TC-CRP: BUDGETED INTEGER LEARNING THROUGH ZERO-SUM WEIGHT TRANSFERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Biological synapses have bounded capacity, and their changes are constrained by the units that constitute them. Programmable synaptic devices face corresponding limits in conductance range and write endurance, with each parameter change requiring programming operations. These constraints motivate learning rules that turn desired changes into useful actions within a finite write allowance. We introduce TC-CRP, which converts class-discriminative learning signals into budget-constrained, zero-sum integer weight transfers. Votes identify coordinates worth modifying, and an executor realises their priorities as paired ± 1 updates. Each pair costs two unit writes; a batch quota and a cumulative ledger control parameter modification throughout training. Complete-pair commits preserve integer weights, exact row balance and coordinate bounds, making the write allowance enforceable as learning proceeds. With full unobstructed execution, ranked selection maximises per-row alignment with the learning signal over the prescribed unit-action set, providing a principled basis for allocating limited writes. Because coordinates are bounded and the budget is spent across batches, a batch's proposed changes may require decisions about what to carry forward. Historical accumulation and execution subtraction let later priorities reflect past signals while retaining the same legal actions. At matched write counts on the two primary image tasks, directional selection outperforms a random-coordinate control. TC-CRP thus connects controllable parameter-modification cost with supervised learning and provides a framework for examining how update selection and historical memory use a limited write allowance.

1 INTRODUCTION

Learning from spike timing requires adjusting synaptic weights so that temporal input patterns support reliable decisions. For on-device adaptation, this learning must operate through the parameter changes available to the synaptic implementation: multi-memristive synapses use programming pulses, and analog in-memory training must account for conductance range and device endurance (Boybat et al., 2018; Rasch et al., 2024). Allocating a finite number of writes to useful weight changes is therefore a concrete training problem for classifiers that read spike or event timing.

EventProp provides exact gradients for spiking networks, while NITRO-D develops integer-only neural-network training (Wunderlich & Pehle, 2021; Pirillo et al., 2025). Error feedback carries information across compressed updates, including in constrained optimisation, and residual learning addresses non-ideal pulse responses in resistive devices (Karimireddy et al., 2019; Islamov et al., 2025; Wu et al., 2026). These advances motivate a further question when integer unit steps, row conservation and a cumulative write limit are required together: how should a learning signal select useful parameter changes while satisfying all three requirements throughout training?

To address these issues, we introduce TC-CRP, a supervised learning rule that allocates a finite write allowance through zero-sum integer weight transfers. We study a classifier that reads integer first-spike latencies and assigns classes with linear scores. Its elementary update adds one to one coordinate and subtracts one from another in the same weight row, preserving the row sum at a cost of two unit writes. The rule admits a pair only when both coordinates remain within bounds and the remaining budget covers the complete transfer. This construction makes the modification cost controllable at every update: even a later reversal consumes additional writes, so the budget

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.

accounts for the full learning history. By preserving valid weights at each update, this construction also removes the need for a separate corrective update. The guarantee applies along the learning trajectory, where each completed update must remain usable for the next decision. Because the resource constraint applies to the sequence of modifications, enforcing it requires the write history alongside the final weights.

Within this write allowance, TC-CRP uses class-discriminative votes to choose which coordinates to modify. When the true class fails to beat its strongest competitor, the input contributes positively to the true-class row and negatively to the competing row; aggregated votes then rank coordinates within each active row. Selecting at most k pairs limits each row to $2k$ unit writes per batch, while the cumulative count J controls total parameter modification. Under full unobstructed execution, ranked selection maximises per-row linear alignment with the priority signal over the prescribed unit-action set. Zero row sums additionally make class scores invariant to a common additive shift of the encoded latencies. Thus the same paired action supports directed learning, a controlled write cost and sensitivity to relative encoded timing. The rule turns a potentially dense vote into a sparse set of actual weight changes. A small per-row quota directly limits modification work while allowing all input coordinates to compete for that quota. The optimisation statement explains why the chosen endpoints are useful: among the prescribed unit changes in that setting, they follow the priority signal most closely.

These per-batch choices also raise a temporal allocation problem: a quota, a coordinate bound or the remaining budget can restrict how much of a learning signal is expressed immediately. We therefore compare three ways of assigning priorities within the same action set. The default rule A uses the current signal; CUM carries accumulated centred demand; H2 additionally subtracts committed actions, allowing subsequent priorities to reflect both demand and execution history. All three share the same integer, row-balance, bound and budget guarantees. With the permitted parameter modifications held fixed, these rules let us distinguish the contributions of directional selection, retained signals and executed changes (Section 3.3). Accumulation lets information from earlier batches influence a later choice even when the immediate write quota is small. Execution subtraction distinguishes demand already expressed through committed actions from demand merely retained in memory. Comparing CUM with H2 therefore tests a specific question about learning history: whether using the record of executed changes improves allocation beyond accumulation alone.

We evaluate these choices through matched-write classification comparisons and controlled tests of update selection and memory (Section 5). At the fixed terminal write count, A exceeds the specified random-coordinate control by 76.554 percentage points on MNIST and 66.620 on Fashion-MNIST, demonstrating the value of directional selection. Memory variants attain higher mean accuracy than A on both primary image tasks, on DVS128 Gesture and on N-MNIST; the primary simultaneous intervals for the image-task gains include zero. Shared-vote experiments further isolate how execution subtraction changes action trajectories, showing how the memory rules redirect subsequent writes.

Our contributions are threefold.

- **Learning with a controllable write cost.** TC-CRP allocates a finite parameter-modification allowance through supervised zero-sum unit transfers, enforcing the cumulative limit throughout training (Section 3).
- **Constraint-preserving, directed updates.** Complete pairs retain integer weights, row balance and bounds; ranked selection has an exact alignment objective under full unobstructed execution (Section 4).
- **Evidence on direction and memory.** Matched-write comparisons and shared-vote tests establish the role of directional selection and isolate how historical demand and executed actions shape subsequent updates (Section 5).

2 Related Work

Class-discriminative updates. The max-score update of Crammer and Singer adds x to the true class and subtracts x from the highest-scoring wrong class (Crammer & Singer, 2003). Our supervised vote follows that direction. We freeze weights within a batch, aggregate votes, and pass the

result through a row-wise zero-sum integer unit executor. The analysis therefore addresses batch aggregation, a fixed quota and a history of budget truncation.

Discrete updates and error feedback. signSGD compresses communication by transmitting gradient signs (Bernstein et al., 2018), and NITRO-D combines local error signals with IntegerSGD for integer CNNs (Pirillo et al., 2025). These methods address representation and communication. Our action-selection problem couples integer unit steps, a zero row sum and a bound on historical writes; a compressed update still needs an execution rule when these three requirements apply together.

Error feedback stores compression error for later updates (Karimireddy et al., 2019); sparsified SGD with memory explicitly accumulates and subtracts transmitted updates (Stich et al., 2018). Safe-EF addresses non-smooth constrained optimisation (Islamov et al., 2025), and dual averaging has been combined with EControl for composite objectives (Gao et al., 2025). Error diffusion compensates requested-versus-implemented setpoints in discrete-time control (Bernstein et al., 2016), while residual learning has been studied for pulse-updated resistive devices with limited response granularity (Wu et al., 2026). We examine accumulation and subtraction within an executor with paired zero-sum unit actions, a $V \neq 0$ gate, bound-induced dropping and a run-wide budget prefix. CUM isolates accumulation from execution subtraction; the resulting analysis concerns this action set and its budget.

Temporal coding and event learning. EventProp computes exact gradients with the adjoint method and jump conditions (Wunderlich & Pehle, 2021). We form integer inputs from first-spike or first-event latencies and decide with a single linear score layer, concentrating the study on restricted weight learning. In the DVS Gesture experiment (Amir et al., 2017), each pooled channel contributes its first event within the first second. The task tests this compressed latency representation and its weight updates.

Integer training in spiking networks. I-LIF uses integer-valued training activations and spike-driven inference (Luo et al., 2024); full-integer online training also addresses convolutional and recurrent SNNs (Gomez & Tang, 2025). Integer representation determines numerical values, while a historical unit-modification budget accounts for executed changes. NeuroTrain surveys training signals and locality in SNNs (Caviglia et al., 2026). Our writes stay within a row; vote formation uses labels and competing-class scores, and action selection sorts coordinates within that row.

Cost accounting. The budget J counts committed unit writes, while recorded wall-clock time is reported separately. Hardware energy depends on implementation and activity: an analytical comparison of rate-coded SNN inference with quantised networks finds advantages only in particular spike-rate and time-window regimes once data movement is counted (Yan et al., 2026). Our experiments measure the write count and elapsed time.

TC-CRP brings class-discriminative votes and historical priorities into one explicitly budgeted action sequence. The executor preserves feasibility at every complete commit and gives each selected transfer a known write cost. Ranked selection supplies an exact per-row objective under full unobstructed execution, and shared-vote experiments isolate how execution subtraction changes the trajectory. This organisation connects learning signals to the actions a constrained parameter store is allowed to execute, while letting the experiments identify which choices improve classification.

3 Problem Setup and TC-CRP

Figure 1 connects class-discriminative votes, coordinate ranking and paired weight transfers, with optional memory supplying priorities across batches.

3.1 Parameters, decision rule, and cost

Let a sample be (x_i, y_i) with $x_i \in \mathbb{Z}^d$ and $y_i \in \{1, \ldots, C\}$. The model is a single linear layer: $W \in \mathbb{Z}^{C \times d}$ starts from $W_0 = 0$, every row satisfies $\mathbf{1}^\top W_c = 0$, and $|W_{cj}| \leq B_w$. Prediction takes the unique maximum of $W_c^\top x_i$, counting a tied maximum as an error. TTFS encodes the input latencies; the classifier reads linear scores.

Each action adds one to one coordinate and subtracts one from another coordinate in the same row. Training cost is $J = 2N_{\text{pair}}$, where N_{pair} is the number of complete pairs committed in the

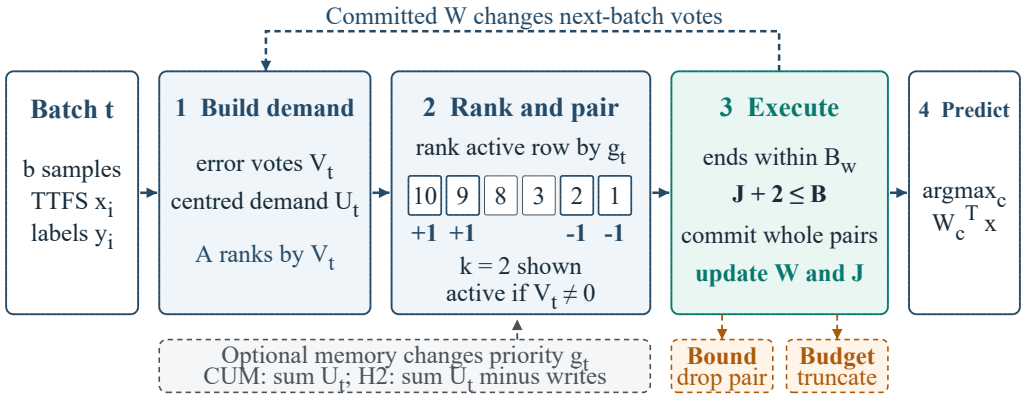


Figure 1: The TC-CRP execution pipeline turns discriminative votes into legal unit updates. Each batch forms a centred demand U, ranks coordinates by priority g, and proposes k pairs with one increment and one decrement each. The executor checks both endpoints and the remaining budget before committing a pair. Bound dropping (1) and budget-prefix truncation (2) determine which proposals survive. The dashed edge traces the effect of committed updates on subsequent votes; the grey block shows optional cross-batch memory.

history, and the even total budget is $\mathcal{B}$. This counts every weight unit modification, including writes later cancelled by a reverse modification. Row-sum conservation allows the weight L_1 norm to grow: $(0,0) \to (1,-1)$ keeps the row sum and increases L_1. Algebraic row sum, terminal weight magnitude and cumulative modification budget thus constrain different aspects of the execution.

3.2 Compact core definition

Fix $1 \le k \le \lfloor d/2 \rfloor$ and $B_w \ge 1$, with all memories initialised to zero. Inside batch t with index set $\mathcal{I}_t$ the pre-batch weights are frozen:

$$s_{ic} = W_c^\top x_i, \quad r_i = \arg\max_{c \neq y_i} s_{ic}, \quad a_i = \mathbf{1}[s_{iy_i} \le s_{ir_i}], \tag{1}$$

$$V_{t,c} = \sum_{i \in \mathcal{I}_t} a_i \left(\mathbf{1}[y_i = c] - \mathbf{1}[r_i = c]\right) x_i, \quad U_{t,c} = dV_{t,c} - (\mathbf{1}^\top V_{t,c})\mathbf{1}. \tag{2}$$

Ties between competing classes take the smallest class index. The memoryless A rule sets $g_t = V_t$; CUM and H2 set $\widetilde{M}_t = M_{t-1} + U_t$ and $g_t = \widetilde{M}_t$. Only rows with $V_{t,c} \neq 0$ act: the coordinates are sorted by g in descending order, with equal values ordered by ascending index, to give π_c, and the rule proposes

$$(p_{c\ell}, n_{c\ell}) = (\pi_{c,\ell}, \pi_{c,d-k+\ell}), \quad \delta_{c\ell} = e_{p_{c\ell}} - e_{n_{c\ell}}, \quad \ell = 1, \dots, k. \tag{3}$$

If either endpoint would leave the bound after its unit step, the whole pair is dropped and no replacement is drawn. The surviving pairs are ordered by class index and then by ℓ, and only a prefix of complete pairs satisfying $J + 2 \le \mathcal{B}$ is committed. Writing ΔW_t for the sum of the committed actions and n_t for the number of committed pairs,

$$W_t = W_{t-1} + \Delta W_t, \quad J_t = J_{t-1} + 2n_t, \qquad M_t = \widetilde{M}_t - \mathbf{1}[\mathrm{H2}]\,\Delta W_t \quad (\mathrm{CUM}, \mathrm{H2}). \tag{4}$$

The demand U of a batch is incorporated exactly once. If the run is saved or reaches an intermediate budget point inside a batch, the original pair list and the commit cursor are retained; resumption continues the commit without recomputing V or adding U again, and H2 subtracts only what was actually committed. Equation (4) describes a complete batch; a partially committed state uses the same $\widetilde{M}_t$ of that batch together with the modifications committed so far.

3.3 The three rules and execution semantics

A is the memoryless default rule of the framework. Its ranking may use either V or U, because (2) only rescales a row by a positive factor and shifts it by a constant, which leaves the coordinate

order unchanged. CUM and H2 rank by their own historical memory, and the subtraction in H2 can change the relative priority of the next batch and, through W, the future votes. The two rules may therefore commit different actions and, in free training, receive different subsequent signals.

"Demand" denotes the integer centred signal of (2). The unit-pair executor determines actual writes. The factor d leaves A's ranking unchanged and sets the relative scale between signal accumulation and unit subtraction in H2; changing it changes the memory rule. We retain this scale throughout the main experiments and examine calibration separately in Appendix I.

The gate and pairing order are part of the algorithm. Only rows with $V \neq 0$ act, even if inactive rows hold memory; a non-zero constant V can produce $U = 0$ and still select pairs by the tie rule. The negative endpoints in (3) retain the original order of the last k entries in the descending list. Under complete execution the endpoint set determines the linear objective; under pair dropping or prefix commits the pairing order determines which actions survive. Fixed class order allocates any limited remaining budget as a class-ordered prefix. These rules align the formulas, experiments and resumable implementation.

4 Theoretical Properties

The properties below treat safe arithmetic and complete pairs as the state-commit boundary; proofs are in Appendix A.

Proposition 1 (feasibility). Starting from zero, after any committed complete pair the weights remain integers, every row sum stays zero, and the bound holds. Inside a batch each coordinate of a row is modified at most once, and each row commits at most k pairs; throughout the run $J \leq \mathcal{B}$. The statement allows in-batch budget truncation and does not guarantee that the budget is reached within a finite number of epochs.

Each pair contains one $+1$ and one -1; the two ends of the sorted list are disjoint, the bound check precedes commitment, and the budget increases in whole pairs. The implementation protects finite integer ranges: weights use int32, memories and centred demands use int64, and float64 scoring stays within the range of exactly representable integer dot products. Proposition 1 establishes integrality and conservation for every committed update state.

Proposition 2 (conditional per-row linear optimality). For a row that satisfies the $V \neq 0$ gate, if no candidate is blocked by the bound and the run is not truncated by the budget prefix, its action $P_k(g)$ satisfies

$$P_k(g) \in \arg \max_{\delta \in \{-1,0,1\}^d,\ \mathbf{1}^\top \delta = 0,\ \|\delta\|_1 \leq 2k} g^\top \delta. \tag{5}$$

Moving a positive endpoint to a larger g and a negative endpoint to a smaller g weakly increases the objective; with $k \leq d/2$, filling up to k pairs does likewise. The statement holds for A's current-batch signal and for the respective memory signals of CUM and H2. On the action set with exactly k pairs, every action has squared norm $2k$, so maximising the inner product also minimises Euclidean distance to g. Allowing fewer pairs changes the action norm and the distance objective.

Bound-induced dropping and budget prefixes preserve Proposition 1 while changing the selection problem: Appendix A gives a case where the committed action gains 8 and another feasible action gains 9. Proposition 2 characterises unobstructed per-row unit selection. The experiments examine how those local choices affect prediction across batches.

Proposition 3 (memory identity). Let D be the sum of the U of all batches that have started, including the current batch whose signal has been incorporated but whose pairs are not fully committed. On each trajectory,

$$M_{\mathrm{CUM}} = D, \qquad M_{\mathrm{H2}} = D - (W - W_0). \tag{6}$$

The identity follows by induction: starting a batch adds the same U to D and to memory, while every committed action adds δ to W and subtracts δ from H2 memory. Each identity refers to its own training trajectory, whose accumulated demand D can differ between CUM and H2. Thus subtraction has an exact ledger interpretation; its classification effect is measured separately.

5 EXPERIMENTS

5.1 COMPARISON QUESTIONS AND COMMON SETUP

The experiments test constraint preservation, the classification effect of directional selection, and the effects of accumulation and execution subtraction. We audit committed states and the conditions of Proposition 2 (Section 5.2), isolate ranked selection with a random-coordinate control at equal J (Section 5.3), and test execution subtraction under shared votes (Section 5.4). Fixed-J classification results and same-feature references then place these mechanisms in task context (Appendices C and F). Real-event supplements and separate calibration diagnostics probe their behaviour in additional settings.

A/CUM/H2 all start from zero weights, and the memory rules start from zero memory; batch size is 128, $k = 128$, the margin threshold is 1, $B_w = 10^9$, and tied predictions count as errors. The image tasks use $d = 784$ and $C = 10$; the event task uses $d = 2048$ and $C = 11$ for DVS128 Gesture and $d = 2312$ and $C = 10$ for N-MNIST. We freeze the encoding, the executor and the decision rule, and vary only the ranking signal: the current-batch vote of A, the accumulated memory of CUM, and the execution-subtracted memory of H2. Accuracy is reported in percent, differences in percentage points (pp), and $\pm$ denotes the sample standard deviation across seeds. Original files and code entries are listed in Appendix D, and per-seed primary values are in Appendix C.

MNIST (LeCun et al., 1998) and Fashion-MNIST (Xiao et al., 2017) both use integer TTFS inputs. The inherited cache provides a pool of 59000 samples; each seed then splits it into 53100 training and 5900 validation samples, with a test set of 10000. The five seeds are 101, 211, 307, 401 and 503, and they change both the split and the training order, while initialisation stays at zero. Within a seed the three methods share the split and the order-generation rule.

The pre-specified validation points are $J = 3187200$, 6374400 and 12748800. The primary result uses the fixed test endpoint at the largest J; all 30 training cells reached it, and each was tested once after training. The test set was also used in earlier work of this project, so the results are comparisons under a fixed protocol.

An equal J matches the number of committed weight-unit modifications. Active rows and legal pairs vary across rules, so sample exposure and computation vary by the time each reaches that endpoint (Appendix C).

Beyond these primary comparisons, Appendix E tests selection temperature, CIFAR-10 inputs, smaller weight bounds, new seeds, larger write budgets and the less restricted H1 rule. Five fresh seeds give CUM$-$A differences of $+0.532$ and $+2.342$ pp, with descriptive 95% intervals excluding zero. On CIFAR-10, mean accuracy is 17.38% for A, 16.15% for CUM and 16.53% for H2. Smaller bounds preserve feasibility while the benefit of subtraction varies with configuration. The temperature study probes cumulative memory, and H1 supplies an unpaired capability reference.

5.2 BUDGET VALIDITY AND CAPACITY AUDIT

Because the bound is part of the model, we also audited the saved weights of all 99 endpoints (90 image validation checkpoints from three arms, five seeds and three budget points, plus 9 event endpoints). The configured bound is $B_w = 10^9$. The largest absolute weight anywhere is 5467 on the image tasks and 61 on the event task, which is 5.5×10^{-6} and 6.1×10^{-8} of the bound respectively; no coordinate reaches the bound in any saved state, and every row sum is exactly zero.

A run-wide argument complements the snapshots: each started batch changes any coordinate by at most one unit, and the maximum recorded batch counts are 5554 for images and 66 for events. With resumption protected against recommitting a batch by the core's synthetic checks, the configured bound stays inactive in these runs. Appendix E.3 separately evaluates binding bounds.

The same audit recomputed all 90 existing validation points from saved predictions and labels. Appendix C reports the curve. Memory rules have a positive mean advantage over A at all three points, with a shrinking gap. The H2-versus-CUM ordering reverses between tasks at the largest point (-0.410, -0.146 and $+0.512$ pp on MNIST against $+1.315$, $+0.397$ and -1.169 pp on Fashion-MNIST). The fixed test endpoint remains the primary comparison.

Table 1: Random-pairing control at the fixed endpoint $J = 12{,}748{,}800$. RAND uses five seeds and the frozen A endpoints serve as the reference arm; differences are $100 \times (\text{A} - \text{RAND})$ in percentage points, and SD is the sample standard deviation across the five seeds. Each task family uses a 97.5% two-sided paired t interval with four degrees of freedom, so the two comparisons cover at least 95% jointly under the paired t model.

Task	RAND (%)	A (%)	Mean diff. (pp)	SD (pp)	Simultaneous (pp)
MNIST	11.558 ± 2.518	88.112	$+76.554$	2.495	$[72.654, 80.454]$
Fashion-MNIST	9.612 ± 6.621	76.232	$+66.620$	5.914	$[57.375, 75.865]$

Table 2: Mechanical comparison under a shared exogenous V. "Different steps" means that the weights or the pair list differ at that step; it does not mean that all components of the 31 states differ.

Task	Batches	First step	Differing steps	Terminal L_1 difference
MNIST	32	identical	31/32	4008
Fashion-MNIST	32	identical	31/32	604

5.3 Random-pairing control

RAND changes A's coordinate selection while retaining its vote-generation rule, $V \neq 0$ gate, bound checks and budget accounting. The resulting weight trajectories generate different numerical votes. In each active row, in ascending class order, RAND draws a uniform permutation ρ from a dedicated CPU generator seeded by $1000003 +$ run seed and proposes $(\rho_\ell, \rho_{d-k+\ell})$, $\ell = 1, \ldots, k$. Blocked pairs are dropped, and the remaining budget admits a prefix of complete pairs. RAND selects coordinates independently of vote magnitude and stores no memory.

RAND isolates the contribution of directional ranking by replacing A's coordinate order with a uniform random permutation while retaining the same executor. At equal J, the comparison measures the classification effect of these two complete policies, including the future votes generated by their evolving weights. Table 1 shows that A outperforms RAND on both tasks at equal J. Both simultaneous lower bounds exceed zero and the pre-set 0.5 pp interpretation threshold. Directional selection therefore has a substantial effect in the tested setting.

All RAND cells committed the full quota and traversed 637200 samples in 4980 vote batches (12 epochs), compared with A's mean traversals of 685507.2 and 659702.4 on the two tasks. The contrast matches committed modifications while exposing the policies to different sample counts. Some RAND cells score below 10%. The control was designed after A's results were known; the six primary memory contrasts retain their original protocol.

5.4 Shared votes: the mechanical effect of subtraction

We feed CUM and H2 the same 32-batch vote sequence generated by A with seed 2026. Each task uses one sequence of 128-sample batches. Holding V fixed isolates changes in actions and parameter states caused by the memory update. Table 2 isolates the mechanical effect of subtraction: identical vote inputs and pair counts lead CUM and H2 to different terminal states. Both commit 40832 pairs, or $J = 81664$, on each task. The shared-vote replay therefore establishes a difference in executed trajectories, while the free-training experiments assess classification.

5.5 Memory under an equal write budget

At the fixed endpoint $J = 12{,}748{,}800$, both memory rules have higher mean accuracy than A on both image tasks: CUM$-$A is $+0.342$ pp on MNIST and $+3.066$ pp on Fashion-MNIST, and H2$-$A is $+0.764$ and $+1.870$ pp. The four primary simultaneous intervals include zero, while H2$-$CUM changes sign across tasks ($+0.422$ and -1.196 pp). Appendix C reports the full panel (Table 4) and per-seed differences (Figure 2). Five fresh seeds in Appendix E.4 add descriptive CUM$-$A intervals that exclude zero. Together, these results show a task-dependent mean benefit of memory and continuing uncertainty under the primary simultaneous analysis.

Table 3: Test accuracy at the fixed endpoint: mean ± sample standard deviation across five seeds (%). MNIST and Fashion-MNIST are image tasks at $J = 12{,}748{,}800$. N-MNIST is an event task with $d = 2312$, evaluated at its own reachable endpoint $J = 14{,}407{,}680$.

Method	MNIST	Fashion-MNIST	N-MNIST
A	88.112 ± 0.474	76.232 ± 1.590	76.90 ± 1.10
CUM	88.454 ± 0.379	79.298 ± 0.440	87.07 ± 0.55
H2	88.876 ± 0.546	78.102 ± 0.954	87.82 ± 0.30

Table 3 adds N-MNIST, an event task evaluated under the same restricted action set at its own reachable endpoint $J = 14{,}407{,}680$. Every contrast excludes zero there: CUM−A is $+10.17$ pp with a descriptive interval of $[8.33, 12.01]$, H2−A is $+10.92$ pp $[9.45, 12.39]$, and H2−CUM is $+0.75$ pp $[0.06, 1.45]$, with all five seeds agreeing in sign within each contrast. These intervals describe order randomness under one fixed split and carry no multiplicity adjustment, so we read them neither as a general advantage of memory nor as evidence that subtraction is ineffective. Appendix H gives the encoding, split and budget.

5.6 Real events: a minimal supplement on DVS128 Gesture

The event supplement in Appendix G extends the comparison to DVS128 Gesture. At $J = 168{,}960$, three-seed means are $36.458 \pm 2.842\%$ for A, $40.394 \pm 1.446\%$ for CUM and $41.435 \pm 1.912\%$ for H2; a single real-valued row-zero-sum ridge fit reaches 31.944%. Memory again raises mean accuracy, with broad descriptive 95% paired intervals that all cross zero.

5.7 Same-feature nearest neighbours: a separate best-validation panel

The same-feature comparison evaluates A, CUM and H2 alongside a zero-sum floating-point linear rule and a multiclass perceptron. It covers two tasks and three seeds (2026, 1, 2); all methods share features, seed-specific splits, the unique-maximum decision rule and a single layer with 7840 parameters. The metric is best validation accuracy within 20 epochs, with 20 validation evaluations and zero test evaluations per cell. Historical modification counts vary across methods.

The three-seed means are listed in Appendix F. The zero-sum floating-point linear reference reaches 90.22 on MNIST and 82.80 on Fashion-MNIST, and the multiclass perceptron reaches 90.45 and 84.26. Both exceed the three unit-pairing rules (A 87.98 and 78.69, CUM 88.61 and 81.16, H2 88.66 and 81.35). The former carries the historical code name `float_linear_free` and centres its weight rows. The panel locates TC-CRP among same-feature alternatives; its different update magnitudes, execution rules and write costs motivate the fixed-J analysis of the proposed mechanism. The calibrated variants, which separate target construction from tracking and classification, are analysed in Appendix I.

6 Discussion and Conclusion

TC-CRP makes the write allowance an operational part of supervised learning. Each accepted pair changes two integer coordinates, preserves their row sum and has a known cost in the cumulative ledger. Bounding and budget checks precede commitment, so every completed update keeps the model inside its execution constraints. Ranked selection then directs the available transfers according to the learning signal, with an exact per-row alignment result under full unobstructed execution.

Directional ranking improves accuracy against RAND at equal J, supporting the use of the learning signal to allocate limited writes. Cumulative memory and execution subtraction retain the same feasible actions while changing their future priorities. Shared-vote experiments verify that subtraction changes trajectories, and fixed-budget classification comparisons measure the resulting task-dependent effects. These results give a concrete framework for studying how a restricted sequence of parameter writes can support learning.

Scope and outlook. The established cost guarantee concerns logical unit writes; the current implementation also performs scoring, voting and sorting, and no hardware energy or general speed

advantage has been measured. Signed row balance is a modelling choice for the encoded-input score invariant, and a logical pair does not establish physical atomicity on a device. Bound blocking and budget prefixes retain feasibility but delimit the alignment optimum. The primary intervals do not establish consistent additional accuracy gains from execution subtraction, and calibration diagnostics show that closer target tracking need not improve classification. The present first-event linear readout and small event-seed set motivate further work on richer representations, broader task coverage, hardware measurement and selectors that adapt to binding bounds and remaining budgets.

The central outcome is a learning rule in which the permitted cost of parameter modification is explicit and enforceable throughout training. TC-CRP specifies the legal action, accounts for every committed write and uses supervised priorities to choose where that allowance goes. This provides a basis for developing learners whose adaptation must operate within a prescribed write budget.

AI USAGE STATEMENT

Generative AI tools and coding agents assisted implementation, experimental design, literature retrieval, mathematical analysis, record checking, interpretation, drafting, translation, typesetting and figure-generation code. They also executed authorized experiment scripts and helped recompute statistics from saved outputs. Reported empirical measurements derive from recorded program outputs and datasets; this revision preserves the historical raw measurements. The authors reviewed the code, evidence, claims and citations and retain responsibility for the submitted content.

ETHICS STATEMENT

This work studies constrained learning on public benchmark datasets (MNIST, Fashion-MNIST, CIFAR-10, DVS128 Gesture and N-MNIST). The study uses existing recordings of human gestures under the published DVS128 Gesture protocol and involves no deployment. Small samples and protocol changes call for care when interpreting accuracy differences. We report the budget J as a count of committed unit writes and wall-clock time as a separate measurement.

REPRODUCIBILITY STATEMENT

The algorithm is fully defined by (1)–(4) together with the gate, the dropping rule, the commit order and the resumption rule; Appendices A–B give the proofs, counterexamples, encodings, splits and arithmetic conditions. Fixed protocols, source snapshots, per-cell records, checkpoint hashes and the audit entries are listed in Appendix D. Synthetic CPU checks cover the invariants, the bound dropping, the budget prefix, the resumption semantics and the arithmetic guards. The primary image comparison uses five seeds with a shared split per seed, a single fixed test endpoint and one test evaluation per cell; the event supplement and the nearest-neighbour panel follow their own frozen protocols and are reported separately.

The accompanying release supplies route-B source, synthetic checks, summary records and calibration evidence. Together with the paper, the released core specifies the algorithm; each reported number traces to a per-cell record and checkpoint hash, and the synthetic CPU checks run with the supplied inputs. Datasets, checkpoints and saved predictions remain in the project archive. The README documents a portable-loader hash mismatch with the historical training gate that affects end-to-end reproduction of the reported accuracies.

REFERENCES

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7243–7252, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Amir_A_Low_Power_CVPR_2017_paper.html.

Andrey Bernstein, Niek J. Bouman, and Jean-Yves Le Boudec. Real-time minimization of average error in the presence of uncertainty and convexification of feasible sets, 2016. URL https://arxiv.org/abs/1612.07287. arXiv:1612.07287.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, volume 80 of *PMLR*, pp. 560–569, 2018. URL https://proceedings.mlr.press/v80/bernstein18a.html.

Irem Boybat, Manuel Le Gallo, S. R. Nandakumar, Timoleon Moraitis, Thomas Parnell, Tomas Tuma, Bipin Rajendran, Yusuf Leblebici, Abu Sebastian, and Evangelos Eleftheriou. Neuromorphic computing with multi-memristive synapses. *Nature Communications*, 9:2514, 2018. doi: 10.1038/s41467-018-04933-y.

Alessio Caviglia, Filippo Marostica, Roberta Bardini, Alessandro Savino, and Stefano Di Carlo. NeuroTrain: Surveying local learning rules for spiking neural networks with an open benchmarking framework, 2026. URL https://arxiv.org/abs/2605.15058. arXiv:2605.15058.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003. URL https://jmlr.org/papers/volume3/crammer03a/crammer03a.pdf.

Yuan Gao, Anton Rodomanov, Jeremy Rack, and Sebastian Stich. Composite optimization with error feedback: the dual averaging approach, 2025. URL https://arxiv.org/abs/2510.03507. arXiv:2510.03507.

Ismael Gomez and Guangzhi Tang. Full integer arithmetic online training for spiking neural networks. In *International Conference on Artificial Neural Networks*, 2025. URL https://arxiv.org/abs/2509.06636. arXiv:2509.06636.

Rustem Islamov, Yarden As, and Ilyas Fatkhullin. Safe-EF: Error feedback for non-smooth constrained optimization. In *International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 26542–26585, 2025. URL https://proceedings.mlr.press/v267/islamov25a.html.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *International Conference on Machine Learning*, volume 97 of *PMLR*, pp. 3252–3261, 2019. URL https://proceedings.mlr.press/v97/karimireddy19a.html.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. URL https://bottou.org/papers/lecun-98h.

Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, 2024. URL https://arxiv.org/abs/2407.20708. arXiv:2407.20708.

Garrick Orchard, Ajay Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9:437, 2015. doi: 10.3389/fnins.2015.00437.

Alberto Pirillo, Luca Colombo, and Manuel Roveri. NITRO-D: Native integer-only training of deep convolutional neural networks. In *European Conference on Artificial Intelligence*, 2025. doi: 10.3233/FAIA250855. URL https://arxiv.org/abs/2407.11698v3. Author version: arXiv:2407.11698v3.

Malte J. Rasch, Fabio Carta, Omobayode Fagbohungbe, and Tayfun Gokmen. Fast and robust analog in-memory deep neural network training. *Nature Communications*, 15:7133, 2024. doi: 10.1038/s41467-024-51221-z.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1809.07599.

Zhaoxian Wu, Quan Xiao, Tayfun Gokmen, Omobayode Fagbohungbe, and Tianyi Chen. Analog in-memory training on general non-ideal resistive elements: The impact of response functions, 2026. URL https://arxiv.org/abs/2502.06309v5. arXiv:2502.06309v5; first posted 2025.

Timo C. Wunderlich and Christian Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 2021. doi: 10.1038/s41598-021-91786-z. URL https://arxiv.org/abs/2009.08378v3.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747. arXiv:1708.07747.

Zhanglu Yan, Zhenyu Bai, Kaiwen Tang, and Weng-Fai Wong. Reconsidering the energy efficiency of spiking neural networks inference from analytical perspectives, 2026. URL https://arxiv.org/abs/2409.08290v6. arXiv:2409.08290v6.

A Proofs and Counterexamples

A.1 Feasibility

The initial state satisfies integrality, the zero row sum and the bound. For any legal pair (p, n) with $p \neq n$, the action $e_p - e_n$ is integer and its components sum to zero. Both new values are checked to be inside the bound before the commit, so the bound still holds afterwards; induction over committed pairs gives the parameter invariants. Since the first k and the last k positions do not overlap when $2k \leq d$, and each endpoint appears once in the list, each coordinate of a row takes part in at most one pair per batch. Dropping pairs or committing a prefix does not break this property. Every pair adds 2 to J, and a pair is committed only when $J + 2 \leq \mathcal{B}$, so the budget invariant holds. Resumption continues along the same list cursor and cannot repeat a committed pair or incorporate U again.

Proposition 1 concerns the API state after each complete pair commit. A physical implementation must manage the transient state between its two writes. When active rows, legal actions or available epochs are exhausted, J can remain below the budget; the invariant is the upper bound $J \leq \mathcal{B}$.

A.2 Ranking optimality and the Euclidean reading

Any feasible δ has the same number m of $+1$ and -1 entries with $m \leq k$. For a given m, exchanging endpoints lets the positive endpoints take the m largest entries and the negative endpoints the m smallest, which yields

$$F(m) = \sum_{\ell=1}^{m} g_{\pi_\ell} - \sum_{\ell=d-m+1}^{d} g_{\pi_\ell}.$$

When $m < k \leq d/2$, $F(m+1) - F(m) = g_{\pi_{m+1}} - g_{\pi_{d-m}} \geq 0$. The maximum is attained at $m = k$, and the complete endpoint set of (3) realises it. The stable tie rule selects one solution. With exactly k executed pairs, $\|\delta\|_2^2 = 2k$, so minimising $\|g - \delta\|_2^2$ is equivalent to maximising the inner product. With at most k pairs, the action norm varies: for a non-zero constant V giving $g = U = 0$, the zero action is nearest to g while k forced pairs still attain the linear objective. The distance interpretation therefore applies to the fixed-k action set.

A.3 Two execution conditions that break optimality

Use one-based indices and let $g = (10, 9, 8, 3, 2, 1)$, $d = 6$ and $k = 2$. The frozen pairs are $(1, 5)$ and $(2, 6)$, and each has linear gain 8.

Bound-induced dropping. Let $B_w = 1$ and $W = (0, 0, 1, 0, -1, 0)$, whose row sum is zero. The negative endpoint of pair $(1, 5)$ would leave the bound, so it is dropped and only $(2, 6)$ with gain

8 remains. Pair $(1, 6)$ is legal with gain 9. Over the action set that allows at most two pairs and respects this bound, the frozen dropping result is not optimal.

Budget prefix. With one pair of budget remaining and no bound obstruction, the executor commits the first pair $(1, 5)$ with gain 8; the feasible pair $(1, 6)$ would give 9. Re-solving for the reduced quota therefore changes the selected action. This counterexample identifies the budget condition in Proposition 2.

A.4 Memory identity

When a batch starts, D and the CUM/H2 memory receive the same U of that trajectory while W is unchanged. Each committed action δ adds δ to W, and only the H2 memory subtracts δ. By induction from $D = M = W = W_0 = 0$, every complete commit state satisfies (6). Saving and pausing do not change these quantities, and resumption does not add U again, so partially committed states also satisfy it. In two free runs, D is determined by different votes produced by different weights, so the identity can only be used trajectory by trajectory.

B Encoding, Data and Implementation Details

B.1 Image features and splits

Image greyscale values are integers $v \in \{0, \dots, 255\}$ with $T = 1024$ and 256 quantisation levels, and brighter pixels fire earlier. For $v > 0$, let the segment index be $h = 255 - v$ and take the midpoint of the integer interval $[4h, 4h + 3]$ rounded down, giving $x = 4(255 - v) + 1$; $v = 0$ uses the silent sentinel 1024. The fixed `quantized_latency_first` encoding flattens each image to 784 int32 first-spike latencies, which feed the single linear layer directly.

The inherited cache excludes the 1000 samples that a historical loader with seed 1701 held out from the original training set, leaving 59000 samples. A new seed uses the first 5900 indices of a CPU `randperm` for validation and the remaining 53100 for training; another generator with the same seed produces a fresh training permutation each epoch. The three methods share the split and order rule within a seed, and each stops at its own J, giving different prefix lengths. The training cap is 20 epochs. The 90 validation checkpoints trace the budget curve; the test endpoint was fixed in advance.

B.2 Event features and ridge

The training archive contains 98 raw recordings and the test archive 24, and all labelled rows of the 122 recordings were converted into 1176 training and 288 test segments. Subject identifiers 1–23 are used for training and 24–29 for testing; class 11 (other) is retained, and so are the duplicated labelled segments of class 8. The test set contains 48 segments of class 8 and 24 of each of the other ten classes, and no segment was removed or rebalanced.

For an annotation start t_0 we consider events in the window $[t_0, \min(t_{\text{end}}, t_0 + 10^6))$ microseconds. A coordinate (u, v, p) maps to $(\lfloor u/4 \rfloor, \lfloor v/4 \rfloor, p)$. For each channel we take the smallest actual timestamp $t_{\min}$ and encode $\lfloor 1024(t_{\min} - t_0)/10^6 \rfloor$; a channel without events is set to 1024. The minimum timestamp defines the first event even when the file contains local timestamp inversions. Polarity occupies separate channels. Each channel contributes one latency feature.

For the ridge reference, training features are $\Phi_i = (x_i - \bar{x}_i \mathbf{1})/1024$, the target Y is one-hot, and the solution is $\alpha = (\Phi\Phi^\top + I)^{-1} Y$ with $W = (\Phi^\top \alpha)^\top$, after which every row is centred. The fixed choice $\lambda = 1$ and zero bias use the original protocol. With raw inputs at prediction time, a global positive factor and a zero row sum preserve the argmax. The floating-point row-sum check uses a numerical tolerance.

B.3 Arithmetic, checkpoints and resource accounting

The reference core checks int64 demand and memory accumulation before execution to prevent silent overflow. Exact float64 dot products follow when inputs and weights are exactly representable integers and $d \max|x| \max|W| \le 2^{53}$. At $\max|x| = 1024$ and $B_w = 10^9$, the image and event

Table 4: All six pre-specified paired differences, in percentage points. Each entry uses a 99.1667% paired t interval with a Bonferroni correction over six contrasts; under the paired t model the family covers at least 95%.

Task	Contrast	Mean (pp)	Bonferroni simultaneous interval (pp)
MNIST	CUM−A	$+0.342$	$[-0.340, +1.024]$
MNIST	H2−A	$+0.764$	$[-1.274, +2.802]$
MNIST	H2−CUM	$+0.422$	$[-1.158, +2.002]$
Fashion-MNIST	CUM−A	$+3.066$	$[-0.163, +6.295]$
Fashion-MNIST	H2−A	$+1.870$	$[-2.999, +6.739]$
Fashion-MNIST	H2−CUM	-1.196	$[-3.067, +0.675]$

upper bounds are 8.02816×10^{14} and 2.097152×10^{15}, both below 2^{53}. These checks certify the arithmetic configuration used here.

Saved state includes W, memory, J, the pending pairs and cursor, the data permutation and the random number state. If a budget point falls inside a batch, `samples_seen` already counts every sample that formed the votes of that batch even when part of the actions are not yet committed. Resumption first consumes the pending actions and does not reread that batch. Source snapshots, protocols and checkpoint hashes make the algorithm identity of each reported number traceable.

Run records come from a single NVIDIA GeForce RTX 4050 Laptop GPU with a CPU pairing core, in WSL with Python 3.8.5 and PyTorch 2.4.1+cu121. The complete image training, validation and test window was 729.155 s; the event v1 stopping window was 2.892 s and the v2 window 5.989 s; the random-pairing control used one window of 198.949 s. Their preparation and saving boundaries differ, and the event windows include exact reuse. We report these observed durations alongside the committed-write counts.

B.4 Update rules of the same-feature neighbours

The floating-point linear reference parameterises a zero row sum as $W_c = P_c - \mathrm{mean}(P_c)\mathbf{1}$ with P starting at zero, uses SGD with learning rate 0.001, and minimises a batch-mean multiclass hinge $\max(0, 1 + \max_{c \neq y} s_c - s_y)$. The multiclass perceptron reference starts from zero W, forms the same class-discriminative votes, and applies $W_c \leftarrow W_c + V_c$ directly on active rows. Both use batch size 128 and the best validation within 20 epochs, with no additional tuning for this manuscript. Their shared input, architecture and decision rule position TC-CRP against methods with different execution rules.

C Raw Values, Statistics and Cost

Table 5 lists the per-seed image endpoints, Table 6 the image cost fields, Table 7 the per-seed event values, Table 8 the per-seed nearest-neighbour validation values, Table 9 the capacity audit, Table 10 the existing validation curve and Table 11 the per-seed random-pairing results.

C.1 Memory variants in the fixed-budget comparison

Table 3 reports fixed-endpoint test accuracy. On the two image tasks both memory rules have a higher mean than A, and CUM−A is positive in all ten task–seed pairs, with a larger average difference on Fashion-MNIST. The event task in the third column is discussed in Section 5.5 and Appendix H. Figure 2 shows the paired differences and the pre-specified simultaneous intervals. Figure 2 shows all six contrasts. The four simultaneous memory-versus-A intervals include zero. H2−CUM is $+0.422$ pp on MNIST and -1.196 pp on Fashion-MNIST; both corresponding intervals extend above $+0.5$ pp and outside $[-0.5, +0.5]$ pp. The mean pattern favours memory over A, while the magnitude and direction of an additional H2 effect remain open under the primary analysis.

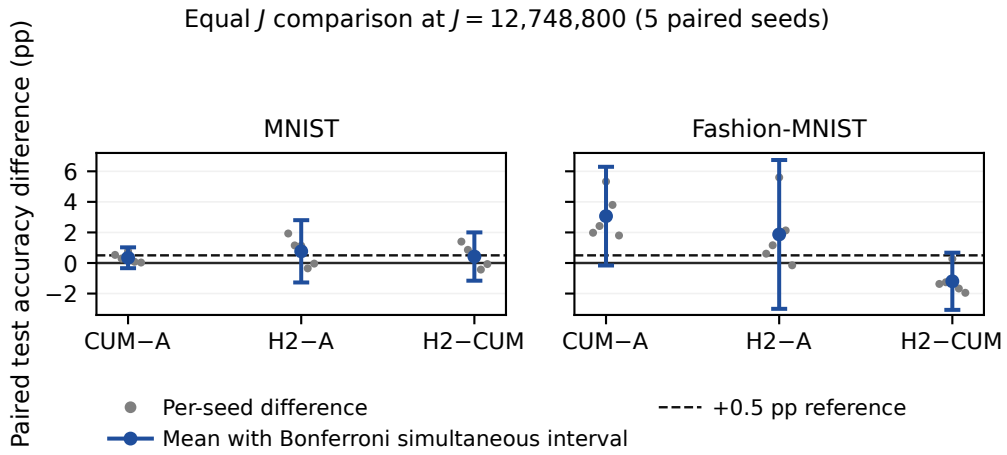


Figure 2: The four memory-versus-A simultaneous intervals all contain zero, and H2−CUM points in opposite directions on the two tasks. Grey dots show per-seed differences; blue dots and error bars show the means and Bonferroni simultaneous intervals of Table 4. The solid line marks zero and the dashed line marks the pre-set $+0.5$ pp feedback-benefit reference. All six contrasts are shown.

Table 5: Per-seed image test accuracy at the fixed endpoint (%).

Task	Seed	A (%)	CUM (%)	H2 (%)
MNIST	101	87.72	88.25	89.65
MNIST	211	87.60	87.89	88.75
MNIST	307	88.04	88.82	89.17
MNIST	401	88.60	88.68	88.25
MNIST	503	88.60	88.63	88.56
Fashion-MNIST	101	77.88	79.86	78.49
Fashion-MNIST	211	76.95	79.37	78.11
Fashion-MNIST	307	73.75	79.08	79.35
Fashion-MNIST	401	75.69	79.49	77.82
Fashion-MNIST	503	76.89	78.69	76.74

C.2 Per-seed image endpoints

Every cell has denominator 10000 and no tied maximum on the test set. With the paired difference d_s computed per task and seed, the six primary intervals are

$$\bar{d} \pm t_{4,\,1-0.05/(2\times 6)} \frac{s_d}{\sqrt{5}}.$$

These are the pre-specified paired t intervals for all six contrasts. Their interpretation reflects five seeds, seed-specific splits and earlier use of the test set in this project.

C.3 Image cost fields

Recorded time runs from cell initialisation or restoration to saving. It includes training, earlier validation and checks, and save overhead; feature preparation and the later centralised test are recorded separately. All three image methods finish at $J = 12748800$, with per-seed cost fields in each cell's `result.json`.

C.4 Per-seed event values and execution volume

Each row has denominator 288, and the single ridge fit is 92/288. The three intervals use $\bar{d} \pm t_{2,0.975} s_d/\sqrt{3}$ to describe variation across order seeds. All three arms end at $J = 168960$.

Table 6: Image cost fields, five-seed means. Sample traversals count repeat exposures; vote batches count `vote_calls`. Write counts appear in the J column.

Task	Method	Sample traversals	Vote batches	Recorded time (s)
MNIST	A	685507.2	5357.4	19.070
MNIST	CUM	709807.2	5547.4	21.582
MNIST	H2	708680.8	5538.6	21.585
Fashion-MNIST	A	659702.4	5155.8	19.222
Fashion-MNIST	CUM	692527.2	5412.4	21.053
Fashion-MNIST	H2	693013.6	5416.2	21.539

Table 7: Per-seed event test values and execution volume. The two `reused=true` cells report `elapsed_seconds`=0 because no training was rerun in this stage.

Method	Seed	Correct/total	Accuracy (%)	Samples	Vote batches
A	101	114/288	39.583333	7312	62
A	211	98/288	34.027778	7312	62
A	307	103/288	35.763889	7312	62
CUM	101	113/288	39.236111	7568	64
CUM	211	115/288	39.930556	7568	64
CUM	307	121/288	42.013889	7440	63
H2	101	114/288	39.583333	7440	63
H2	211	119/288	41.319444	7824	66
H2	307	125/288	43.402778	7440	63

C.5 Per-seed best validation values of the same-feature references

Table 8: Per-seed best validation accuracy (%) of the same-feature panel.

Task	Seed	A	CUM	H2	Zero-sum float linear	Perceptron
MNIST	2026	87.779659	88.711864	88.745761	90.372884	91.203392
MNIST	1	87.847459	88.508475	88.457626	89.779663	89.508474
MNIST	2	88.305086	88.610172	88.762712	90.508473	90.644068
Fashion-MNIST	2026	78.474575	80.745763	80.576271	82.711864	84.288138
Fashion-MNIST	1	78.220338	80.932206	81.152540	82.474577	83.559322
Fashion-MNIST	2	79.372883	81.813562	82.322037	83.203393	84.932202

All entries are best validation percentages from the original 20-epoch, three-seed protocol. The five-seed fixed-endpoint test comparison appears separately in Appendix C.2.

C.6 Capacity audit of the saved endpoints

Every state has exactly zero row sums. The per-batch unit-step bound and the recorded batch counts certify that $B_w = 10^9$ is inactive in these original runs. The separate smaller-bound experiments are reported in Appendix E.3.

C.7 Existing validation budget curve

All 90 points were recomputed from saved predictions and labels and matched the recorded values. The CUM$-$A mean is $+1.495$, $+1.451$ and $+0.451$ pp on MNIST and $+3.580$, $+2.892$ and $+2.515$ pp on Fashion-MNIST, while H2$-$CUM is -0.410, -0.146 and $+0.512$ pp on MNIST against $+1.315$, $+0.397$ and -1.169 pp on Fashion-MNIST. The existing curve reveals shrinking memory gaps and task-dependent reversal across the three budgets.

Table 9: Capacity audit over all saved endpoints. "At bound" counts coordinates whose absolute value equals $B_w = 10^9$, and "max batches" is the largest number of started batches recorded at any endpoint of that task.

Task	States	max $\|W\|$	B_w	max $\|W\|/B_w$	At bound	Max batches
MNIST	45	5467	10^9	5.47×10^{-6}	0	5554
Fashion-MNIST	45	4778	10^9	4.78×10^{-6}	0	5430
DVS128 Gesture	9	61	10^9	6.10×10^{-8}	0	66

Table 10: Existing validation curve, recomputed from saved predictions and labels without new model evaluations. Five-seed mean ± sample SD (%), at the three fixed budget points.

Task	Method	$J = 3{,}187{,}200$	$J = 6{,}374{,}400$	$J = 12{,}748{,}800$
MNIST	A	86.329 ± 0.986	86.624 ± 0.915	87.166 ± 0.376
MNIST	CUM	87.824 ± 0.439	88.075 ± 0.933	87.617 ± 0.332
MNIST	H2	87.414 ± 0.959	87.929 ± 0.556	88.129 ± 0.432
Fashion-MNIST	A	74.973 ± 3.249	76.407 ± 1.568	77.437 ± 1.554
Fashion-MNIST	CUM	78.553 ± 1.058	79.298 ± 0.813	79.953 ± 0.808
Fashion-MNIST	H2	79.868 ± 1.031	79.695 ± 0.780	78.783 ± 1.112

C.8 Random-pairing control per seed

D Traceable Reproduction Entries

Algorithm: core definition and proofs, reference implementation and pairing implementation are archived with the project records under `docs/budget_tracking_v1/` and `snn_cuda/snn_cuda/`. Images: the fixed protocol and the per-cell records of the equal-J comparison. Events: the revised protocol, the per-cell records of the v2 run and the stopping record of the original budget. Mechanism: the shared-vote summary. Neighbours: the per-cell records of the same-feature panel. The random-pairing control adds a frozen protocol, an approval record, a ledger, per-cell results, a summary and an independent artifact audit. The audit verified ten training cells, thirty validation records, ten test records and ten reference endpoints against saved prediction tensors and hashes, and reported no problems.

E Route-B Confirmation Experiments

The follow-up experiments added after the first draft vary selection temperature, task, weight bound, seed and write budget, and include a less restricted H1 reference. All arms use complete-pair commitment and the same budget accounting, with selectors and memory specified below. The final ledgers record 144 completed training cells and 2882 seconds on one RTX 4050. E5 also has at least eight earlier attempts, including a recorded 31.179-second first window; its second early window is absent from the retained summary. The records therefore cover at least 152 attempts. Appendix E.6 traces E5 endpoint revisions and retained adjustment records.

E.1 Temperature selection with cumulative memory

The implemented TAU rule accumulates U as cumulative demand and draws positive endpoints by Gumbel-top-k on logits $g/(\tau \max|g| + 10^{-12})$, then negative endpoints from the remaining coordinates using negated logits. Temperature varies stochastic selection within this memory rule. For a fixed signal with distinct rankings, small temperature approaches the ranked endpoint sets; ties, pairing order, blocked pairs and budget prefixes still shape the executed trajectory. The study uses two image tasks, seeds 101/211/307, $J = 12{,}748{,}800$ and $B_w = 10^9$; all 30 cells reach the budget. Within TAU, mean MNIST accuracy changes from 88.03% at $\tau = 0.01$ to 86.53% at $\tau = 10$ and 38.31% at $\tau = 100$; Fashion-MNIST changes from 79.21% to 77.68% and 44.73%. The broad high-accuracy range at lower temperatures and the drop near uniform selection reveal the importance of directional priority within cumulative memory.

Table 11: Per-seed random-pairing control results at the fixed endpoint $J = 12{,}748{,}800$.

Task	Seed	A (%)	RAND (%)	Difference (pp)
MNIST	101	87.72	10.91	76.81
MNIST	211	87.60	9.39	78.21
MNIST	307	88.04	15.24	72.80
MNIST	401	88.60	9.35	79.25
MNIST	503	88.60	12.90	75.70
Fashion-MNIST	101	77.88	17.04	60.84
Fashion-MNIST	211	76.95	7.32	69.63
Fashion-MNIST	307	73.75	2.18	71.57
Fashion-MNIST	401	75.69	16.10	59.59
Fashion-MNIST	503	76.89	5.42	71.47

Table 12: Cumulative-memory TAU: test accuracy (%), three-seed mean $\pm$ sample SD. All five columns are temperatures; no A endpoint is included.

Task	$\tau = 0.01$	$\tau = 0.1$	$\tau = 1$	$\tau = 10$	$\tau = 100$
MNIST	88.03 ± 0.63	88.34 ± 1.20	88.03 ± 0.46	86.53 ± 0.31	38.31 ± 5.05
Fashion-MNIST	79.21 ± 0.57	78.70 ± 0.40	77.96 ± 1.79	77.68 ± 0.96	44.73 ± 3.93

E.2 A HARDER TASK REVERSES THE MEMORY RANKING

CIFAR-10 is converted with the same encoder as the image tasks: RGB, Rec.601 grey scale, 256 levels and a first-spike latency per pixel, giving $d = 1024$. The 50,000 training images form the pool; each seed holds out 5,000 for validation and trains on the rest; the test set is 10,000. Seeds 101/211/307; $J \in \{3.19, 6.37, 12.75\}$ million with the largest point as the test endpoint; 12 cells, all reaching the budget. A exceeds RAND by about 6.6 pp in mean accuracy and exceeds CUM and

Table 13: CIFAR-10 first-spike task, test accuracy (%), three-seed mean $\pm$ sample SD.

Method	A	CUM	H2	RAND
Test accuracy (%)	17.38 ± 0.97	16.15 ± 1.13	16.53 ± 1.76	10.80 ± 1.57

H2 by about 1.2 and 0.9 pp. The CIFAR-10 result reverses the mean memory ranking seen on the two simpler image tasks, illustrating the task dependence of historical priorities.

E.3 WHEN THE WEIGHT BOUND ACTUALLY BINDS

Here $J = 320{,}000$ and $B_w \in \{16, 64, 256\}$; three arms, two tasks and three seeds give 54 completed cells. Table 14 reports all 18 task–bound–method configurations. Accuracy and drop fractions summarize the three seeds; max $|W|$ is the maximum across them. All 54 cells reach the budget with zero row sums. The bound binds at 16 and in some configurations at 64; it stays inactive at 256. H2$-$CUM means at bounds 16/64/256 are $-0.047/-0.963/+0.760$ pp on MNIST and $-1.133/+0.467/+0.043$ pp on Fashion-MNIST. Subtraction's relative accuracy varies non-monotonically as the bound tightens. At MNIST bound 64, all three differences are negative ($-0.99, -1.00, -0.90$ pp).

E.4 FIVE FRESH SEEDS

Seeds 607/709/811/907/1009 were not used in any earlier protocol. Settings match the equal-J comparison; 30 cells, all reaching $J = 12{,}748{,}800$; $B_w = 10^9$. The descriptive CUM$-$A intervals exclude zero on both tasks, adding a second seed panel to the original multiplicity-adjusted analysis. The new H2$-$CUM intervals include zero, as do those from the original image comparison and the event supplement. Across these settings, CUM's mean advantage over A is more stable than the extra effect of H2.

Table 14: Weight-bound grid, all configurations. Drop is the fraction of proposed pairs removed at the bound. $B_w = 256$ produces no dropping.

Task	Arm	B_w	Drop	Test (%)	max $\lvert W \rvert$
MNIST	A	16	0.451	85.56	16
MNIST	CUM	16	0.906	86.44	16
MNIST	H2	16	0.907	86.39	16
MNIST	A	64	0.003	84.42	64
MNIST	CUM	64	0.397	86.51	64
MNIST	H2	64	0.403	85.55	64
MNIST	A	256	0.000	82.67	81
MNIST	CUM	256	0.000	86.12	131
MNIST	H2	256	0.000	86.88	130
Fashion-MNIST	A	16	0.144	70.72	16
Fashion-MNIST	CUM	16	0.881	78.03	16
Fashion-MNIST	H2	16	0.880	76.89	16
Fashion-MNIST	A	64	0.000	70.98	57
Fashion-MNIST	CUM	64	0.251	77.13	64
Fashion-MNIST	H2	64	0.246	77.60	64
Fashion-MNIST	A	256	0.000	70.98	57
Fashion-MNIST	CUM	256	0.000	78.32	129
Fashion-MNIST	H2	256	0.000	78.37	129

Table 15: Fresh seeds: paired test-accuracy differences (pp) with descriptive 95% paired t intervals; the pre-registered Bonferroni family is reported in Table 4.

Task	Contrast	Mean	Sample SD	Descriptive 95% interval
MNIST	CUM−A	$+0.532$	0.380	$[+0.061, +1.003]$
MNIST	H2−A	$+0.442$	0.890	$[-0.663, +1.547]$
MNIST	H2−CUM	-0.090	0.741	$[-1.010, +0.830]$
Fashion-MNIST	CUM−A	$+2.342$	0.989	$[+1.114, +3.570]$
Fashion-MNIST	H2−A	$+2.224$	2.009	$[-0.270, +4.718]$
Fashion-MNIST	H2−CUM	-0.118	1.700	$[-2.229, +1.993]$

E.5 Less restricted H1: an unpaired reference

The historical H1 arm relaxes unit-amplitude and support limits and supplies a capability reference under different seeds and protocol. Its saved artifacts supply the comparison. H1 retains integer weights and zero row sums while relaxing unit-amplitude and support restrictions. Its accuracy is about 2.9 and 6.6 pp higher under a separate protocol. The comparison places the proposed unit-pair rules alongside a nearby, higher-accuracy alternative.

Table 16: Historical H1 reference and fixed-budget A (mean $\pm$ sample SD), shown with their respective seeds, epoch protocols and execution restrictions.

Task	Method	Seeds	Test accuracy (%)
MNIST	H1 (relaxed)	5	91.04 ± 0.25
MNIST	A (fixed budget)	5	88.11 ± 0.47
Fashion-MNIST	H1 (relaxed)	5	82.82 ± 0.16
Fashion-MNIST	A (fixed budget)	5	76.23 ± 1.59

E.6 Reachability ceiling of the budget axis

E5 initially requested $J = 25{,}497{,}600$. With 53100 samples, batch size 128, 20 epochs and at most 1280 pairs per batch, the theoretical ceiling is $20\lceil 53100/128 \rceil\, 2560 = 21{,}248{,}000$. The first A cell stopped at $19{,}675{,}904$. A second attempt targeted $19{,}123{,}200$: six A cells completed and one CUM cell stopped at $18{,}964{,}736$. The final endpoint was set to $16{,}000{,}000$, where all 18 cells completed.

The resulting four-point curve follows a reachability-informed protocol. Memory-minus-A means stay positive at all four budgets; from 12.75 to 16.00 million, the gaps decrease by $0.25/0.80$ pp for CUM/H2 on MNIST and $0.86/0.58$ pp on Fashion-MNIST.

The resource record comprises the final-run counts and windows above together with the retained early-attempt summaries. This manuscript revision reuses those records.

F SAME-FEATURE BEST-VALIDATION PANEL

Table 17: Same-feature nearest neighbours: three-seed mean of the best validation accuracy within 20 epochs (%). The panel reports validation outcomes under each method's own historical modification count.

Method	MNIST	Fashion-MNIST
A	87.9774	78.6893
CUM	88.6102	81.1638
H2	88.6554	81.3503
Zero-sum floating-point linear	90.2203	82.7966
Multiclass perceptron	90.4520	84.2599

The panel uses the same features, the same split within a seed, the unique-maximum decision rule and a single layer with 7840 parameters for all five methods, over seeds 2026, 1 and 2. The metric is the best validation accuracy within 20 epochs, with 20 validation evaluations and zero test evaluations per cell.

G REAL-EVENT SUPPLEMENT: DVS128 GESTURE

DVS128 Gesture (Amir et al., 2017) uses the official subject-disjoint split: 1176 labelled segments from 23 training subjects and 288 segments from 6 test subjects. For each segment we use events within one second of the start and up to the annotated end; the 128×128 space is pooled to 32×32 with both polarities, and the integer latency of the first event in each channel forms a 2048-dimensional input. Each channel contributes its first event from the real recording.

The event endpoint was revised once after a training-reachability check. At the original $J = 337920$, A with seed 101 reached the endpoint while CUM with seed 101 reached $J = 302080$ after 20 epochs, stopping the queue before event accuracy evaluation. The revised endpoint $J = 168960$ reused the two saved states exactly and trained the remaining seven cells. Seeds 101, 211 and 307 vary training order; a single ridge fit uses fixed $\lambda = 1$, and ten test evaluations follow model freezing. The final event panel is a descriptive supplement under this disclosed revision. Table 18 reports endpoint accuracies and Figure 3 shows per-seed values. Both memory rules have a higher mean than A, and all three H2$-$CUM seed differences are positive, averaging $+1.042$ pp. The descriptive intervals cross zero. This real-event panel extends the feasibility and memory comparison to first-event input.

The ridge reference uses the same input representation, centres features and weight rows, and fits once at $\lambda = 1$. It obtains 92/288 correct predictions as a real-valued row-zero-sum comparison. The paired unit-update rules reach higher mean accuracy on this first-event representation.

H N-MNIST EVENT DATASET

N-MNIST (Orchard et al., 2015) records handwritten digits with a 34×34 ATIS sensor. We use the official split of 60000 training and 10000 test records and the encoding shared with the other event experiments: a one-second window from the first event, both polarities on separate planes, no pooling, and a per-coordinate earliest-event latency quantised to 0 to 1023 with 1024 for a coordinate that stays silent. Each record is therefore a 2312-dimensional integer vector and no floating-point front end is involved. The reachability probe of seed 101 spends 19.15, 16.81 and 16.88 times the per-epoch capacity of $1{,}200{,}640$ for A, CUM and H2, so the endpoint is fixed at twelve times

Table 18: Event test accuracy at the fixed endpoint $J = 168{,}960$ (%). A/CUM/H2 are three-seed means $\pm$ sample SD; the ridge entry is a single real-valued row-zero-sum fit without an equal-J constraint and without a cross-seed SD.

Method	Test accuracy (%)	Fit or run identity
A	36.458 ± 2.842	three order seeds; equal J
CUM	40.394 ± 1.446	three order seeds; equal J
H2	41.435 ± 1.912	three order seeds; equal J
Real-valued row-zero-sum ridge	31.944	single $\lambda = 1$ fit; not equal J

Table 19: Event paired differences with descriptive 95% paired t intervals (pp), calculated across three order seeds without multiplicity adjustment.

Contrast	Mean (pp)	Descriptive 95% interval (pp)
CUM$-$A	$+3.935$	$[-5.288, +13.158]$
H2$-$A	$+4.977$	$[-5.739, +15.692]$
H2$-$CUM	$+1.042$	$[-0.452, +2.536]$

capacity, $J = 14{,}407{,}680$. Seeds 101, 211, 307, 401 and 503 give 15 cells, all reaching the endpoint before any test row was read, and each cell was tested once. The saved weights are integer with exactly zero row sums, and the accuracy values in Table 3 are the five-seed means of these cells.

I Calibrated Extensions and Execution Diagnostics

The calibration analyses examine how demand rescaling changes class margins, temporal accumulation and tracking, using validation data or fixed external vote streams. The primary A/CUM/H2 rules and fixed-J test comparison remain as defined above. This revision reuses saved results from those analyses.

I.1 Definition and signal geometry

For a nonzero zero-sum row U, define $D = \max(k\|U\|_\infty, \|U\|_1/2)$ and $z^* = kU/D$; set $z^* = 0$ at $U = 0$. Then $\mathbf{1}^\top z^* = 0$, $\|z^*\|_\infty \le 1$ and $\|z^*\|_1 \le 2k$. With $Q = 2^{16}$, balanced largest-remainder rounding produces integer q with zero sum, $\|q\|_\infty \le Q$, $\|q\|_1 \le 2kQ$ and coordinate error $|q_j/Q - z_j^*| < 1/Q$. The positive and negative sides share the same rounded total; remainder ties use ascending coordinate index. S-CUM accumulates q; S-H2 additionally subtracts $Q\Delta W$ for actual committed updates. Both retain the original active-row gate and paired unit executor. Consequently S-H2 tracks the transformed demand, not the original U.

The continuous map preserves each nonzero row's direction. Class-specific factors $\alpha_c = k/D_c$ nevertheless change the proposed margin increment from $(U_y - U_c)^\top x$ to $(\alpha_y U_y - \alpha_c U_c)^\top x$, which can reverse its sign. On 12 saved Fashion-MNIST states (1536 sample occurrences and 13824 class-pair comparisons), 461 of 7108 positive increments turn negative and 662 of 6660 negative increments turn positive; continuous-to-fixed-point rounding causes zero further sign changes. The diagnostic isolates sign reversals in proposed class margins.

Temporal accumulation also changes: for $u = (1, 1, -1, -1)$ and $k = 1$, the demands $100u, -u, -u$ sum to $98u$ while their continuous calibrations sum to $-u/2$. On an external Fashion-MNIST sequence generated by A from zero over 32 batches, the cumulative raw-versus-calibrated cosine is 0.5401 after subtracting the vector common to all class rows. The class-common component cancels in score differences. This fixed sequence isolates a mechanism that the closed-loop experiment examines separately.

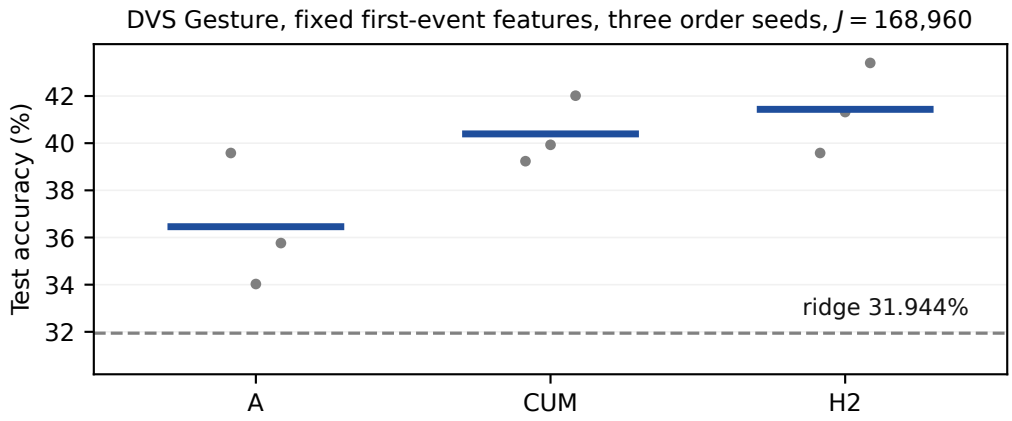


Figure 3: Both memory rules lie above A on the real event endpoint, while all three intervals cross zero. The panels show per-seed test accuracy for the three integer rules with a mean line for each method; the dashed line is the single ridge result of 31.944%. No interval error bars are drawn here, and the statistical intervals are given in Table 19.

I.2 Tracking versus classification

Let $S = \sum q/Q$ include demands incorporated so far and $E = S - W$. For a fixed comparison class c,

$$(W_y - W_c)^\top x = (S_y - S_c)^\top x - (E_y - E_c)^\top x.$$

This identity exactly decomposes the state along each training trajectory. The resulting target S depends on that trajectory. On four saved Fashion-MNIST endpoint states, S has 0.136–0.373 pp higher validation accuracy than W, while the worst configuration's S remains 2.254 pp below its H2 reference. These fixed-state results quantify the tracking target alongside the closed-loop outcomes.

For constant calibrated demand $z = (1, 1, -1, -1)/2$, $k = 1$, zero initialization, stable index ties and unblocked execution, S-CUM repeatedly selects endpoints 0 and 3 and has tracking error $\|nz - W_n\|_2 = n$. S-H2 alternates endpoint sets $(0, 3)$ and $(1, 2)$, with error 1 at odd steps and 0 at even steps. The tie example exposes their different tracking dynamics; Table 20 gives the fixed-stream measurement. The closed-loop confirmation compares H2, S-CUM and S-H2 at seeds 17/29/43 and

Table 20: Calibrated tracking on fixed external 32-batch vote streams, measured in weight units. Each rule commits 40832 pairs per task ($J = 81664$); the ratio compares terminal errors.

Task	S-CUM error	S-H2 error	Ratio
MNIST	1258.4117	28.1693	44.67
Fashion-MNIST	1183.2089	25.5644	46.28

$J = 3187200$ on two image tasks: 18 runs and 54 validation checkpoints. Seeds vary both split and order. Table 21 reports every task and arm. On Fashion-MNIST, S-H2 minus H2 is negative on all three seeds (mean -2.2429 pp), as is S-H2 minus S-CUM (mean -2.5819 pp). Both predefined improvement gates fail, stopping further resources for this extension. The confirmation remains separate from the positive development runs.

I.3 Demand scale and fixed quota

A fixed shared-scale diagnostic uses the first eight batches to select D_{ref} and the remaining 24 with $D_t = \max(D_{\text{ref}}, \max_c D_{tc})$. Every suffix batch stays within that reference. The continuous demand therefore preserves cumulative direction by construction, and its rounded class-centred cosine is 0.9999999965. Mean demand L_1 is 11.5103% of row capacity, while the fixed-k executor spends $J = 61440$, 8.69 times the observed cumulative L_1 path length.

Table 21: Separate closed-loop calibration confirmation: three-seed mean validation accuracy (%) at fixed $J = 3187200$; Table 20 reports the external-stream tracking trajectories.

Task	H2	S-CUM	S-H2
MNIST	87.3333	86.7966	86.1073
Fashion-MNIST	78.6328	78.9718	76.3898

For one legal pair with residual-coordinate gap g, the exact change of integer residual squared norm is $2Q(Q - g)$. Under the fixed shared-scale stream, 49.87% of pairs leave this norm unchanged or higher, and 88.97% of unit actions reverse the preceding action at the same coordinate. In 240 matched row states, switching from residual-ranked to current-demand-ranked coordinates raises mean demand projection per pair from 0.0074 to 0.1144 while changing mean tracking gain per pair from $+5.696 \times 10^9$ to -7.160×10^9. The result exposes a trade-off between local demand alignment and target tracking.

A separately trained adaptive-quota rule on the *row-normalized* S-H2 base retains pairs with gap $g > Q$. Its six-run, three-seed confirmation misses the Fashion-MNIST improvement gate (mean adaptive-minus-fixed validation difference -2.1525 pp at equal J; MNIST $+1.1017$ pp). The result shows how a locally favourable tracking criterion can lose classification accuracy after closed-loop training.

I.4 A SUFFICIENT CONDITION FOR UNCHANGED ENDPOINT SETS

For shared external demands, zero initialization and a complete active-row step, let $C_t = \sum_{s \le t} U_s$ and $W = W_{t-1}^{\mathrm{H2}}$. H2 selects from $C_t - W$. Set $\epsilon = \|W\|_\infty$ and sort C_t in descending order. If

$$\min\{C_{(k)} - C_{(k+1)},\ C_{(d-k)} - C_{(d-k+1)}\} > 2\epsilon,$$

then subtracting W preserves both endpoint sets: each selected-versus-unselected coordinate gap decreases by at most 2ϵ. Complete unblocked commits therefore have identical net actions. Pairing order can still differ, and blocked or prefix-truncated commits follow the executor's order. At tied selection boundaries, direct inspection resolves whether subtraction changes the action. This sufficient condition characterises one precise regime of action agreement.

I.5 EVIDENCE PROVENANCE AND LIMITATIONS

The source records are the saved calibration-geometry v2 analysis, frozen-sequence replay, demand/execution analysis, three-seed confirmation, fixed-shared-scale and execution-quota analyses, corrected same-state replay, and adaptive-quota stage-one summary. Their hashes accompany the revision supplement. The identities, counterexamples and observations above provide the arguments used for the calibration results.