares ends at 88.62 percent, within 0.4 points of the clean chip (89.01; remap-only at 30% is 89.02; interval -1.0 to $+0.2$, 0 of 3 seeds ahead). On all eight seeds D ends at 86.64 percent. D uses about 1,475 burst steps to do

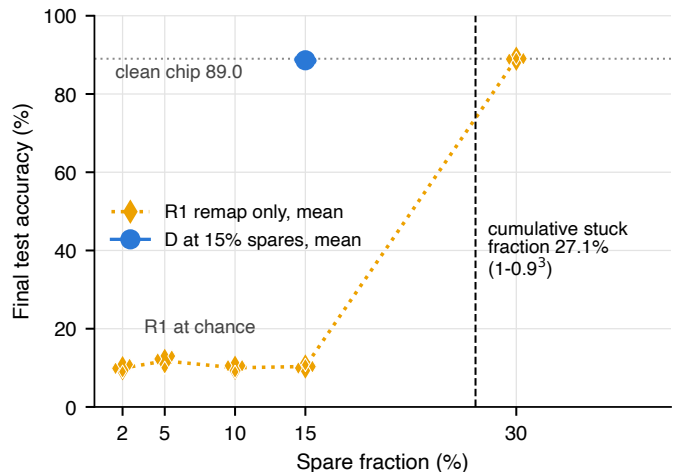


Fig. 6. Spare sweep. Remap only (R1, 3 seeds) is at chance for 2, 5, 10 and 15% spares (9.9, 11.8, 10.0, 10.3 percent) and equals the fault-free chip at 30% (89.02); the cliff is the cumulative stuck fraction of 27.1%, by construction. D at 15% spares ($n=3$) ends at 88.62 percent. Notice that D spends about 1,475 burst steps to do so. The 88.62 value is a 3-seed mean; on eight seeds D ends at 86.64.

this. This is a fixed-schedule result at one network and one schedule, and it trades compute for area. We do not read it as a spare-area saving. Figure 6 shows the sweep. Against C'-matched, D's gap is noise-level at 2% spares (+0.49), mixed at 5% (+0.70), and consistent at 10% (+2.33), all with 3 seeds.

Pacing has a price over the lifetime. Figure 7 shows accuracy after each wave. On the five original seeds, D serves 86.09, 87.44 and 88.50 percent after waves 1, 2 and 3. C'-matched serves 88.92, 86.18 and 87.44. D's lead over C'-matched on those seeds exists only at end of life, and it did not replicate (Section V-C). Averaged over the lifetime, D minus C'-matched is -0.17 points on these five seeds (2 of 5 seeds, mixed) and -0.70 points over all eight seeds (95% interval -1.59 to $+0.19$; D ahead on 2 of 8, mixed), because D's gate skipped a burst on seeds 105 and 106 (-1.93 , -2.77). D minus B-matched is +4.32 points (5 of 5 seeds, Holm 0.0027, significant). Pacing holds spares back, so D misses the strict clean-minus-2 floor after wave 1 in all five seeds. In an ablation with equal steps, pacing adds about 0.7 points (3 seeds, consistent but Holm $p = 1$, confounded: the ablated arm fires two fixed bursts and D fires three adaptive ones). After the E3 result we do not read this as an accuracy gain of pacing or gating over step-matched baselines, with the caveat that the gate tested is the drift-prone one. Pacing is a lifetime trade-off: it lowers early accuracy and did not raise final accuracy over a matched rival as implemented.

H. Several designs, the same final accuracy: what each costs

Table VIII puts the designs that reach about the same final accuracy side by side. Matching D on final accuracy within our resolution (about 1.7 points on the matched rival) is itself a result: it tells a reader that the choice can be made on cost, not on a margin we could not show. The rows differ in what they need and in how they fail, not in a ranking.

Which design to choose depends on what the deployment can afford. If only final accuracy matters, the simplest design

TABLE VIII

DESIGNS THAT REACH THE SAME FINAL ACCURACY AT MATCHED BURST STEPS, AND WHAT EACH COSTS (RESNET-9, CIFAR-10, THREE WAVES OF 10%, 15% SPARES). FINAL ACCURACY IS THE MEAN OVER n SEEDS, IN PERCENT. LIFETIME IS THE MEAN OVER THE THREE PER-WAVE ACCURACIES, GIVEN AS THE ARM MINUS D IN POINTS; THE E1-LAST ROW IS EXPLORATORY AND NOT PRE-SPECIFIED. “SETTINGS” LISTS THE CHOICES A USER MUST MAKE, AS OUR IMPLEMENTATION EXPOSES THEM; IT IS A QUALITATIVE LIST, NOT A MEASURED QUANTITY. THIS IS A COMPARISON OF COSTS AND FAILURE MODES, NOT A RANKING.

Design	Final acc. (n)	Lifetime vs D	Spares?	Canary / held-out set?	Settings to tune	When it retrains	Known failure mode
D (controller)	86.64 (8)	—	Yes, living	Yes (gate and plateau stop)	Several: burst rate, gate threshold, plateau rule, expected waves, BN batches	After a wave, in an adaptive burst, if the canary drops	Gate reference drifted and skipped a needed burst on 2 of 8 seeds (about 3 points behind on those)
C'-matched (capacity trigger)	86.64 (8)	+0.70 ($n = 8$; D ahead on 2 of 8)	Yes, living	No	Burst rate and burst length (set to D's step budget)	In a fixed burst when a wave overflows the spare pool	None observed in the main test; living spares gave +0.03 over frozen at the tuned rate (noise-level)
E1-last (one Liu-style retrain)	86.08 (8)	-24.02 ($n = 8$)	Yes	No	Burst rate and length; which wave	Once, at the last wave	Serves at 10–17% between waves once the pool is empty; D ahead on 6 of 8 seeds, not distinguishable
B-matched (retrain only)	84.72 (5)	-4.32 ($n = 5$)	No	No	Burst rate and length	After every wave, fixed burst	3.78 points below D (5 of 5 seeds); not re-run on seeds 105–107
R1 (remap only)	9.90 (8)	Collapses after wave 1	Yes	No	None	Never	At chance once the pool is exhausted (E2, a BatchNorm refit, does too: 10.58)

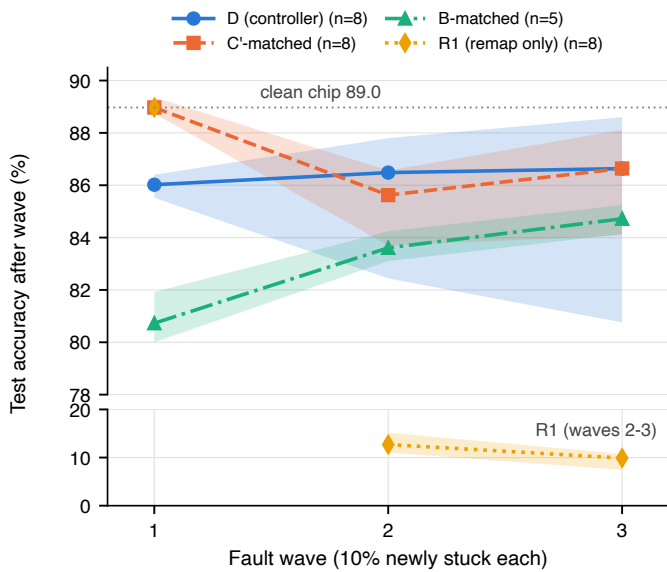


Fig. 7. Accuracy after each wave (mean over seeds; band is the range over seeds). D and C'-matched: eight seeds (D 86.02, 86.48, 86.64; C'-matched 88.97, 85.62, 86.64). B-matched: five seeds (80.73, 83.61, 84.72). R1: eight seeds (88.97, 12.71, 9.90). Notice the trade-off: pacing costs D accuracy after wave 1, and after wave 3 the D and C'-matched means are equal. The five-seed values in the text differ from these eight-seed means.

that matches D is C'-matched: it needs no canary set and no gate, and it is not exposed to the gate drift we found. One terminal retrain (E1-last) is enough only if accuracy in service between waves does not matter, because it serves at 10–17 percent until the last wave. If the lifetime curve matters, retraining in service is needed; C'-matched is 0.70 points higher than D on lifetime over 8 seeds (interval -1.59 to +0.19 for D minus C'-matched), so the two are not significantly different on it. Spares together with in-service retraining are the necessary core: retraining alone ends 3.78 points lower,

and remap alone ends at chance once the pool is exhausted. What D adds is a canary-driven trigger and paced spare use, and at matched burst steps we did not measure that they buy accuracy; the gate also has the drift failure mode above.

I. Other networks, fault patterns and stress cases

Table IX lists the other settings. These rows are separate three- or five-seed runs. E3 was run only on the default ResNet-9 setting, so the eight-seed result does not by itself cover these rows. A three-seed row cannot be significant under our rule ($n \geq 5$ and all seeds in one direction), so these rows are not evidence of absence of an edge; they are also too few seeds to establish one. Against C'-matched, D is ahead on 3 of 3 seeds in 7 of the 10 three-seed rows (all except 5% spares, six small waves and calibrated drift at 5% spares), and behind or mixed on VGG6 and in S4. On CIFAR-100, D's gap over the best C'-matched rate is +7.16 points (3 of 3 seeds, indicative); the rates were picked on test accuracy of seed 42, and the C' rate sweep may overstate the gap. On ViT-tiny the gap is mixed (+1.50, 4 of 5 seeds). On VGG6 it is mixed (-0.52, 3 of 5 seeds): the edge does not reproduce.

D has no edge or trails in three settings. With many small waves and a gate fixed to the best-so-far canary, D trails C'-matched at equal steps: -2.22 (six waves) and -0.41 (eight waves), both mixed. On hardware-calibrated scenarios (S4, small steady drift), D trails B-matched by 0.65 to 3.54 points; the two cases with 5% spares are negative (0 of 3 seeds). We believe, but have not tested, that the canary gate and plateau stop end bursts too early under small steady drift. The seed 105 and 106 traces point to the same family of failure (Section V-E). The drift size and timing in S4 are our assumptions and drive the result. The D2 variant (an adaptive trigger that stops bursts earlier), specified before any

run, addressed this. It passed its calibrated-scenario criterion in 4 of 4 scenarios and failed its main-test criterion, so D stays the controller.

J. What the parts contribute

We ablated D’s parts (3 seeds; no row can be significant). Importance ranking beats random order by 1.65 to 1.73 points, consistent. Ranking is prior art and a design choice here. Fisher saliency is no better than weight magnitude (+0.00 and +0.29, mixed). Dropping the BatchNorm refit changes the result by -0.08 points (mixed, old rate only). Removing the gate and plateau stop costs 0.79 points at the old rate; we never ran this at the tuned rate. We therefore claim none of these parts as a gain. The part that carries weight in our evidence is the retraining bursts (Section V-D); living spares showed no tuned-rate accuracy effect (Section V-F).

K. Closed loop on a probe-estimated fault map

All main results use an oracle fault map. We replaced it with a map estimated from counted, noisy probe reads. The estimator and its pre-specified tests belong to the companion paper (in preparation). Here we use it as a black box. It reads each 256 by 256 tile with random ± 1 probes, decodes a sparse set of stuck cells, and falls back to a full readout when decoding hits a cap. We ran D on the estimated map and on the oracle map with the same seed and the same fault realization (42 pairs, equal fault-map hashes). Pre-specified bars were non-inferiority at -1.0 points (dense) and -0.5 points (sparse).

At 2% read noise, we define the gap as D on the estimated map minus D on the oracle map. The mean gap is -0.04 points in a dense regime and -0.07 in a sparse regime (5 seeds each; ranges -0.42 to $+0.24$ and -0.24 to $+0.02$). Both are above the pre-specified bars of -1.0 and -0.5 points. On the 3-seed pre-specified run, the largest per-seed gap is 0.24 points. One identical oracle rerun differed by 0.24 points, and rerun noise elsewhere has an RMS of 0.46 points, so we cannot separate these gaps from noise. This is not an equivalence test.

Three limits apply. First, the dense regime is a full-readout fallback in 99% of tiles, at 1.12 times the reads of a full readout. It shows that a measured map suffices, not that probing localizes. Only the sparse regime tests probing. Second, the pre-specified reads bar (at most 0.25 of a full readout on every seed) held on the 3-seed run. On five seeds, one seed used 0.2525, so that clause was not met. Third, at 4% noise the sparse extension was killed: the worst-wave recall of new stuck cells was 0.853 to 0.894 against a bar of 0.90. Accuracy still held ($+0.06$ points sparse, $+0.30$ dense). The result is a no-difference-detected result with 3 to 5 seeds, not an equivalence test. It covers stuck-at faults, one tile size and 2% to 4% read noise. It addresses the oracle-map objection only within that scope.

VI. DISCUSSION AND LIMITATIONS

A. What the evidence supports

The evidence supports four statements. First, a remap that restores a frozen reference can erase learned compensation.

At the tuned rate, living spares showed no measurable accuracy effect ($+0.03$ points, 3 seeds, pre-specified, NOT SUPPORTED); the earlier $+2.48$ was an old-rate artefact, and our design cannot test remap after retraining. Second, retraining is needed once the pool is exhausted. A forward-only BatchNorm refit stays at chance on 8 of 8 seeds, 76.06 points below D, in the tested regime only. Third, at equal burst steps and a tuned rate, D beats retraining-only by 3.78 points, significant after Holm correction on five seeds (a sequential design: seeds were extended from 3 to 5 after a first look). Fourth, on default ResNet-9, D is not distinguishable from the best matched living-spares rival (eight seeds, mean -0.005 , 95% CI -1.69 to $+1.68$) or from a one-shot Liu-style retrain at D’s compute (-0.55 , Holm $p = 0.66$) on final accuracy; we ran no equivalence test, and the result is heterogeneous (D ahead on 6 of 8 seeds, behind by about 3 points on 2). The one-shot retrain serves far below D through life (exploratory, -24.02 points over 8 seeds). These statements hold only in simulation, for one controller, one network and one fault model; E3 was run only on default ResNet-9. In other settings D is ahead of the matched rival on 3 of 3 seeds in 7 of 10 three-seed rows, including CIFAR-100 ($+7.16$ points), but 3-seed rows cannot reach significance under our rule, so they neither support nor rule out an edge there.

The most useful result for other groups is the evaluation record. A 34-point lead fell to 4.2 points when we tuned the rival’s rate, to 1.06 when we matched compute, and to -0.005 when a pre-specified replication added three seeds. The living-spares control points the same way: a gain of 2.48 points at an old rate vanished at the tuned rate. For this controller, the margin was not interpretable until it had a tuned baseline at matched burst steps, and not trustworthy until a replication confirmed it: our own five-seed $+1.06$ points did not survive one. We recommend the same checks for any repair-controller claim. This is a recommendation from one measured case, not a measurement of other papers.

B. Why simple matched baselines are not distinguishable from the controller

We did not test why D is not distinguishable from them. Three readings fit the data, and each is a hypothesis. First, once spares are living and the burst has D’s compute, the order and the pace of the remap may matter little. Second, a canary gate that reads a moving reference can skip a burst that a fixed schedule would run (Section V-E). Third, C’-matched and E1 get the same step count as D with no way to under-spend. Our evidence does not rank these readings. Practitioners can take one lesson: spend the retraining compute. Keeping learned weights when a spare is remapped is sensible by construction, but we did not show that it matters for accuracy.

C. Limitations

- **Scope.** The headline is a ResNet-9 / CIFAR-10 result for one controller. Other networks and fault patterns are 3- to 5-seed runs, E3 was not repeated on them, and D is ahead of the matched rival in several of them.

TABLE IX

D AGAINST C'-MATCHED AND B-MATCHED IN OTHER SETTINGS (FINAL ACCURACY, POINTS). SEEDS: 3 UNLESS STATED. STATUS FOLLOWS SECTION IV. RATES FOR CIFAR-100 WERE PICKED ON TEST ACCURACY OF SEED 42. THESE RUNS ARE SEPARATE FROM THE EIGHT-SEED MAIN TEST AND ARE NOT A GENERAL MARGIN OVER C'-MATCHED.

Setting	n	D – C'-matched	D – B-matched	Status (vs C'-matched)
2% spares	3	+0.49 (3/3)	+0.78	noise-level
5% spares	3	+0.70 (2/3)	+2.37	mixed
10% spares	3	+2.33 (3/3)	+3.41	consistent
25% faults per wave	3	+2.14 (3/3)	+3.74	consistent
CIFAR-100	3	+7.16 (3/3)	—	consistent (df=2)
VGG6	5	−0.52 (3/5)	+2.58 (4/5)	mixed
ViT-tiny	5	+1.50 (4/5)	+8.20 (5/5)	mixed (vs C'-matched)
Weibull timing (S1)	3	+0.94 (3/3)	+3.05	consistent
Stuck-at plus drift (S2)	3	+1.11 (3/3)	+3.51	consistent
Line faults (S3)	3	+0.73 (3/3)	+2.79	consistent
Six small waves, gate fixed	3	−2.22 (1/3)	—	mixed
Calibrated drift S4b, 15% spares	3	—	−1.05 (1/3)	mixed
Calibrated drift S4b, 5% spares	3	−4.95 (0/3)	−3.54 (0/3)	negative

- **Simulation only.** Every result is from one simulator (CrossSim) with stuck-at faults injected by us. There is no device measurement. Main runs have no read noise, programming error or drift. The total stuck rate and the HRS to LRS mix in S4 come from one second-hand report; timing is assumed.
- **No measured controller margin over matched rivals on the default setting.** D has no measured margin over C'-matched (eight seeds, mean -0.005 , 95% CI -1.69 to $+1.68$, Holm $p = 1.0$) or over E1-last (Holm $p = 0.66$) on final accuracy; effects of up to about 1.7 points either way are not excluded. We ran no equivalence test, and the results are heterogeneous (D ahead on 6 of 8 seeds, behind by about 3 points on the 2 gate-skip seeds). The five-seed $+1.06$ is an interim-look result that did not replicate. A single-seed gap under 1.22 points is within rerun noise.
- **Retraining-only is a five-seed result.** B-matched was not run on seeds 105 to 107, so the $+3.78$ gap rests on five seeds, extended from three after a first look. D's two weak seeds (105, 106) would lower it; we did not measure how much.
- **The gate has a drift failure mode.** The diagnosis is post-hoc and rests on two seeds. A repaired gate (a fixed pre-deployment reference) was not tested. Canary-gated retraining and the plateau stop have prior art and no measured benefit at the tuned rate.
- **Sequential design.** Seeds 103 and 104 were added after a first look, and E3 added seeds 105 to 107 after a five-seed analysis. D versus B-matched was extended from 3 to 5 seeds after a first look, and its Holm value depends on the 15-member family H (appendix).
- **The network matters.** D's edge over C'-matched is mixed on VGG6 and ViT-tiny. CIFAR-100 and ViT-tiny have three and five seeds.
- **Calibrated drift is negative.** D trails retraining-only in S4 (negative with 5% spares, mixed with 15%), and the cause is an untested hypothesis.
- **Oracle map in the main results.** Closed loop at 2% read noise narrows this; the 4% extension was killed.
- **Controller inputs.** D needs the expected number of waves. Its priors came from one seed's checkpoint, and one priors file moved a single-seed result by 7.7 points.
- **Cost and compute matching.** Matching is on burst steps only. D's saliency, canary and BatchNorm-refit passes are not charged, rivals receive D's step count in hindsight, and the matched arms' rate (0.5) was not re-swept at matched compute. Write endurance is not costed. Labelled data must be available on the device or at the edge.
- **Baselines.** E1's burst rate was not re-swept. An earlier, simpler Liu-style reimplementation (remap, then plain retraining, rate swept) trailed D by 2.03 points on three seeds; it is not the faithful comparison and is not used as evidence. Its 0.3–0.5 rate plateau is only the source of E1's fixed rate. Retraining-free compensation schemes were not run.
- **Selection and provenance.** ResNet-9 rates were picked on test data of seed 42 (first 5,000 test images; both picks printed), and D's settings were developed on the first seeds, which are also evaluation seeds; no held-out seeds were used for tuning. D has more settings than the matched rivals, which have only a learning rate. Pooled seeds mix fault maps from two GPU models; seeds 103 to 104 and 105 to 107 ran as separate batches, the GPU model is not recorded in the stored run metadata, and a hardware effect on the two failing seeds (105, 106) cannot be ruled out. No run records its code commit, and one snapshot commit holds the code used.

D. Next steps

A pre-specified test of a gate that measures against the pre-deployment canary would show whether the drift failure mode costs accuracy. We would also run B-matched on seeds 105 to 107 under a pre-specified plan, sweep E1's burst rate, test the overwrite in a policy that remaps after retraining (the tuned-rate control is done), and run spare counts between 15 and 30%.

VII. CONCLUSION

A repair controller’s margin is only as credible as the baseline it beat. Ours looked 34.0 points ahead of a living-spares rival. Tuning the rival’s learning rate left 4.2 points, matching its burst steps left 1.06, and a pre-specified eight-seed replication left -0.005 (95% CI -1.7 to $+1.7$). Almost all of the apparent win came from an unfair baseline, and measuring that step by step is the contribution of this paper.

What is real stays real. At equal burst steps and tuned rates, D beats retraining-only by 3.78 points in 5 of 5 seeds, significant after Holm correction. A forward-only BatchNorm refit does not rescue a chip whose spare pool is exhausted. Living spares match frozen-reference spares at the tuned rate ($+0.03$ points). On default ResNet-9, D is not distinguishable from the matched living-spares rival or from a one-shot Liu-style retrain on final accuracy, and it serves far better than the one-shot retrain through life; Section VI states every limit, including the untested gate fix. The simulator catalogue and the public code, runs and analysis let others re-derive each number and run the same control on their own controllers. Tuned, step-matched baselines and a replication are the price of a credible margin.

APPENDIX

MEMBERS OF HOLM FAMILY H

Table X lists the 15 comparisons of the headline family H, taken from the stored statistics file. The statistics method, including this family definition, was written before any p -value was computed (results/derived/stats-final.md in the public repository, section 1, “fixed before looking at any p -value”); this is the author’s own record, not an independently timestamped one. The family contains D versus B-matched and D versus C’-matched on the main test, and thirteen further comparisons of D against C’-matched (best-rate C’ on CIFAR-100) in secondary settings, among them VGG6, which is included although it is not a win. Because the family is large, the Holm-adjusted p of every member depends on the others. The main-test D versus C’-matched member uses all 8 seeds, replacing the 5-seed contrast of the first analysis. D versus B-matched was never run on seeds 105 to 107 and not on the stress settings.

Under our rule (Holm below 0.05 and a one-sided sign test below 0.05, which needs $n \geq 5$) none of the 3-seed rows can be called significant even when the Holm value is small (CIFAR-100, S2); they are labelled consistent, not significant.

USE OF AI TOOLS

The author used generative AI tools (Claude) to assist with code and writing; the author takes full responsibility for the content.

AVAILABILITY

Code, configurations, raw run records and analysis scripts: <https://github.com/prateek1656/NeuroFault>. Commit identifiers cited in this paper (018276a, 32aa9ae) refer to the author’s private, timestamped development repository, which is available to reviewers on request.

TABLE X

HOLM FAMILY H (15 COMPARISONS; FINAL ACCURACY, POINTS, PAIRED t). WINS COUNT SEEDS WHERE D IS AHEAD. HOLM p IS WITHIN THE FAMILY.

Contrast (D minus ...)	n	Mean	Wins	Holm p
Main: B-matched	5	+3.78	5/5	0.0014
Main: C’-matched	8	-0.005	6/8	1.000
2% spares: C’-m.	3	+0.49	3/3	0.247
5% spares: C’-m.	3	+0.70	2/3	1.000
10% spares: C’-m.	3	+2.33	3/3	0.325
15% faults/wave: C’-m.	3	+2.69	3/3	0.325
20% faults/wave: C’-m.	3	+1.35	3/3	0.255
25% faults/wave: C’-m.	3	+2.14	3/3	0.065
Continuous regime: C’-m.	3	+1.29	3/3	0.515
CIFAR-100: best C’-m.	3	+7.16	3/3	0.024
VGG6: C’-m.	5	-0.52	3/5	1.000
ViT-tiny: C’-m.	5	+1.50	4/5	0.325
Weibull timing (S1): C’-m.	3	+0.94	3/3	0.515
Stuck-at plus drift (S2): C’-m.	3	+1.11	3/3	0.019
Line faults (S3): C’-m.	3	+0.73	3/3	0.515

REFERENCES

- [1] C. Liu, M. Hu, J. P. Strachan, and H. Li, “Rescuing memristor-based neuromorphic design with high defects,” in *Proc. ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.
- [2] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, “Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar,” in *Proc. Design, Automation and Test in Europe Conference (DATE)*, 2017, pp. 19–24.
- [3] Z. Song, Y. Sun, L. Chen, T. Li, N. Jing, X. Liang, and L. Jiang, “ITT-RNA: Imperfection tolerable training for RRAM-crossbar-based DNN accelerator,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 1, pp. 129–142, 2021.
- [4] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, “Fault-tolerant training enabled by on-line fault detection for RRAM-based neural computing systems,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1611–1624, 2019.
- [5] G. Jung, M. Fouda, S. Lee, J. Lee, A. Eltawil, and F. Kurdahi, “Cost- and dataset-free stuck-at fault mitigation for ReRAM-based deep learning accelerators,” in *Proc. Design, Automation and Test in Europe Conference (DATE)*, 2021, pp. 1733–1738.
- [6] W. Li, Y. Wang, H. Li, and X. Li, “RRAMedy: Protecting ReRAM-based neural network from permanent and soft faults during its lifetime,” in *Proc. IEEE International Conference on Computer Design (ICCD)*, 2019, pp. 91–99.
- [7] S. Jin, S. Pei, and Y. Wang, “On improving fault tolerance of memristor crossbar based neural network designs by target sparsifying,” in *Proc. Design, Automation and Test in Europe Conference (DATE)*, 2020, pp. 91–96.
- [8] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, “Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems,” in *Proc. ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.
- [9] C.-H. Tung, B. K. Joardar, P. P. Pande, J. R. Doppa, H. Li, and K. Chakrabarty, “Dynamic task remapping for reliable CNN training on ReRAM crossbars,” in *Proc. Design, Automation and Test in Europe Conference (DATE)*, 2023, pp. 1–6.
- [10] W. Li, Y. Wang, C. Liu, Y. He, L. Liu, H. Li, and X. Li, “On-line fault protection for ReRAM-based neural networks,” *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 423–437, 2023.
- [11] L. Xia, W. Huangfu, T. Tang, X. Yin, K. Chakrabarty, Y. Xie, Y. Wang, and H. Yang, “Stuck-at fault tolerance in RRAM computing systems,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 102–115, 2018.
- [12] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, “Use ECP, not ECC, for hard failures in resistive memories,” in *Proc. International Symposium on Computer Architecture (ISCA)*, 2010, pp. 141–152.
- [13] S. Khaleghi and W. Rao, “Repairability enhancement of scalable systems with locally shared spares,” arXiv preprint arXiv:1812.01209, 2018.
- [14] M. Sabri, M. Riera, and A. González, “Hamun: An approximate computing method to prolong the lifespan of ReRAM-based accelerators,” *Journal of Systems Architecture*, vol. 166, p. 103444, 2025.

- [15] A. Dorostkar, H. Farbeh, and H. R. Zarandi, "ReSAFT: An efficient stuck-at fault-tolerant scheme for ReRAM-based process-in-memory accelerators," *Future Generation Computer Systems*, vol. 185, p. 108685, 2026.
- [16] H. Shin, M. Kang, and L.-S. Kim, "Fault-free: A fault-resilient deep neural network accelerator based on realistic ReRAM devices," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1039–1044.
- [17] A. R. Bommana, F. Firouzi, C. Ogbogu, B. K. Joardar, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "SEC-CiM: Selective error compensation for ReRAM-based compute-in-memory," in *Proc. IEEE International Test Conference (ITC)*, 2024, pp. 177–186.
- [18] A. R. Bommana, F. Firouzi, C. Ogbogu, B. Joardar, J. Doppa, P. P. Pande, and K. Chakrabarty, "SEC-CiM+: Selective and reconfigurable error compensation for ReRAM-based compute-in-memory," *ACM Trans. Des. Autom. Electron. Syst.*, 2026.
- [19] A. R. Bommana, F. Firouzi, C. Ogbogu, B. K. Joardar, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "DEAR: Dependable 3D architecture for robust DNN training," in *Proc. Design, Automation and Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [20] R. Maharjan and C. Lu, "Training once, deploy anywhere: Defect-robust memristor accelerators without re-training," in *Proc. International Symposium on Quality Electronic Design (ISQED)*, 2026, pp. 1–8.
- [21] Y. A. Khan, Z. Han, and A. Sengupta, "STAR: Astrocyte-inspired state-augmented repair for supervised memristive AI hardware systems," arXiv preprint arXiv:2607.15415, 2026.
- [22] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Gutttag, "What is the state of neural network pruning?" in *Proc. Machine Learning and Systems (MLSys)*, vol. 2, 2020.
- [23] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," in *Int. Conf. Learning Representations (ICLR)*, 2018.
- [24] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs created equal? A large-scale study," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- [25] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith, "Show your work: Improved reporting of experimental results," in *Proc. Conf. Empirical Methods in Natural Language Processing and Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2185–2194.
- [26] X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Sepah, E. Raff, K. Madan, V. Voleti, S. E. Kahou, V. Michalski, D. Serdyuk, T. Arbel, C. Pal, G. Varoquaux, and P. Vincent, "Accounting for variance in machine learning benchmarks," in *Proc. Machine Learning and Systems (MLSys)*, vol. 3, 2021.
- [27] V. Joshi, M. L. Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, p. 2473, 2020.