Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

When Blocks Outlive Tasks: Three Safeguards for Task-ID-Free Routing in Growing Continual Learners

Nishant Kumar

23dp0004@iitism.ac.in

Indian Institute of Technology (ISM) Dhanbad

Saurabh Srivastava

Indian Institute of Technology (ISM) Dhanbad

Research Article

Keywords: continual learning, dynamic architectures, task-agnostic inference, routing, progressive networks, model selection

Posted Date: November 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11559005/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

When Blocks Outlive Tasks: Three Safeguards for Task-ID-Free Routing in Growing Continual Learners

Nishant Kumar[1*] and Saurabh Srivastava[1]

[1]Department of Computer Science and Engineering, Indian Institute of Technology (ISM) Dhanbad, Dhanbad, 826004, Jharkhand, India.

*Corresponding author(s). E-mail(s): 23dp0004@iitism.ac.in;
Contributing authors: ssrivastava@iitism.ac.in;

Abstract

Growing continual learners add blocks for new data and freeze old ones. Without task labels, task-identity predictors pick the answering block but assume one block per task. Learners that grow when a block saturates break that assumption: a block learns several data periods and keeps changing while the router relies on it. For task-identity-free routing over such blocks' complete predictions we state three safeguards for replay-based growers, each tested by removal: the active block replays its earlier periods until it freezes; the router's records are written at freeze from final weights; and later training never overwrites the router's state for a frozen block, kept per block or as exact running statistics. In a pre-declared, Holm-corrected confirmation on ten independent seeds, 19 of 20 tests (four hypotheses on five benchmarks) held, except safeguard 2 on Permuted-MNIST. On CLEAR-10's natural drift one block covers nearly every year, so block choice matters little. A freeze-time competence record, compared among blocks claiming the same data, picks a revised block's competent version, but its centring lets version pairs capture other blocks' inputs, so banks keeping old versions route better without it. A bank/router loss split finds selection loss large but router- and stream-dependent: on Split CIFAR-100 the blocks hold 0.80; a per-block router reaches 0.51, and an analytic shared router using every training row 0.62, above a nearest-class-mean baseline (0.58) but not RanPAC (0.75), both non-growing, on the same features. Under per-period testing, RanPAC beats every routed bank on every frozen-feature stream.

Keywords: continual learning, dynamic architectures, task-agnostic inference, routing, progressive networks, model selection

1 Introduction

A continual learner that grows its architecture protects old knowledge by adding parameters for new data and freezing what it already has (Rusu et al. 2016; De Lange et al. 2022; Wang et al. 2024a). After several growth events it holds several blocks, and in many growth families each block, together with what it builds on, is a complete predictor: a column of a progressive network with its own head (Rusu et al. 2016), or the network as it stood after an expansion (Wu et al. 2024). Growth stores knowledge, but it leaves open which of the stored predictors should answer a given input.

Without a task label, the usual answer is task-identity prediction: each block scores how familiar an input is, and the most familiar block answers (Aljundi et al. 2017; Kim et al. 2022b,a; Lin et al. 2024).

This works because, in these methods, one block is trained on one task and then frozen, so its statistics describe that task, are written once and never change. Asking "which block?" is then the same as asking "which task?".

Learners that decide for themselves when to grow do not keep that correspondence (Lee et al. 2020; Zhu et al. 2024; Wang et al. 2025a; Yan 2026). A natural rule is to keep training the active block while it still absorbs new data and to freeze it when it saturates. A block then learns several periods, several blocks may cover parts of one period, and the active block changes after the router has already learned to use it. Three things go wrong. The active block forgets its own earlier periods while it learns the next, so knowledge can be lost before it is ever frozen. Statistics written while the block trained describe a block that no longer exists. And a router trained on a stream with no old examples learns to prefer the newest block.

This paper is about routing in that regime. A router receives the input and the complete predictions of the available blocks, never a task label, a period index or a block's birth period, and selects or mixes those predictions. Every block's internal computation is left as its recipe specifies.

Growth pays only if the router retrieves what the blocks store. We therefore split every router's accuracy into what the bank holds (the fraction of rows some block gets right) and what the router loses by choosing, a *recoverability split* (Eq. (5), Section 6). Without replay until freeze the bank itself loses knowledge that no router can bring back; without freeze-time records, or with router state that later training overwrites, the knowledge is there and the router misses it. With all three safeguards in place selection remains a large loss, and its share depends on the router and the stream: on Split CIFAR-100 the blocks hold 0.80, so the bank loses 0.20, a per-block router a further 0.29 (0.51) and an analytic shared router 0.19 (0.62, Section 10.2); on CLINC150 the shared router loses 0.02 to selection and the bank 0.08. On the same frozen features a nearest-class-mean classifier reaches 0.58 without growth, and RanPAC (McDonnell et al. 2023) 0.75. In a chain of transformer layers grown inside FLAN-T5 the best router (0.93) clears the nearest-class-mean classifier on the chain's frozen input (0.87) but not RanPAC (0.96). The contribution is therefore diagnostic: a controlled study of how block lifetime, record freshness and version competition affect output routing when blocks span several periods. The three rules are safeguards for the replay-based growers tested here, not conditions shown to be necessary in general: they keep routing over grown blocks from failing, but they do not make growth the better learner. The selection gap that remains, and the conditions under which growth would pay, are the open problems of the regime.

Contributions

1. **The regime and its requirements** (Section 3). We state eight requirements for task-ID-free routing over blocks that outlive tasks (Table 1) and record data periods, blocks, block versions and growth events separately, with growth traces varied as a controlled factor.
2. **Three safeguards, each tested by removal** (Section 4): replay until freeze, freeze-time records, and router state that later training cannot overwrite (held per block, or accumulated as exact running statistics). In a pre-declared confirmation on ten independent seeds, 19 of 20 hypotheses (H1–H3 for the safeguards and H4 for the competence record, on five benchmarks, Holm-corrected within each) held in 10 of 10 runs, each with a 95% bootstrap interval above 0.01; the one failure is safeguard 2 on Permuted-MNIST (+0.0005; Section 9.3). An initial pre-declared evaluation, whose ten runs per benchmark (five seeds, two orders) are not independent, showed the same pattern. The tests confirm contrasts for one construction (R19) on one architecture family: H1 removes the buffer together with the freeze-time rebuild that depends on it, and H3 changes the scorer as well as sharing it (a per-block control closes most of that gap), so the rules are safeguards for the tested growers rather than proven necessary conditions. On CLEAR-10's natural yearly drift no hypothesis held, because one block already classifies nearly every year. An analytic shared router (R21) meets the third safeguard through exact running statistics and beat the per-block router in all 100 of its runs; at the floats the per-block router keeps it stays ahead on four of five benchmarks, and with that router's record rows as well only on CLINC150 (Section 10.2).
3. **Which score carries the routing** (Sections 5.1 and 9.9). A reference construction (R19) combines own-space Mahalanobis distance (Lee et al. 2018; Kim et al. 2022a; Lin et al. 2024), the block's own confidence (Kim et al. 2022b) and input density, all written at freeze; it carries the ablations and is not claimed as the best router. Per-block calibration shows which term carries the routing on which stream.

4. **A version-fork test, and a competence signal that helps locally but harms the bank** (Sections 6 and 5.2). Two versions of one block whose records are written on the same rows tie on input-only familiarity scores by construction. A freeze-time competence record, centred over the blocks whose nearest group for the input coincides (R20r, Eq. (4)), raised the rate at which the competent version is picked on all five benchmarks, on constructed and on naturally arising versions. The same centring damages routing over the whole bank: with old versions routed beside every other block it falls below the uncentred record (R20) end to end on fresh seeds on all six benchmarks, and on four of them 96–100% of the rows it loses are inputs of periods the version pair's block never learned (Section 9.7). We regard this failure mode, not the formula, as the general finding: a competence signal that selects correctly between the versions of one block can draw inputs away from the blocks that own them once it is compared across the bank. R20r is an instance of it, not a proposed router for banks that keep versions.
5. **A descriptive comparison** (Section 9) with the published task-free rules (Table 3) on faithful progressive networks, a depth-growing chain, Split CIFAR-100, CLINC150 and CLEAR-10 on frozen features, and a chain of transformer layers grown inside FLAN-T5, under eight growth traces including births off period boundaries and revisited distributions. Which score routes best changes with the stream.
6. **Where the accuracy goes** (Section 9.4). The recoverability split locates every router's loss in the bank or in the selection, beside two learners that never grow (nearest-class-mean and RanPAC) on the same features. Under each stream's per-period test no router over a grown bank reached RanPAC on any frozen-feature stream, on two sets of seeds. Under CLEAR-10's streaming test the routed banks (0.955–0.956) are above RanPAC with its own ridge rule (0.934) and below it with one fixed ridge chosen post hoc (0.959).

Scope

When to grow is not the subject: growth events come from a fixed trace or a saturation rule, both fixed in advance; the saturation rule is a limited illustration, not a tested controller. Training uses period labels (records are rebuilt at the end of each period and keep one covariance per period); only inference is task-ID-free, so the setting is not boundary-free continual learning. Output routing evaluates every block for every input, so no computation saving is claimed for column banks; for the depth chain all endpoints come from one pass. Task-aware PNN appears only as a privileged reference.

2 Related work

Growth-based continual learning

Architecture-based methods allocate new parameters to new data (De Lange et al. 2022; Wang et al. 2024a). PNNs add one column per task with lateral connections from the frozen earlier columns (Rusu et al. 2016). DER concatenates the features of frozen extractors under one retrained classifier (Yan et al. 2021). LLaMA Pro inserts identity-initialised blocks into a pretrained language model and tunes only the new blocks (Wu et al. 2024). Several systems decouple growth from task boundaries: CN-DPM adds experts by a Dirichlet-process criterion (Lee et al. 2020), TAME when the loss shifts (Zhu et al. 2024), SEMA when it detects a shift (Wang et al. 2025a), and MaRS expands routing slots over a frozen encoder (Yan 2026). In stream mining, cPNN carries progressive columns to evolving streaming time series, adding one column at each known drift and revealing each batch's labels after its prediction (Giannini et al. 2023); DYNcPNN decides after each detected drift whether to add a column or keep training the current one (Giannini et al. 2026). Growth that ignores task boundaries is therefore not new; what these systems leave open is how a router over the grown parts should be built when a part keeps changing after it is first routed to, and when no label arrives after the prediction to say which part was right.

Task-identity prediction

Expert Gate trains one autoencoder per task and forwards an input to the expert that reconstructs it best (Aljundi et al. 2017). Kim et al. (2022b) show that class-incremental learning decomposes into within-task prediction and task-identity prediction, and that the latter is an out-of-distribution detection

problem; each task head's maximum softmax is their baseline. MORE trains each task head with replayed data as out-of-distribution examples and uses class-conditional Mahalanobis scores (Kim et al. 2022a), after Lee et al. (2018). TPL predicts task identity by a likelihood ratio that combines such scores with the replay data (Lin et al. 2024). SupSup infers which supermask to use by one gradient step on the output entropy of their superposition (Wortsman et al. 2020), and AdapterCL selects the adapter by perplexity (Madotto et al. 2021). All of these fit one descriptor per task after that task. Our scores come from this line; our rules concern when and from which weights the descriptor is written, and what happens when it describes a block that has since changed.

Selecting grown modules without a task label

MoCL weights per-task modules by cosine matching between the input and learned task vectors (Wang et al. 2024b). L2R learns a router after the modules are trained, on a small memory (Araujo et al. 2024). Any-SSR routes among low-rank subspaces with an analytic recursive-least-squares (RLS) router that accumulates second-moment statistics incrementally and solves for routing weights in closed form, eliminating recency drift without storing old examples (Tong et al. 2025). FlyPrompt's REAR extends this with a brain-inspired random-expanded projection (Yan et al. 2026). SEE lets each expert decide whether to answer (Wang et al. 2025b), and SEED fine-tunes one expert of a fixed pool per task and ensembles all of them at test time (Rypeść et al. 2024). Gradient-trained routers that are shared across modules are what our third safeguard rules out at memory zero; we include L2R and a shared scorer as tests of it. Analytic shared routers (Any-SSR, REAR) meet this safeguard by construction; we test their recursive-least-squares routing on our bank as R21 (Section 10.2).

Learners that never grow, on frozen features

A nearest-class-mean classifier on fixed features admits new classes at almost no cost (Mensink et al. 2013), and on a frozen pretrained model such prototypes are a strong class-incremental baseline by themselves (Zhou et al. 2025). Analytic learners go further. Because a ridge readout depends on the data only through its second moments, accumulating them period by period reproduces joint training exactly (Zhuang et al. 2022); RanPAC does this after a frozen random projection with a nonlinearity, which makes the classes closer to linearly separable (McDonnell et al. 2023); we run its second phase only, without its first-session adapter tuning, because our encoders stay frozen. LoRanPAC stabilises RanPAC by truncating the SVD of the lifted features (Peng et al. 2025). Our CIFAR-100 and CLINC150 blocks learn on frozen features, so we run both kinds of learner on the same features. Together they set the bar that growth with task-free routing has to beat.

Routing over a changing pool of models

UniRoute represents each model by its errors on clusters of prompts, so a new model is added by profiling it (Jitkrittum et al. 2026); GraphRouter and ICL-Router also admit new models without retraining (Feng et al. 2025; Wang et al. 2026), CERA-MoA routes among agents whose competence changes as they learn (Jiang et al. 2026), and Dynamic Weighted Majority keeps one performance weight per expert (Kolter and Maloof 2007). Our competence record differs in where it is written (from a block's own buffered rows at freeze, with no data kept afterwards) and in what it is for (separating versions of one block that familiarity cannot).

Routing stability under expansion

StaR-MoE anchors earlier routing decisions when experts are added (Guo et al. 2026), CARE-MoE stabilises routing against expert conflicts (Lee et al. 2026), and in frozen mixture-of-experts models better routes than the router's choice exist but go unselected (Yoon et al. 2026). A gap between what a model contains and what its router reaches is known in general; we measure it per example (Section 6).

Each ingredient above has prior art. A targeted search found no work that combines them (routing over blocks that learn several periods and are revised, rules tested by removal, a competence record for versions), but we make no priority claim.

Table 1 Requirements for task-ID-free routing over blocks that outlive tasks. N1–N7 are checked for every router; N8 concerns the growth controller and is out of scope here.

	Requirement	How it is checked
N1	no task or period identifier at inference	information contract; tested
N2	a block may learn across several periods	shared, every-k and saturation traces
N3	no old examples are kept after a block freezes	the block's buffer is cleared at freeze; tested
N4	router evidence about a frozen block is never overwritten	per-block frozen state vs. shared trained routers at memory 0
N5	cope with revisions: the active block keeps changing	freeze-time records; version fork
N6	new blocks are admitted without refitting old state	records of frozen blocks are never rewritten; tested
N7	cheap endpoints	depth chain: every endpoint from one pass
N8	signals the growth controller can use	out of scope (calibration reported only)

3 Problem setting

Stream, blocks and growth traces

A stream delivers periods $t = 1, \ldots, T$ with training data $\mathcal{D}_t$. The learner holds blocks $u_1, \ldots, u_K$; only the newest trains. A *growth event* adds a block and freezes the previous one. A block's *periods* are those during which it was trainable, and its *versions* are its weights at the end of each of them, identified by a digest. A *growth trace* is the list of growth events. In the *aligned* trace every period starts with a birth; the other traces break that correspondence (Table 4, Figure 1). The *saturation* trace is not fixed in advance: the chain grows when its active block saturates (Section 7).

Candidates and routers

Candidate k is a complete predictor $p_k(y \mid x)$, the output of block k in its recipe's architecture with its own head, masked to the classes it has seen, together with features $c_k(x)$ computable without the answer: the block's hidden state $h_k(x)$, its predictive entropy and its top log-probability. A router maps the input x and the candidate features to weights,

$$p(y \mid x) = \sum_{k=1}^{K} \alpha_k(x)\, p_k(y \mid x), \qquad \alpha(x) = \operatorname{softmax}_k s_k(x), \tag{1}$$

and *hard selection* answers with $\arg\max_k s_k(x)$ alone. In practice, the mixture term of Eq. (5) is negligible across all runs (at most 0.001), so hard selection is the effective regime throughout. A one-hot α reproduces candidate k exactly, and a router update never changes a candidate's output; both are tested.

Information contract

No router receives the period of an input, the current period index or a block's birth period at inference. At training time, two uses of period information are declared: (i) records are written from the rows a block saw while trainable, which uses the growth history to partition training rows by block; and (ii) period labels index the groups within each block's records so that competence counts and statistics are per-period. Both are consumed before freeze and never at inference. Period labels also appear in evaluation metrics.

Identifiability

Task-free routing is well posed only if the input determines the answer. On Split-MNIST and Split CIFAR-100 the classes of different periods are disjoint. On Permuted- and Rotated-MNIST the labels repeat but the input distributions differ. With disjoint heads, uniform mixing equals max-softmax task inference: if each class y is seen by one block $k(y)$ only, the mixture is $p(y \mid x) = p_{k(y)}(y \mid x)/K$, whose argmax is the top class of the most confident block. The two therefore coincide on the class-incremental column benches (Table 10).

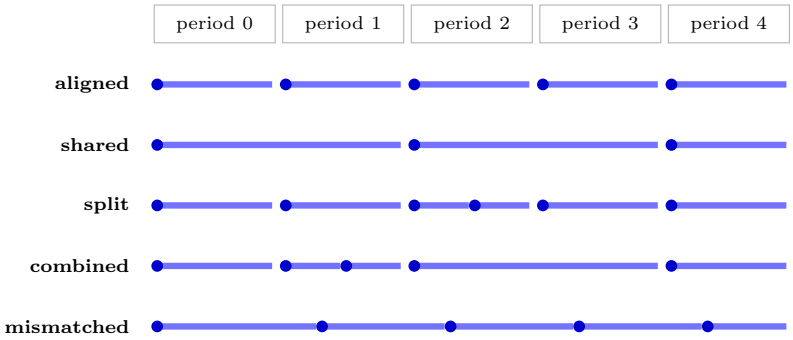


Fig. 1 Fixed growth traces on a five-period stream. Each bar is the span during which one block trains; a dot is its birth. Afterwards the block is frozen and remains a candidate. *Aligned*: one block per period. *Shared*: a block trains on two periods. *Split*: two blocks within period 2. *Combined*: a mid-period birth in period 1 and none at the start of period 3. *Mismatched*: as many blocks as aligned, each born 30% into a period, so every block but the last trains on the end of one period and the start of the next. The revisit stream (seven periods, Table 4) and the saturation trace, which is data-driven and reported per run, are not drawn. Each trace is trained as its own backbone run.

4 Three safeguards

The three rules below are what changes when a block learns several periods. Each is stated with the failure it prevents and the ablation that tests it (Table 2). We call them safeguards: removing any one degrades routing. The ablations remove each from R19, the only construction that uses all three; the other constructions that accumulate statistics from the block's data (R17, R18, R16, R4m) use some of the rules, and we expect, but test only in part, that they need them in the same way. Routers that keep no state (max-softmax, SupSup) satisfy them trivially and remain competitive on some streams; the safeguards bind only when the router uses what it builds from the block.

Safeguard 1: replay until freeze

A block that trains on period $t+1$ forgets period t like any network. In the one-block-per-task regime this never happens, because the block is frozen at the end of its task. Here it does, and the lost knowledge is gone before any router sees the block: no routing rule can recover what no candidate holds. The active block therefore keeps a buffer of its own earlier periods and replays it while it trains. At freeze the buffer is used one last time, to write the block's records, and then cleared (N3). Old data is never kept for a frozen block.

Safeguard 2: records are written at freeze time

Statistics a block accumulates while training describe the block as it was when each row passed through it. When the block keeps learning, its hidden space moves, and early statistics describe a block that no longer exists. The record of the active block is therefore rebuilt, at the end of every period, from its buffered rows passed through its current weights. The last rebuild happens at freeze and is final: a frozen block's record is never written again (N6). This is also what makes revisions safe (N5): a revised block's record describes the revised block.

Safeguard 3: router state that later training cannot overwrite

Once a block freezes, nothing that happens later may overwrite what the router holds about it (N4). The safeguard protects the router's evidence about the block, not its decisions: new blocks and new rows still change which block wins an input. A router with parameters shared across blocks and *trained by gradient descent* on the stream breaks this: with no memory it sees only the newest period's data, its gradient favours the newest block, and it drifts toward it. Two constructions meet the safeguard. State held per block and frozen with the block cannot drift, because a block's score of an input depends only on that block's own record; this is how R19 and every per-block rule meet it. An analytic shared router that accumulates exact second-moment statistics (recursive least squares, as in Any-SSR (Tong et al. 2025) and FlyPrompt (Yan et al. 2026)) meets it differently: each row enters its statistics as a separate summand that later rows add to but never alter, so its closed-form solution equals joint training on every row seen,

Table 2 Ablations. Each row removes one rule from the full construction on the same frozen bank, stream and memory. H1 removes the block buffer together with the freeze-time rebuild; P(H=1) isolates the replay contribution (Section 9.2).

Arm	What is removed	Tests
R19 on the no-replay bank	replay until freeze (bank trained without the block buffer)	safeguard 1
R19-on	freeze-time rebuild; statistics accumulated online while training	safeguard 2
R19-tied	per-period covariance; one covariance pooled over a block's periods	safeguard 2 (a block's periods differ)
R8a; L2R (R5)	per-block frozen state; a shared trained scorer, at memory 0; L2R needs a memory	safeguard 3
R19-noD / R19-noC / R18	own-space distance / confidence / input density	which term carries the score

at memory 0 (Section 10.2). The ablation removes the safeguard with a gradient-trained shared scorer (R8a); the analytic router is tested separately as R21.

5 Routers

5.1 R19: published scores under the three safeguards

For block k let $h_k(x)$ be its hidden state. Its record, written at freeze from its buffered rows, holds a mean $\mu_{k,g}$ for each (period, class) group g it was trained on, and one within-class covariance $\Sigma_{k,t}$ for each of its periods t, shrunk toward a scaled identity by $\max(\lambda_0, H/(N+H))$ for N rows in H dimensions. Groups are keyed by period as well as class because domain-incremental streams reuse labels. With $d_k(x) = \min_g (h_k(x) - \mu_{k,g})^\top \Sigma^{-1}_{k,t(g)} (h_k(x) - \mu_{k,g})$,

$$s_k(x) = -\frac{d_k(x)}{2T} + \frac{\log \max_y p_k(y \mid x)}{\tau_c} + \frac{\overline{\log \mathcal{N}}(x;\, m_k, v_k)}{\tau_x}, \tag{2}$$

where the last term is the mean per-coordinate log-density of x under a diagonal Gaussian (m_k, v_k) of the block's training inputs, written by the same records. The first term is the class-conditional Mahalanobis score of Lee et al. (2018) in each block's own space, as used for task prediction by MORE and TPL (Kim et al. 2022a; Lin et al. 2024); whitening by the block's own covariance reduces sensitivity to feature scale, and whether distances are comparable across blocks is checked empirically. The second is max-softmax task inference (Kim et al. 2022b). The third is the input-familiarity family of Expert Gate (Aljundi et al. 2017). Departures from the sources: no out-of-distribution replay and no likelihood-ratio denominator.

Fixed scales

$\tau_c = 0.05$ is the scale of the max-softmax baseline. $T = 30$ and $\tau_x = 0.25$ were fixed on the development benches (Split- and Permuted-MNIST, shared trace, seed 0, memory 0) before any Rotated-MNIST run; Rotated-MNIST, every other seed and every other benchmark are untouched by them.

5.2 Versions and the competence record (R20, R20r)

Revisions put several versions of one block in front of the router (N5). Two versions whose records are written on the same rows tie on the input term by construction, and nothing in Eq. (2) measures whether a version is *right* where its data lies. The rows that write the means are labelled, so the same pass can count, per group, how often the block's own head was right: $n_{k,g}$ rows and $r_{k,g}$ correct. R20 adds the

competence of the nearest group,

$$s_k^{\mathrm{R20}}(x) = s_k(x) + \frac{1}{\tau_r} \log \frac{r_{k,g^*} + 1}{n_{k,g^*} + 2}, \qquad g^* = \arg\min_g d_{k,g}(x), \tag{3}$$

with a Laplace estimate so that an empty group gives $\log \frac{1}{2}$. The record costs two counts per group, is written at freeze like the rest, and needs no data afterwards. $\tau_r = 0.05$ was set equal to τ_c in advance and not tuned.

Compare competence only among blocks that claim the same data (R20r)

Eq. (3) compares each block's competence on *its own* nearest group. Between blocks that hold different periods, that comparison measures how hard their tasks were rather than who owns the input, and a block whose own period was hard is pushed away from the rows it owns. R20r therefore centres a block's log-competence on the mean over the blocks whose *own* nearest group is the same group g_k^*,

$$s_k^{\mathrm{R20r}}(x) = s_k(x) + \frac{1}{\tau_r}\Big(\ell_{k,g_k^*} - \frac{1}{|B_{g_k^*}|} \sum_{j \in B_{g_k^*}} \ell_{j,g_k^*}\Big), \qquad \ell_{j,g} = \log \frac{r_{j,g} + 1}{n_{j,g} + 2}, \tag{4}$$

where $B_g = \{j : g_j^* = g\}$ is the set of blocks whose nearest group for x is g, with no restriction to versions of the same block: distinct blocks sharing a group are also compared (split and combined traces), and revisited distributions receive new period identifiers. A block alone in its group contributes zero, so R20r equals R19 whenever blocks hold disjoint periods, and it equals R20's difference between two versions that share their nearest group (both tested). It adds no scale. Writing the competence counts from 500 held-out rows per block instead of the record rows changed the R20r−R19 pick-rate difference by less than 0.01 on all four fork benchmarks (10/10 runs in the same direction). R20r was designed after a development run on the growing chain showed the failure above, and revised once on the development fork: a first version centred over every block *holding* a record of g_k^*, so an old version whose only group was the first period won later-period inputs on its first-period competence (a regression test now covers this). Both steps are declared. The centring has a cost that the fork cannot show: with old versions routed beside every other block, the competent version's bonus draws inputs the pair does not own (Section 9.7).

5.3 Comparators

The comparators (Table 3) are the recipes' own newest-block rule, uniform mixing, and the published task-free rules: MoCL-style keys with several keys per block (Wang et al. 2024b), L2R trained post hoc on the memory (Araujo et al. 2024), max-softmax (Kim et al. 2022b), SupSup's one-shot entropy gradient (Wortsman et al. 2020), taken per example over the blocks' outputs rather than per batch over supermasks, a faithful Expert Gate with one undercomplete autoencoder per block (Aljundi et al. 2017), and a per-block input density. A shared answer-trained scorer (R8a) and L2R test safeguard 3. R12 is a selector from our own earlier, unpublished work: a per-block key gate combined with a shared selector, designed for the growing chain. It is run in that setting (Section 7) as well as on the column bank.

6 Measurements

Recoverability split

For a held-out example (x, y) let $H(x) = \mathbf{1}[\exists k : \arg\max_{y'} p_k(y' \mid x) = y]$ indicate that *some* candidate is right and $R(x)$ that the router is right. Then

$$P(R{=}1) = P(H{=}1) - P(H{=}1, R{=}0) + P(H{=}0, R{=}1). \tag{5}$$

Table 3 Routers compared, by four properties. None reads a task label, a period index or a birth period at inference. Arms marked † are adaptations of the cited mechanism to complete-output routing, each with a record of its departures.

Tag	Router	Supervision	State	Update
R0	newest block (recipe default)	none	none	–
R1	uniform mixture	none	none	–
R4m	several keys per block† (Wang et al. 2024b)	block's training inputs	keys per block	gradient, frozen
R5	L2R† (Araujo et al. 2024)	mixture NLL	shared router	post hoc, memory
R8a	shared answer-trained scorer	candidate NLL	shared scorer	gradient
R11	input density	block's training inputs	Gaussian per block	at freeze
R14	max-softmax (Kim et al. 2022b)	none	none	–
R15	SupSup one-shot† (Wortsman et al. 2020)	none	none	–
R16	Expert Gate (Aljundi et al. 2017)	block's training inputs	autoencoder per block	frozen
R12	UCLBR selector (ours, earlier)	mixed	keys + shared gate	gradient
R17	own-space Mahalanobis† (Lee et al. 2018; Kim et al. 2022a)	labelled buffer rows	record per block	at freeze
R18	R17 + max-softmax, after TPL† (Lin et al. 2024)	as R17	as R17	at freeze
R19	R18 + input density (Eq. 2)	as R17	as R17	at freeze
R20	R19 + competence record (Eq. 3)	as R17	+ two counts per group	at freeze
R20r	R20, compared within a nearest group (Eq. 4)	as R17	as R20	at freeze
References: each block alone; hindsight best (bounds hard selection only); PNN with the task label (privileged, never ranked with the task-free routers).				

The first term depends on the bank alone; it is what safeguard 1 protects. The second is the error the router could have avoided by choosing a candidate already present; it is what safeguards 2 and 3 protect. It is a hindsight quantity: it counts rows on which some candidate was right, not rows on which the information available before the answer identifies that candidate. We call it selection error descriptively; as for oracle gaps in routing between language models or between experts (Shihab et al. 2026; Yoon et al. 2026), how much of it a deployable router can recover is not established by its size. The third counts examples that only a mixture gets right. All three are computed after every period; we report the final values.

Calibration of the scores

Each score (and each term of Eqs. (2)–(3)) is evaluated per block as a detector, by AUROC for "this row is from one of my periods" and "my prediction on this row is right", and by how often the block that owns a row (the last block trained on its period) scores highest.

The version fork

The fork isolates competence from familiarity. One seed's bank is trained twice from the same initialisation, with and without the block's replay buffer; the two runs are bit-identical through the first period (tested by weight digest). Every version receives the same seeded record rows (seed $7919s + v$ for version index $v \in \{0, 1, 2\}$, where s is the run seed). After a block's later period we have, for the same block, a version that replayed its earlier period and a version that forgot it. They differ only in whether the block replayed its buffered earlier-period rows while it learned the later period, so their competence differs by design while their records are written on the same rows. A third version is the replay run's block as it stood after its first period (*stale*). For each pair of versions and each router, the two versions are offered as the only candidates on the rows of the block's earlier period, in both orders, and we report how often the router picks the version that is right on rows where exactly one of them is (a tie scores $\frac{1}{2}$), and the routed accuracy. Input-only scores tie by construction, so the fork measures what remains when familiarity carries no information.

Table 4 Growth traces. Births are (period, fraction of the period). Each trace is trained as its own backbone run. Equal block counts control parameters, not exposure. On the ten-period streams (CIFAR-100, CLINC150) the shared trace has a birth every second period (five blocks). In the revisit stream periods 5 and 6 draw fresh training rows of the distributions of periods 0 and 1 (disjoint from the first presentation) and reuse their test sets; its accuracies are means over the five distinct distributions.

Trace	Births (5-period stream)	Blocks	What it tests
aligned	start of each period	5	the one-block-per-task reference
shared	starts of 0, 2, 4	3	a block over two periods (N2); the main trace
every-k	every $k \in \{2, 3, 4\}$ periods	–	how many periods a block can hold
split	aligned + mid-period 2	6	two blocks within one period
combined	aligned − start of 3, + mid-period 1	5	placement at equal block count
mismatched	start of 0, then 30% into 1, 2, 3, 4	5	births off period boundaries, at aligned's block count
revisit	aligned on 7 periods; 5 and 6 return to 0 and 1	7	one distribution learned by two blocks, at different times
saturation	chain's own rule (above)	data	growth decided by the learner

7 Experimental setup

Backbones

Progressive networks, faithful. Columns and lateral connections use Avalanche's `PNNLayer` and `PNNColumn` (Carta et al. 2023) unchanged, with one head per column over the global label space, masked to the classes it has seen. Avalanche's automatic column-per-task growth is switched off, so columns are added by the growth trace alone (tested). *The growing chain.* Block 0 is a ReLU network; block $k \geq 1$ is a residual block $h + W_2\,\mathrm{relu}(W_1 h)$ with $W_2 = 0$ at birth, so that a new endpoint equals the previous one, as in LLaMA Pro's identity initialisation (Wu et al. 2024). Each new head is a copy of the previous head. Every endpoint comes from one forward pass (N7). The chain grows by saturation: before period p a block is born if, at the end of period $p-1$, the active block lost more than 0.03 accuracy on its buffered rows of an earlier period, fit period $p-1$ below 0.95 on its buffered rows, or had learned three periods. The thresholds were fixed before any run, and the decisions are reported per run.

Benchmarks

Split-, Permuted- and Rotated-MNIST (LeCun et al. 1998) (five periods) are the screen; Split- and Permuted-MNIST are the development benches and Rotated-MNIST is untouched. Split CIFAR-100 (Krizhevsky 2009) (ten periods of ten classes, seeded class orders) is the class-incremental case at realistic scale. Following the frozen-feature protocol, each image is mapped once by an ImageNet-pretrained ResNet-50 (He et al. 2016) (torchvision weights, input resized to 224) to its 2048-dimensional feature; the encoder never sees CIFAR-100 labels and never trains. The growing blocks learn on these features.

CLINC150 (Larson et al. 2019) (the 150 in-scope intents, out-of-scope rows dropped) is the text case: one period per published domain, ten periods of 15 intents. Two settings are used. On progressive-network columns, each utterance is mapped once by the frozen encoder of FLAN-T5-base (Raffel et al. 2020; Chung et al. 2024), mean-pooled to a 768-dimensional feature, as for CIFAR-100. In the *T5 chain* each block is one T5 encoder layer appended to the frozen FLAN-T5-base encoder, born as a copy of the layer below with its attention and feed-forward output projections zeroed (an exact identity, as in LLaMA Pro); block k's endpoint is the mean-pooled, final-normed stream after it, read by a linear head copied from block $k-1$ at birth, and the input-density term reads the frozen encoder's pooled output. Because the frozen stream is large in norm, SGD barely moved the appended layer, so the T5 chain trains with Adam (learning rate 10^{-3}, three epochs per period); everything else is as on the other banks.

CLEAR-10 (Lin et al. 2021) is the natural-drift case, added in the independent-seed confirmation (Section 9.3): Internet photographs of 11 classes (ten objects and a background class) in yearly buckets from 2004 to 2014: ten periods of the same 11 classes, a domain-incremental stream whose drift is real rather than constructed, with 3,300 training and 550 test images per period, mapped once by the same frozen ResNet-50. Unless stated otherwise we use each bucket's own test split (CLEAR's i.i.d. protocol); stage G adds its streaming protocol, which tests the bank after each bucket on the next bucket's test split.

Table 5 Run order, platforms and outcomes. Each stage had a stop rule fixed before it ran. Every comparison in this paper uses runs from one platform; the platform of each table is stated in its caption. CI: GitHub Actions CPU runners; local: one laptop CPU; Colab: one NVIDIA T4.

Stage	Experiments	Platform, seeds	Outcome
A	full screen: 3 MNIST benches × {aligned, shared, shared without block buffer} × memory {0, 200}; ablations (Table 2); calibration	CI, seeds 0–4	every removal hurt on every bench; the declared router-superiority gate (R19 or R20 above the strongest published rule on ≥ 2 benches including Rotated-MNIST) failed, so no superiority claim is made
B	version fork; every-k, split and combined traces; buffer size; the growing chain with saturation growth	local (fork), CI (rest), seeds 0–4 / 0–2	the fork separates competence from familiarity; R20r designed and revised once on it (Section 5.2)
C	Split CIFAR-100 on frozen image features; CLINC150 on frozen language-model features; the T5 chain	CI / local / Colab, seeds 0–2 (T5 chain 0–3)	the safeguards hold on all three; which score leads changes
D	initial pre-declared evaluation: H1–H4 below on fresh seeds 10–14 × two orders (runs not independent)	CI (MNIST H1–H3), local (fork, CIFAR-100, CLINC150)	19 of 20 in 10/10 runs; seed-level Holm $p = 0.125$ (Table 8)
E	after stage D, screening only: the sweep cells CI lost to a resource limit (Rotated-MNIST traces and buffer; Rotated and Permuted chains); two new traces, mismatched timing and revisit, on the three MNIST benches; RanPAC on the frozen features; the saturation rule without its cap; a fourth T5-chain seed	local, seeds 0–2; Colab (T5 seed 3)	storage intact under both traces ($P(H{=}1)$ within 0.010 of aligned); off-boundary births cost selection, and exposed a record-writing gap, fixed (Section 9.5); RanPAC above every routed bank; the uncapped rule never fired, and its single block beat the routed chain on Split and Permuted
F	pre-declared confirmation on independent seeds: H1–H4 and R21>R19 on the five benches plus CLEAR-10 (E1); naturally arising versions routed with every block (E2, H6); NCM and RanPAC on the same seeds	local, fresh seeds 20–29, one order	H1–H4: 19 of 20 hold again, R21>R19 on 5 of 5, CLEAR-10 null (Table 9); H6 holds on 5 of 5, but R20r falls below R19 and R20 end to end once old versions are routed with every block (Table 14)
G	pre-declared, after a further review: R21 at R19's allocated floats, on R19's record rows, and both (E4), then at R19's kept floats (E4K); the cost of centring with versions kept, with intrusion measured (E5); CLEAR-10 under its streaming protocol, with NCM and RanPAC (E6)	local, seeds 20–29 (E4, E6), fresh seeds 30–39 (E5), one order	R21 stays above R19 at R19's kept floats on 4 of 5 (tie on Permuted), with R19's rows as well only on CLINC150 (Table 19); R20>R20r on 6 of 6, with more intrusion on 5 of 6 (Table 15); under streaming block choice still matters little

Memory and training

Each progressive-network column has two hidden layers of 100 units and trains with SGD (learning rate 0.05, momentum 0.9, mini-batch 128) for five epochs per period; routers with parameters train for five epochs per period (learning rate 3×10^{-3}), post-hoc routers for 30. Two memories are kept apart. The block buffer (safeguard 1; 500 rows by default, swept over 100–2000) belongs to the active block, is replayed while it trains, writes its records and is cleared at freeze. The router memory is 0 (the main condition, N3/N4) or 200 rows in a seeded reservoir (Vitter 1985) whose membership is identical for every router (tested). Routers act on a frozen bank: the backbone is trained once per trace and seed, a snapshot is kept after every period, and each router is trained period by period on the snapshot it would have seen. Stored bytes are reported for buffer rows (discarded at freeze), router records (kept) and the router memory.

Metrics

After every period, every router and every candidate is evaluated on every period's held-out split. Reported are final mean accuracy, the three terms of Eq. (5), hard-selection accuracy, mass on the newest block, the calibration AUROCs, and stored bytes. Continual-learning metrics use Avalanche's metric classes (Carta et al. 2023).

8 Programme and statistics

Screening, initial evaluation and confirmation

Stages A–C are screening: consistent signs across seeds and per-period tables, with no inferential claim. Stage D followed a written freeze of the method, the contrasts and the test; because its runs are not independent (below), stage F is the confirmation and stage D an initial evaluation. The test is a one-sided exact paired sign-flip test with Holm correction (Holm 1979) over the family of each benchmark. With n

paired runs the smallest attainable one-sided p is 2^{-n} (0.031 at $n = 5$, 0.001 at $n = 10$). The test confirms direction, not magnitude: effect sizes are reported alongside the p-values; stage F adds seeded bootstrap intervals (10,000 resamples) over ten independent seeds (Table 9), and an effect is said to exceed 0.01 only when that interval does. The families are corrected within each benchmark, not across all 20 tests; a cross-benchmark correction would raise the threshold but would not change any outcome except the already-failed H2 on Permuted-MNIST.

Hypothesis family (fixed before stage D)

All on fresh seeds and a second order, shared trace, memory 0, final mean accuracy unless stated; one-sided paired sign-flip tests, Holm over the family: (H1) R19 on the bank without the block buffer < R19 (safeguard 1); (H2) R19-on < R19 (safeguard 2); (H3) R8a < R19 (safeguard 3); (H4) R20r > R19 in the version fork's decisive-row pick rate (pair replay vs. forgot, the block's first period). The family is Holm-corrected within each benchmark (five families of four) on Split-, Permuted- and Rotated-MNIST, Split CIFAR-100 and CLINC150. The first four families were fixed on 12 Oct 2026; the CLINC150 family, added on 28 Oct 2026, applies the same four hypotheses unchanged and was declared after the other stage-D runs but before any CLINC150 run of stage D. Rotated-MNIST, Split CIFAR-100 and CLINC150 are untouched by any choice of scale. Router comparisons with published rules are reported descriptively, with no test. Two guards, not in the family: R20r must not fall below R19 by more than 0.01 on the main screen or on the growing chain's saturation trace (when every block's nearest group is its own, as on these disjoint blocks and on the chain, the two are equal by construction, so both guards are met trivially and only the fork rows are informative); and in the fork, on each of the other five pair–period rows, its mean must not fall below R19's by more than 0.05. All six fork rows are reported.

Independent seeds (stage F)

The ten runs per benchmark in stage D are five seeds in two orders, so they are not independent: at the seed level every held hypothesis is 5/5 but reaches only Holm $p = 0.125$. Stage F therefore repeats the family on ten fresh seeds (20–29, order 0), ten independently trained banks per benchmark on one platform (local CPU), with hypotheses, settings and analysis written down before any run on these seeds; it is the confirmation. The only deviation, a fix to the CLEAR-10 feature extractor that had skipped 279 PNG and GIF images, was made before any feature file existed and is logged with the declaration. Stage F adds CLEAR-10 as a first test (not a replication), a seeded 95% bootstrap interval for every test, and a practical threshold $\delta = 0.01$, with the guards of stage D. Family 2 tests R21>R19 on the replay-500 screen and E2 tests one hypothesis on naturally arising versions (H6, Section 9.7), each Holm over the benchmarks. An earlier post-hoc extension of stage D with seeds 15–16 (all 16 tests held over 14 runs) is superseded by stage F.

Budget, centring and streaming (stage G)

Stage G answers three questions left open by stage F, each declared before any run on its seeds and tested as in stage F. E4 limits R21 to R19's allocated floats (H7a), to R19's record rows (H7b) and to both (H7c), each tested as > R19, on the stage-F banks, whose digests and R19 and R21 accuracies reproduced stage F exactly. E4K, declared after a later audit found that R19 fills only part of its allocation, repeats the first and third limits at the floats R19 keeps (H10a, H10b; R19 reproduced E4 exactly). E5, on fresh seeds 30–39, tests R20 > R20r end to end with versions at replay 0 (H8) and whether R20r sends more rows to a version of a block that never learned their period than R20 does (H9). Each is Holm-corrected over the six benchmarks. E6 routes CLEAR-10 under its streaming protocol with two two-sided tests, R19 against R0 and R21 against R19, Holm over the two. One-seed smoke runs on a seed outside both sets came before the declaration was complete; they are disclosed with it, and no setting changed after them.

9 Results

Unless stated otherwise, results use the shared trace and memory 0 and report final mean accuracy over all periods, averaged over runs. PNN with the task label is the privileged reference throughout.

Table 6 Safeguard 1. $P(H{=}1)$, the fraction of test rows that *some* candidate gets right, with and without the active block's replay buffer, and max-softmax (R14), a rule that keeps no state, on the same banks. Shared trace, memory 0. MNIST: CI run 38242387262, 5 seeds; CIFAR-100: CI run 38275457654, 3 seeds; CLINC150 columns: local CPU, 3 seeds; T5 chain: Colab T4, 4 seeds.

	Split	Permuted	Rotated	CIFAR-100	CLINC150	T5 chain
$P(H{=}1)$, buffer 500	0.978	0.962	0.960	0.804	0.919	0.953
$P(H{=}1)$, no buffer	0.597	0.897	0.909	0.446	0.465	0.626
R14, buffer 500	0.717	0.913	0.832	0.503	0.694	0.860
R14, no buffer	0.371	0.813	0.740	0.287	0.325	0.488

Table 7 Safeguard ablations in the screen: R19 and the reference as final accuracy, each removal as the paired difference to R19 (runs below R19). MNIST: CI run 38242387262, seeds 0–4; CIFAR-100: CI run 38275457654, seeds 0–2. R20r was added after the MNIST screen; it equals R19 by construction on these disjoint blocks, as the CIFAR-100 column shows.

Variant	Split	Permuted	Rotated	CIFAR-100
PNN + task label (reference)	0.978	0.952	0.939	0.804
R19 (all three safeguards)	0.818	0.948	0.884	0.496
− replay until freeze (safeguard 1)	−0.305 (5/5)	−0.087 (5/5)	−0.078 (5/5)	−0.193 (3/3)
− freeze-time records (safeguard 2)	−0.276 (5/5)	−0.001 (5/5)	−0.012 (5/5)	−0.137 (3/3)
− per-period covariance (safeguard 2)	−0.171 (5/5)	−0.006 (5/5)	−0.013 (5/5)	−0.040 (3/3)
− per-block state, R8a (safeguard 3)	−0.619 (5/5)	−0.621 (5/5)	−0.450 (5/5)	−0.336 (3/3)
− own-space distance	−0.232 (5/5)	0.000 (0/5)	−0.016 (5/5)	−0.001 (2/3)
− confidence	−0.001 (4/5)	+0.004 (0/5)	−0.001 (2/5)	−0.037 (3/3)
− input density (= R18)	+0.020 (0/5)	−0.053 (5/5)	−0.013 (5/5)	−0.026 (3/3)
+ competence record (R20)	−0.019 (5/5)	−0.000 (4/5)	−0.010 (5/5)	−0.049 (3/3)
+ competence within a group (R20r)	–	–	–	0.000 (0/3)

9.1 Is the knowledge still there? (safeguard 1)

Without the buffer the active block forgets its first period before it freezes, and the bank loses what no router can recover: $P(H{=}1)$ falls by between 0.05 and 0.45 (Table 6). The loss is a property of the bank, so it also hits a rule that keeps no state at all (R14). On Split-MNIST the bank's ceiling rises with the buffer, 0.943, 0.967, 0.978, 0.984 and 0.987 for 100, 250, 500, 1000 and 2000 rows (CI, run 38325750828 except 500), and R19 rises from 0.746 to 0.82 by 500 rows and then levels off. On Permuted-MNIST the ceiling saturates at 0.962 from 500 rows. On Rotated-MNIST (local CPU, seeds 0–2) the ceiling rises from 0.944 to 0.967 and R19 from 0.845 to 0.902 over the same range, without levelling off. The mixture term of Eq. (5) was at most 0.0011 in any of 2586 router runs, so every router's gap to $P(H{=}1)$ is examples it could have answered, in hindsight, by choosing a block that was already right.

9.2 Each safeguard, removed

In the screen every safeguard hurt on every benchmark when removed (Table 7, Figure 2). Freeze-time records and the per-period covariance matter most on Split-MNIST and CIFAR-100 and little on Permuted-MNIST, where input density carries the routing (Section 9.9). A stale record is worse than none: on Split-MNIST, CIFAR-100 and CLINC150, R19 with online records (0.541, 0.359, 0.498) is below R19 with the distance term removed altogether (0.586, 0.495, 0.750): stale distances do active harm. The shared trained scorer is the largest failure everywhere: at memory 0 it drifts to the newest block (all of its weight on the newest block on Split-MNIST, 0.82 on Permuted- and Rotated-MNIST). Given a router memory of 200 old rows it recovers part of the loss (gap to the reference −0.283 / −0.039 / −0.143), and L2R, trained post hoc on that memory, reaches −0.204 / 0.000 / −0.054, close to R19 at the same memory on Permuted- and Rotated-MNIST; both need stored old examples, which the regime excludes (N3). The competence record of R20, compared across blocks, costs up to 0.05; R20r removes that cost by construction.

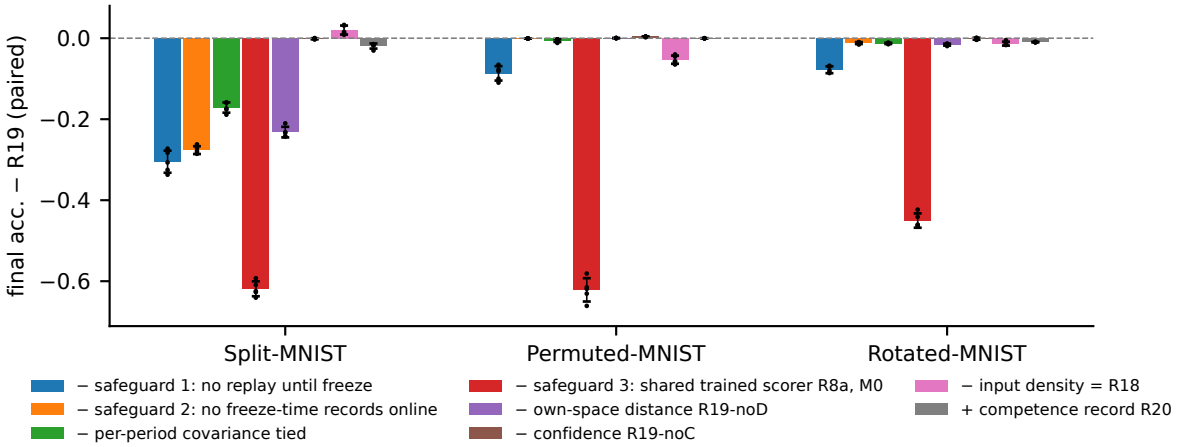


Fig. 2 Safeguard ablations on the MNIST screen (Table 7): paired difference to R19 per seed (dots), mean and s.d. (bars). CI run 38242387262.

Table 8 Initial evaluation (stage D), descriptive: mean ± s.d. of the paired difference over ten runs (fresh seeds 10–14 × two orders) and the runs in the predicted direction. The runs are not independent: run-level $p = 0.001$ (Holm ≤ 0.0039) overstates the evidence, and at the seed level every 10/10 entry is 5/5, Holm $p = 0.125$. Stage F (Table 9) is the confirmation. MNIST H1–H3: CI run 38326069660; MNIST H4, all four CIFAR-100 hypotheses, and all four CLINC150 hypotheses: local CPU. The family was fixed before these runs.

	H1 R19−no buffer	H2 R19−online rec.	H3 R19−shared scorer	H4 R20r−R19
Split-MNIST	.298±.022 (10/10)	.268±.014 (10/10)	.583±.102 (10/10)	.481±.057 (10/10)
Permuted-MNIST	.095±.027 (10/10)	.000±.001 (6/10)*	.614±.026 (10/10)	.051±.018 (10/10)
Rotated-MNIST	.105±.034 (10/10)	.028±.013 (10/10)	.401±.054 (10/10)	.169±.044 (10/10)
Split CIFAR-100	.194±.012 (10/10)	.141±.014 (10/10)	.335±.009 (10/10)	.188±.030 (10/10)
CLINC150	.393±.010 (10/10)	.312±.012 (10/10)	.611±.014 (10/10)	.192±.032 (10/10)

*Run-level Holm $p = 0.068$.

Matched control for safeguard 3

A per-block scorer (R8p) replaces the shared scorer (R8a) with an identical architecture duplicated per block and frozen with it (ten paired runs per benchmark on the stage-D seeds). R8p closes 55–80% of the shared-to-R19 gap (gap closed = (R8p − R8a)/(R19 − R8a)) (Split-MNIST 0.560, Permuted 0.723, Rotated 0.803, CIFAR-100 0.349; vs. R8a 0.202/0.325/0.481/0.164 and R19 0.814/0.949/0.883/0.499 on the same runs). R8p > R8a and R19 > R8p in 10/10 runs and 5/5 seeds on every benchmark ($p = 0.001$ runs, $p = 0.031$ seeds). The per-block scorer recovers most of the gap, supporting the interpretation that sharing compounds the problem (H3), but R8p also changes the training trajectory and the scorer architecture, so the control is not single-factor. Freeze-time records supply the remaining gap to R19.

Initial evaluation

Nineteen of the twenty hypotheses point the predicted way in 10 of 10 runs (Table 8). H2 on Permuted-MNIST does not: the mean difference is +0.0004 (s.d. 0.001), in 6 of 10 runs. This is the bench where the own-space distance carries no weight in the screen (Table 7), so records of the hidden space, stale or not, barely enter the score. All four point the predicted way on CLINC150 (10/10 runs each), the only non-image benchmark in the family; H3 shows the largest effect (+0.611), consistent with the shared scorer's near-total collapse on intents. H1 compares R19 on two banks, and on the bank without a buffer there are no buffered rows from which to rebuild records at freeze, so R19 there runs on online records; H1 therefore removes the buffer together with the freeze-time rebuild that depends on it. The router-independent drop in $P(H{=}1)$ (Table 6) is what isolates safeguard 1. The guards on the main screen and the chain are met

Table 9 Stage F: the confirmatory family and R21>R19 on ten independent seeds (20–29, one order, local CPU), with CLEAR-10 as a first test. Mean paired difference and seeded 95% bootstrap interval. Every entry on the five upper benchmarks is positive in 10 of 10 runs, one-sided $p = 0.001$, Holm-adjusted $p = 0.0039$ (H1–H4, within each benchmark) or 0.0059 (R21, over the six benchmarks), except *Permuted-MNIST H2: 6/10, $p = 0.066$, not shown. †Holds but below the practical threshold $\delta = 0.01$. CLEAR-10: no entry holds (Holm $p \geq 0.24$).

	H1 R19−no buffer	H2 R19−online	H3 R19−R8a	H4 R20r−R19 (fork)	R21−R19
Split-MNIST	.301 [.292, .310]	.278 [.273, .282]	.624 [.616, .630]	.475 [.452, .495]	.124 [.118, .132]
Permuted-MNIST	.088 [.081, .096]	.0005 [−.000, .001]*	.630 [.616, .642]	.042 [.034, .050]	.003 [.002, .003]†
Rotated-MNIST	.078 [.074, .081]	.014 [.012, .016]	.449 [.440, .458]	.139 [.128, .153]	.053 [.051, .055]
Split CIFAR-100	.178 [.171, .186]	.129 [.125, .134]	.327 [.322, .332]	.200 [.177, .222]	.130 [.126, .133]
CLINC150	.395 [.390, .400]	.306 [.298, .314]	.619 [.613, .625]	.197 [.188, .206]	.095 [.093, .098]
CLEAR-10	.001 [−.000, .002]	.000 [−.000, .000]	.000 [−.001, .001]	.001 [−.006, .009]	.000 [−.001, .001]

trivially (R20r = R19 on disjoint blocks; chain, local CPU, ten runs per MNIST bench). The fork guard holds on 24 of the 25 other rows (five per benchmark); the exception is Section 9.6.

9.3 Confirmation on independent seeds and on natural drift

On ten independently trained banks per benchmark the confirmation reproduces stage D's pattern (Table 9). The same 19 of 20 hypotheses hold, 10/10 on independent seeds (Holm $p = 0.0039$), with bootstrap intervals above 0.01 and effects within a few hundredths of stage D. The same one fails: safeguard 2 on Permuted-MNIST (+0.0005, 6/10). The guards behave as in stage D: the screen guards are met trivially, and the fork guard fails on the same Split-MNIST row (replay vs. forgot, later period, −0.315) and holds on the other 29 rows of the six benchmarks. R21 beats R19 on all five benchmarks (50 of 50 runs, Holm $p = 0.0059$); on Permuted-MNIST the difference (+0.003) is below the practical threshold.

On CLEAR-10 no hypothesis holds, and every difference is within 0.001 of zero, because every block classifies nearly every year: the newest block alone reaches 0.950, every router from max-softmax to R21 lies between 0.958 and 0.960, the bank's ceiling is 0.980, and of a revised block's two versions exactly one is right on only 3% of its first period's rows. With frozen ImageNet features and yearly drift of this size the blocks do not differ in what they know, so the safeguards, which protect the choice of block, have nothing to protect. They bind when blocks hold different knowledge, which the class- and domain-incremental benchmarks above produce and this natural drift does not.

CLEAR-10 under its streaming protocol

CLEAR's authors note that the i.i.d. protocol flatters continual learners and also test each year's learner on the next year (Lin et al. 2021). Routing year p+1's test rows over the bank after year p (stage G, seeds 20–29, mean over the nine steps) separates the blocks only slightly more. Every routed rule lies between 0.955 and 0.956; R19 beats the newest block alone (0.950) in 10 of 10 runs by +0.005, below δ (two-sided Holm $p = 0.004$), and ties R21 ($p = 0.74$). The best single block reaches 0.956 and the row oracle 0.969, a hindsight gap of 0.014 that no rule closes. On eight seeds RanPAC reaches 0.958, within 0.007 of R19 (above it on 7 of 10 seeds in all). On the other two, one or two steps collapsed (0.09 to 0.58) at the ridge its own validation rule chose, which brings its ten-seed mean to 0.934. These are failures of the ridge selection on single steps, not of the stream: in a post-hoc check on the same seeds, projection and steps, one fixed $\lambda = 10^6$ (the rule's most frequent choice) gives 0.959 with no step below 0.936, and every fixed λ from 10^4 to 10^7 keeps all 90 steps above 0.90. NCM reaches 0.858. Under either protocol, CLEAR-10 bounds the scope of the safeguards rather than testing them.

9.4 Routers across benchmarks and traces

No task-free rule reaches the reference on the class-incremental benches (Table 10). The gap is largest on Split CIFAR-100, where every rule stays about 0.3 below it and the best published rules (SupSup, input density, max-softmax) and R19 lie within 0.011 of each other. No score is best everywhere: own-space distance leads on Split-MNIST, CLINC150 and in the T5 chain; input density leads on Permuted-

Table 10 Routers, descriptively (no tests): final accuracy, shared trace, memory 0. Columns come from different platforms and are not compared with each other: MNIST, CI, 5 seeds; CIFAR-100, CI, 3 seeds; CLINC150 columns, local CPU, 3 seeds; T5 chain, Colab T4, 4 seeds (no task-labelled PNN exists for the chain; its best single block reaches 0.952). On the column benches with disjoint classes R1 equals R14 (Section 3). The highest task-free routed entry per column is in bold. Last two rows: learners that never grow, on the same frozen features (for the T5 chain, its frozen input stream), local CPU. NCM keeps one mean per class and no examples and is deterministic (`ncm_reference.py`); RanPAC (McDonnell et al. 2023) accumulates the Gram matrix and class sums of a frozen random projection ($M = 10\,000$, ReLU) and picks its ridge by its own rule, mean of 3 seeds (`ranpac_reference.py`). Ridge on the unprojected features ($\lambda = 100$; accumulated statistics reach it exactly) gives 0.711, 0.941 and 0.941.

	Split	Permuted	Rotated	CIFAR-100	CLINC150	T5 chain
PNN + task label (reference)	0.978	0.952	0.939	0.804	0.919	–
$P(H{=}1)$ (bank ceiling)	0.978	0.962	0.960	0.804	0.919	0.953
R0 newest block	0.199	0.273	0.417	0.160	0.180	0.361
R1 uniform	0.717	0.763	0.779	0.503	0.694	0.823
R4m MoCL keys	0.610	0.863	0.850	0.385	0.640	0.804
R8a shared scorer	0.199	0.326	0.434	0.160	0.187	0.488
R11 input density	0.552	**0.952**	**0.890**	0.506	0.786	0.854
R14 max-softmax	0.717	0.913	0.832	0.503	0.694	0.860
R15 SupSup	0.716	0.849	0.798	**0.507**	0.703	0.855
R16 Expert Gate	0.634	0.643	0.722	0.473	0.661	0.761
R17 own-space distance	0.831	0.768	0.867	0.419	**0.821**	**0.931**
R18 R17 + confidence	**0.838**	0.895	0.871	0.470	0.785	0.926
R19 R18 + input density	0.818	0.948	0.884	0.496	0.803	0.927
R20 + competence	0.798	0.948	0.875	0.448	0.762	0.925
R20r + competence within a group	–	–	–	0.496	0.803	0.927
NCM, no growth	–	–	–	0.581	0.873	0.873
RanPAC, no growth	–	–	–	0.747	0.962	0.962

and Rotated-MNIST; on CIFAR-100 SupSup, input density and max-softmax are level, and own-space distance is last. R19, which sums the three, is within 0.02 of the best rule on each of these six streams, while each single score is at least 0.1 below the best on some stream. This pattern was noticed after the screen and is not a declared criterion; we report it and claim nothing from it. A cross-benchmark rank correlation (Kendall's τ over the 12 task-free routers at memory 0, shared trace) quantifies the instability: mean $\tau = +0.47$ across the ten benchmark pairs, ranging from $+0.85$ (Permuted–Rotated, where input density dominates both) to $+0.13$ (Split–CIFAR-100, where the top five are within 0.01). R19 is the most rank-stable router (ranks 2–5 across all five streams); input density swings between rank 1 and rank 10. The scores are not new (Section 2); what the table shows is how they rank in this regime once the three safeguards are in place, without which R19 fails (Table 7).

Where the remaining accuracy goes

Because the mixture term of Eq. (5) is negligible, a router's gap to $P(H{=}1)$ is selection error. With the three safeguards in place it is large, and its share depends on router and stream. On Split CIFAR-100 the bank holds 0.804 and the best per-block rule reaches 0.507; on CLINC150 the figures are 0.919 and 0.821. Both are below a nearest-class-mean classifier on the same frozen features (0.581 and 0.873) and well below RanPAC (0.747 and 0.962). An analytic shared router narrows the gap (0.612 and 0.901, Section 10.2), so that on CLINC150 most of its remaining loss is in the bank (0.08) rather than in selection (0.02), and clears the nearest-class-mean bar but not RanPAC; on CLINC150 even the task-labelled reference (0.919) stays below RanPAC. In the T5 chain, where the blocks change the representation, own-space distance reaches 0.931, above the nearest-class-mean classifier on the chain's frozen input (0.873) but below RanPAC on it (0.962). In all three streams the blocks start from a frozen pretrained feature, and a learner that never grows, given that feature, is the stronger learner: the safeguards cannot recover an advantage that the growth mechanism itself does not earn.

The traces separate the published rules (Table 11). Expert Gate, designed for one block per task, is the best rule on the aligned trace (tied on Permuted-MNIST) and loses 0.23 to 0.31 when a block trains on two periods (shared; Rotated-MNIST 0.954 to 0.722); with a router memory of 200 rows it recovers (gap

Table 11 Growth traces (CI; aligned and shared: run 38242387262, 5 seeds; the others: run 38325750828, seeds 0–2). Final accuracy at memory 0, buffer 500. The Rotated-MNIST cells of that run were lost to a resource limit; they are in Table 12, from one other platform.

	Split-MNIST							Permuted-MNIST						
Trace	$P(H)$	R0	R4m	R11	R14	R16	R19	$P(H)$	R0	R4m	R11	R14	R16	R19
aligned	.997	.199	.857	.712	.614	.890	.845	.970	.277	.955	.955	.908	.955	.953
shared	.978	.199	.610	.552	.717	.634	.818	.962	.273	.863	.952	.913	.643	.948
every-3	.944	.381	.611	.743	.801	.704	.853	.950	.447	.878	.945	.924	.650	.939
every-4	.926	.199	.437	.530	.686	.642	.814	.940	.281	.878	.931	.898	.839	.928
split	.997	.199	.817	.746	.319	.890	.735	.972	.284	.953	.953	.899	.951	.951
combined	.982	.199	.713	.630	.428	.770	.738	.970	.286	.881	.953	.920	.839	.952

Table 12 Growth traces on the local CPU (`run_local_sweeps.sh`, seeds 0–2; numbers from `sweeps_local_table.py`): the Rotated-MNIST cells the CI run lost, and the two traces added after stage D, mismatched timing and revisit, on all three MNIST benches with a local aligned reference. Final accuracy at memory 0, buffer 500; for revisit, the mean over the five distinct distributions. One platform; not compared with Table 11.

Bench	Trace	$P(H)$	R0	R4m	R11	R14	R16	R19	R20r
Rotated	aligned	.981	.400	.904	.930	.820	.956	.920	.920
	shared	.961	.409	.849	.891	.826	.722	.884	.884
	every-3	.931	.540	.781	.861	.840	.708	.868	.868
	every-4	.891	.426	.713	.837	.788	.765	.827	.827
	split	.984	.414	.895	.931	.800	.955	.919	.919
	combined	.977	.406	.883	.914	.803	.846	.903	.897
	mismatched	.971	.453	.842	.908	.841	.850	.887	.886
	revisit	.987	.501	.899	.934	.798	.958	.924	.924
Split	aligned	.997	.199	.862	.711	.610	.890	.844	.844
	mismatched	.991	.198	.649	.616	.735	.729	.825	.822
	revisit	.995	.199	.815	.702	.386	.891	.782	.782
Permuted	aligned	.971	.284	.955	.955	.915	.955	.954	.954
	mismatched	.973	.373	.938	.950	.935	.802	.944	.944
	revisit	.979	.277	.960	.959	.905	.957	.958	.958

−0.156 / 0.000 / −0.008 on the three MNIST benches), which is consistent with its per-block autoencoder forgetting the block's first period while the block learns the second. R19 holds its accuracy when blocks span two to four periods. When two blocks share one period (split and combined traces, Split-MNIST), max-softmax, input density and R19 lose, R14 most (0.319 on split), and Expert Gate is best (0.890 and 0.770 against R19's 0.735 and 0.738). That case, several blocks holding the same classes, is not solved by any of our rules; it is the clearest open problem the traces expose.

9.5 Births off period boundaries and revisited distributions

Rotated-MNIST

The untouched bench repeats the pattern of the development benches (Table 12). Expert Gate is best on the aligned and split traces (0.956, 0.955) and falls to 0.722 when a block trains on two periods. R19 holds 0.83 to 0.92 on every trace, and input density, the term that carries this bench (Table 17), is level with it or up to 0.021 above it on every trace except every-3. Two blocks within one period cost nothing here (R19 0.919 against 0.920 aligned): the labels repeat, so both blocks are competent on their shared period, unlike the class-incremental case of Section 9.4.

Births off period boundaries

In the mismatched trace each block trains on the end of one period and the first 30% of the next, at the aligned trace's block count. Storage is intact: $P(H{=}1)$ moves by at most 0.010. The loss is in selection, and it falls on the rules that fit one descriptor per block while the block trains: Expert Gate loses 0.11

Table 13 The version fork, pair replay vs. forgot on the block's first period (H4). Pick: how often the router picks the version that is right, on rows where exactly one version is; routed: hard-selection accuracy on that period. Ten runs per bench (seeds 10–14 × two orders), local CPU.

	Split		Permuted		Rotated		CIFAR-100	
Rule	pick	routed	pick	routed	pick	routed	pick	routed
R11 input density	0.500	0.471	0.500	0.836	0.500	0.670	0.500	0.364
R14 max-softmax	0.692	0.653	0.887	0.935	0.806	0.815	0.811	0.591
R17 own-space distance	0.281	0.268	0.477	0.827	0.332	0.571	0.194	0.142
R19	0.490	0.464	0.870	0.930	0.751	0.782	0.718	0.524
R20	0.972	0.916	0.931	0.945	0.941	0.877	0.931	0.678
R20r	0.971	0.915	0.921	0.942	0.920	0.866	0.907	0.660
replay version alone		0.942		0.949		0.890		0.728
forgot version alone		0.000		0.723		0.451		0.000

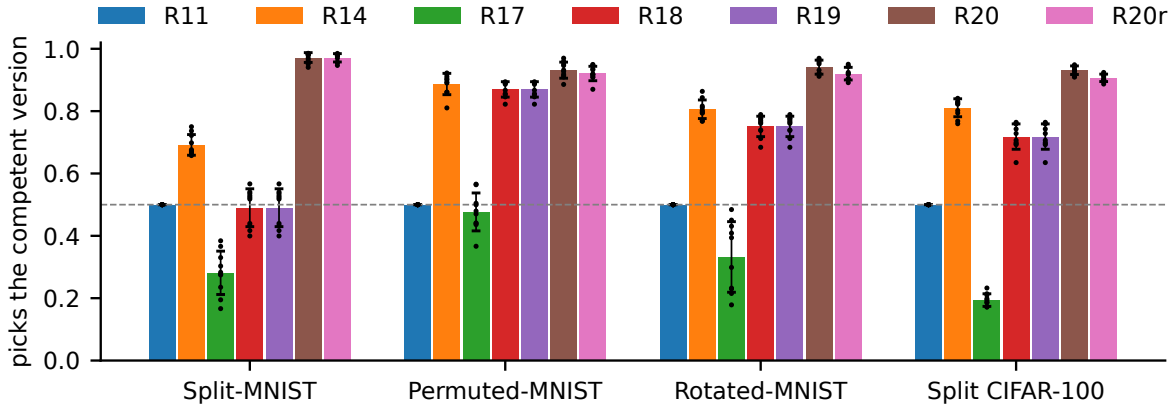


Fig. 3 The fork's decisive rows (Table 13): rate at which each rule picks the competent version, one dot per run; 0.5 (dashed) is a tie. Local CPU.

to 0.16 and MoCL keys 0.02 to 0.21. R19, whose records are written at freeze, loses 0.010 to 0.033; it is the best rule on Split-MNIST (0.825; max-softmax 0.735), and input density stays best on Permuted- and Rotated-MNIST (0.950, 0.908). These runs exposed an implementation gap: a block frozen inside a period, after an earlier period, lost its record of the earlier one, because its buffer had been cleared at freeze before that period's record reset. The tested fix hands those rows over at freeze; before it, R19 fell to 0.628 on Split-MNIST, one more instance of safeguard 2. No other trace freezes such a block, so no other number changes.

Revisited distributions

When periods 0 and 1 return with fresh training rows, a second block learns each. Every router splits its mass on those rows between the old and the new block (R19: 0.34/0.34, 0.49/0.51 and 0.39/0.43 on Split-, Permuted- and Rotated-MNIST), and R19's accuracy on the revisited distributions rises from before the revisit to the end (0.879 to 0.937, 0.951 to 0.963, 0.917 to 0.949). On Permuted- and Rotated-MNIST R19's mean over the five distributions is slightly above aligned (0.958 and 0.924 against 0.954 and 0.920). On Split-MNIST the revisit blocks also attract rows of periods they never saw: R19 loses 0.13 on periods 2–4 (0.811 to 0.679) and max-softmax 0.51 (0.598 to 0.084), while Expert Gate, which scores the input alone, is unaffected (0.891). A returning distribution is thus another instance of the open case above, several blocks holding the same classes.

Table 14 E2, versions that arise without construction (stage F seeds 20–29, local CPU, hard selection). "Final": the final blocks only (R20r equals R19 there); "+v": plus each revised block's snapshot at the end of its first period. Pair pick: rate at which the rule prefers the right version of a revised block (H6), first period, rows where exactly one version is right. Best of the rules shown, with versions, in bold (R11 and R14 are in the released results; R11 reaches 0.954 on Permuted-MNIST at replay 0).

		Split	Permuted	Rotated	CIFAR-100	CLINC150
replay 0	ceiling, final / +v	.597 / .997	.891 / .971	.910 / .982	.451 / .906	.467 / .939
	R19, final	.491	.862	.806	.296	.397
	R19, +v	.777	.948	**.904**	**.467**	.706
	R20, +v	**.826**	**.951**	**.904**	.441	**.757**
	R20r, +v	.622	.948	.825	.304	.488
	R21, +v	.609	.911	.898	.295	.628
	pair pick R19 / R20r	.829 / .976	.905 / .926	.874 / .919	.758 / .881	.790 / .968
replay 500	ceiling, final / +v	.978 / .996	.962 / .972	.962 / .981	.804 / .912	.918 / .944
	R19, final	.839	.948	.897	.532	.807
	R19, +v	**.840**	.952	.915	**.539**	.815
	R20, +v	.836	**.953**	.911	.464	.779
	R20r, +v	.776	.952	.911	.362	.796
	R21, +v	.788	.952	**.946**	.422	**.869**
	pair pick R19 / R20r	.853 / .871	.675 / .679	.759 / .772	.755 / .817	.749 / .764

9.6 Versions: the fork

On the fork, input density ties by construction (0.500 everywhere), and own-space distance alone prefers the *wrong* version on three of the four benches (Table 13, Figure 3): where a version's data lies in its own space says nothing about whether it is still right there. Max-softmax separates the versions partly, because the forgetting version is less confident on the classes it lost. The competence record separates them best: R20r picks the competent version on 0.91 to 0.97 of the decisive rows, and its routed accuracy is within 0.03 of the replay version alone on the MNIST benches. On CIFAR-100 it closes about two thirds of the distance between R19 and the replay version (0.660 against 0.524 and 0.728). The uncentred R20 picks slightly more often on the fork (by 0.00 to 0.02), where both candidates share a group and centring only removes signal; centring is there for the disjoint blocks of the main screen, where R20 costs up to 0.049 against R19 and R20r costs nothing (Table 7). On CLINC150 (not in Table 13) R20r picks the competent version on 0.97 of the decisive rows, against R19's 0.78.

The one failed guard row is on Split-MNIST, pair replay vs. forgot, the block's later period: R20r's pick rate is 0.427 against R19's 0.681 (-0.254; R20 0.372, so the competence record itself causes it). On that period both versions are right on almost every row (0.995 and 0.997 alone), so decisive rows are few, and routed accuracy is unchanged (0.996 for R20r, 0.997 for R19). The same row passes on the other four benchmarks (-0.004, -0.024, -0.037, and -0.012 on CLINC150). We did not tune R20r further on the stage-D seeds.

9.7 Versions that arise without construction

The fork builds its two versions. E2 asks whether the competence record still works when versions arise on their own and the router faces every block at once. In the shared trace a block learns two periods; a system that keeps versions holds, besides the final blocks, the frozen snapshot of each such block at the end of its first period. We route the test rows of all periods over the final blocks alone and over the final blocks plus these snapshots, on the safeguard-1 bank (replay 500), on banks with replay 0 and 100, and on the stage F seeds. Every candidate gets the same 500 record rows per period it saw, written through its own weights; R21 gets the same rows, so it is weaker here than in the main screen, where it accumulates every training row. H6, declared in advance, compares R20r and R19 on the rate at which they pick the right one of a block's two versions, on the block's first period at replay 0.

H6 holds on all five benchmarks: R20r picks the right version more often than R19 ($+0.021$ to $+0.178$, 10/10 runs each, Holm $p = 0.0059$; Table 14). On CLEAR-10 it is not shown (-0.000), since there exactly one version is right on only 3% of the rows. The competence record therefore works as it did on the fork, also on versions that nobody constructed. Keeping the versions pays: at replay 0 the snapshots hold what

Table 15 Stage G, E5: why centring costs accuracy when versions are kept (fresh seeds 30–39, local CPU, replay 0, final blocks plus each revised block's first-period snapshot, hard selection, means over ten seeds). Intrusion: share of test rows of periods a revised block never learned whose top choice is one of its versions (mean over revised blocks). Loss share: fraction of the rows R20r loses against R20 that are such intrusions. On CLEAR-10 every block covers every year, so intrusion costs nothing there.

		Split	Permuted	Rotated	CIFAR-100	CLINC150	CLEAR-10
accuracy	R19	.771	.948	.905	.468	.711	.958
	R20	.816	.951	.905	.448	.756	.958
	R20r	.624	.946	.824	.301	.492	.958
intrusion	R19	.118	.005	.077	.110	.046	.193
	R20	.108	.004	.076	.121	.050	.194
	R20r	.287	.009	.186	.164	.120	.194
loss share, R20r vs. R20		.99	.65	.96	.99	1.00	.81

the active block later forgot, the ceiling rises from 0.45–0.91 to 0.91–0.997, and R19's end-to-end accuracy rises by 0.09 to 0.31 (10/10 runs on every benchmark; exploratory). At replay 500 the gain is at most 0.02, and the replay bank without versions still beats the replay-0 bank with them on CIFAR-100 (R19 0.532 against 0.467).

End to end, however, R20r is the wrong rule once versions are kept. At replay 0 it is below the uncentred R20 on every run on Split-, Rotated-MNIST, CIFAR-100 and CLINC150 (-0.08 to -0.27; Permuted-MNIST -0.003; exploratory), and at replay 500 it falls below R19 by 0.06 on Split-MNIST and 0.18 on CIFAR-100. The cause is the centring of Eq. (4). For an input that belongs to another block, the two versions of a revised block often still share their nearest group; centring then gives the more competent version a bonus of half the gap in log-competence divided by τ_r, while a block alone in its group receives zero. With $\tau_r = 0.05$ a gap of 0.5 is worth 5 in score units, so the pair draws inputs it does not own. R20 has no such bonus: every block's term is the log of its own competence, at most zero, so the pair gains nothing from sharing a group; a version can still outscore a lone block through competence or familiarity. Per-row picks stay high inside the pair (0.93–0.99 at replay 0) while end-to-end accuracy falls.

Stage G tested this account on fresh seeds, declared in advance (Table 15). R20 beats R20r end to end with the versions present on all six benchmarks (H8, 10 of 10 runs on the five benchmarks above and 7 of 10 on CLEAR-10, Holm $p \leq 0.021$), by 0.08 to 0.26 on Split-, Rotated-MNIST, CIFAR-100 and CLINC150 and by less than δ on Permuted-MNIST (+0.005) and CLEAR-10 (+0.0004). R20r's intrusion exceeds R20's on the five benchmarks (H9, 10 of 10, Holm $p = 0.0059$; Permuted-MNIST +0.005, below δ), and not on CLEAR-10. Row by row, 96 to 100% of the rows R20r loses against R20 are intrusions on Split-, Rotated-MNIST, CIFAR-100 and CLINC150, and 65% on Permuted-MNIST. R20 and R20r differ only in the centring term, so H8 is an intervention on the centring with every other part of the score fixed, and the row-by-row split locates its cost: the centring costs accuracy because version pairs take over inputs that belong to other blocks. On the same seeds R20r again picks the right version more often than R19 (H6 repeated as a secondary check: +0.021 to +0.169, 10 of 10 runs on each of the five benchmarks). The centring is therefore right for choosing between the versions of one block and harmful for routing over a bank that keeps versions beside other blocks. In that setting the uncentred R20 leads at replay 0 on Split-MNIST and CLINC150 and is within 0.03 of R19 elsewhere; at replay 500, where versions differ little, R19 or R21 leads. A rule that centres only when the input lies in the shared group's region remains to be designed and tested.

9.8 In the growing chain

In the chain a new block starts as an identity on the previous endpoint and its head as a copy, yet the newest endpoint (R0) still loses most of the old periods (Table 16). The ordering of the rules is close to that on the columns of the same data: R19 leads on Split-MNIST and is level with input density, or up to 0.012 below it, on Permuted- and Rotated-MNIST, the streams where input density carries the routing on the columns too. The UCLBR selector (R12) (ours, earlier), designed for the chain, is below R19 by 0.04 to 0.20. Inside FLAN-T5 the safeguards hold as on the columns: without the block buffer R19 falls

Table 16 The growing chain: every endpoint from one forward pass. Final accuracy at memory 0, seeds 0–2 per MNIST column and 0–3 for the T5 chain. Each column comes from one platform: Split and Permuted saturation and Split shared, CI run 38325750828 (the development benches); columns marked *, local CPU (the cells the CI run lost to the resource limit, rerun by `run_local_sweeps.sh`); T5 chain inside FLAN-T5-base on CLINC150, Colab T4. Highest task-free entry per column in bold.

	MNIST chain, saturation			MNIST chain, shared			T5 chain	
	Split	Permuted	Rotated*	Split	Permuted*	Rotated*	shared	saturation
$P(H{=}1)$	0.927	0.900	0.930	0.957	0.856	0.939	0.953	0.949
R0 newest block	0.363	0.238	0.424	0.192	0.255	0.446	0.361	0.420
R1 uniform	0.704	0.579	0.727	0.630	0.536	0.744	0.823	0.805
R4m MoCL keys	0.610	0.747	0.766	0.618	0.720	0.816	0.804	0.768
R11 input density	0.745	**0.874**	**0.858**	0.548	**0.819**	**0.857**	0.854	0.835
R12 UCLBR selector	0.601	0.743	0.763	0.608	0.718	0.813	–	–
R14 max-softmax	0.704	0.688	0.778	0.631	0.621	0.772	0.860	0.851
R16 Expert Gate	0.689	0.775	0.770	0.618	0.562	0.711	0.761	0.754
R17 own-space distance	–	–	–	–	–	–	**0.931**	**0.921**
R19	**0.800**	0.865	**0.858**	**0.773**	0.807	0.848	0.927	0.911
R20	0.733	0.856	0.853	0.729	0.799	0.835	0.925	0.905
R20r	0.800	0.865	0.858	0.773	0.807	0.848	0.927	0.911

Table 17 Owner top-1: how often the block that owns a test row (the last block trained on its period) scores highest, per term of R19 and for the competence term alone; shared trace, buffer 500, memory 0. MNIST: CI, 5 seeds; CIFAR-100: CI, 3 seeds; CLINC150 columns: local CPU, 3 seeds. Chance is 1/3 on MNIST (three blocks) and 1/5 on the ten-period streams. Last row: the single-score rule with the highest final accuracy on that bench (Table 10).

Term	Split	Permuted	Rotated	CIFAR-100	CLINC150
own-space distance (R17)	**0.830**	0.598	0.751	0.421	**0.851**
confidence (R14)	0.722	0.934	0.746	**0.534**	0.709
input density (R11)	0.522	**0.999**	**0.769**	0.461	0.807
R19 (sum)	0.827	0.990	0.824	0.532	0.825
competence alone (R20 term)	0.324	0.356	0.321	0.190	0.147
best single-score rule	R17	R11	R11	R15 ≈ R11 ≈ R14	R17

from 0.927 to 0.553, online records cost 0.02 and removing the own-space distance 0.05, each in 4 of 4 seeds. The shared scorer (R8a) falls 0.54 to 0.56 below R19 in three seeds, but only 0.11 below in the fourth (0.823 against 0.929), so in this chain a shared scorer collapsed less; we report it as an exception to safeguard 3 rather than as noise. In the chain, own-space records alone (R17) are level with R19: removing confidence raises R19 by 0.003 and removing input density lowers it by 0.001. R20r equals R19 on every chain run, as designed, and so do the ten-run MNIST chains of stage D (0.799 / 0.872 / 0.858, local CPU), where R20 is below them (0.740 / 0.864 / 0.853).

The saturation rule did not produce irregular traces: in every MNIST run the first block was frozen at the three-period cap (births at periods 0 and 3, plus 4 on Permuted- and Rotated-MNIST when the second block saturated), and in the T5 chain every birth came at the cap (periods 0, 3, 6, 9), so the fixed traces carry the variation. Without the cap the rule never fired (local CPU, seeds 0–2): its forgetting reading never rose above 0 and its fit reading never fell below 0.97, because both are taken on rows the active block replays, so one block learned all five periods. That single block, with the same 500-row buffer, reached 0.838, 0.917 and 0.814 on Split-, Permuted- and Rotated-MNIST, against 0.798, 0.865 and 0.858 for R19 on the capped chain on the same machine (bank ceilings 0.927, 0.902 and 0.930). On two of three streams the learner that did not grow beat the routed bank, and on Permuted-MNIST even what the grown bank held, as in the frozen-feature comparison (Table 10). The safeguards decide whether routing over grown blocks works; they do not make growing the better choice.

9.9 Why the scores work: calibration

The term that most often ranks the owner block first is the term of the best single-score rule on Split-, Permuted- and Rotated-MNIST and CLINC150 (Table 17). On CIFAR-100 the three single-score rules built on confidence or input density are within 0.004 of each other, and own-space distance, the term with the lowest owner top-1 there, is clearly last. The per-block AUROC for "this row is from my period" tells the same story where the terms differ widely (Split-MNIST: distance 0.88, input density 0.68; Permuted-MNIST: 0.69 and 0.94) but not on every bench, because it measures each block on its own and not the comparison between blocks. Calibration on held-out rows is therefore a cheap way to see which score a stream needs. Competence alone is near or below chance at picking the owner block on every bench: a block's accuracy on its own groups tells how hard its period was and nothing about whether an input belongs to it. This is why R20, which compares competence across blocks, costs accuracy (Table 7), and why R20r centres it over the blocks whose nearest group for the input coincides (Eq. (4)).

9.10 Cost

Records are written once and kept; buffer rows are discarded at freeze. We count the floats a router must keep: its filled statistics, a symmetric matrix by its upper triangle and a random projection by its seed, for every router alike. After the shared-trace stream R19 keeps 31 to 35 thousand floats on MNIST and 73 to 82 thousand on CIFAR-100, CLINC150 and CLEAR-10; our implementation allocates a full covariance for every block–period pair, filled or not, and stores 0.16 to 0.58 million. Expert Gate's autoencoders take 0.16, 0.69 and 0.26 million on MNIST, CIFAR-100 and CLINC150, input density and MoCL keys 5 to 20 thousand, and max-softmax none. The block buffer holds 500 rows of the input (0.39 million floats on MNIST, 1.0 million on CIFAR-100 features) while its block trains. R19's record grows with the square of the hidden width, one covariance per period a block trained on: in the T5 chain (width 768, ten block periods) the covariances alone are 3.0 million floats, and our implementation stores 30.1 million. R20 adds two counts per group. R21 (Section 10.2) keeps one shared Gram matrix over its random projection (4.5 million floats at $M = 3000$) and one column of cross-correlations per block, 55 to 146 times what R19 keeps; as implemented, with the full Gram matrix and the projection stored, it takes 11.3 to 15.2 million. RanPAC's accumulated statistics (Table 10) take 51 million floats at $M = 10\,000$. At $M = 1000$ they take 0.60 and 0.65 million on CIFAR-100 and CLINC150, about R19's allocation and 7 to 9 times what it keeps, and RanPAC still reaches 0.678 and 0.934 (local CPU, 3 seeds), above every routed result on those streams. On a column bank every candidate runs for every input, so the cost of a prediction grows with the number of blocks; in the chain all endpoints come from one pass.

10 Discussion

10.1 What the results show and what they leave open

What is ours

The scores are published. The contribution is the regime, the three safeguards, each of whose removal lowered accuracy in a confirmation on independent seeds, and the finding that a competence signal which picks correctly between the versions of one block damages routing across the bank. A router built from the published scores under these safeguards did not beat the strongest published rule on every bench, and we do not claim that it does; which score wins depends on whether blocks differ more in their inputs or in their hidden spaces, and per-block calibration shows which.

What the safeguards do not fix

The safeguards remove the losses the regime adds but do not close the gap that task-free selection leaves. Under each stream's per-period test RanPAC outperforms every routed bank on frozen features (Section 9.4), and R21, given the per-block router's record rows and the floats it keeps, stays ahead of that router only on CLINC150 (Section 10.2). We do not read this as an argument against growth in general: frozen features suit a learner that never grows, while growth is meant for streams that need the representation to change, which ours do not test. What the result does establish is a reporting standard: a

method paper for this regime should give the bank's ceiling, the routed accuracy and the best learner that never grows on the same features, side by side.

What the confirmation does and does not show

The family tests removals from one construction (R19) on one architecture family per bench and a small network. It does not show that a rule keeping no state (max-softmax, SupSup) needs safeguards 2 and 3; such rules satisfy them trivially and remain competitive on CIFAR-100. H1 removes the buffer and with it the freeze-time rebuild, so safeguard 1 is isolated by the bank's ceiling, which no router enters. Stages D and F share the architecture and benchmarks.

The competence record

R20r was designed after a development run and revised once on the development fork (Section 5.2); its evidence is stage F on independent seeds and stage D on CIFAR-100, untouched by either step. A system that keeps versions should route without the centring (R20 at replay 0, R19 when replay keeps the versions close), or use R20r only to choose between versions of one block; a centring that serves both cases is open.

Scope of the evidence

Two scales of R19 were fixed on the development benches; Rotated-MNIST, every other seed, CIFAR-100, CLINC150 and the chains are untouched by that choice. The networks are small and the image and text features frozen; only the T5 chain trains transformer layers (four seeds), and it is a screen, outside stages D and F. CLEAR-10, the one stream with natural drift, is too easy for its blocks to differ under either test protocol, so the safeguards are untested on natural drift that does separate blocks. Runs were spread over three platforms because of compute limits; each comparison uses one. Our saturation rule mostly fired at its period cap and, uncapped, never fired: it read saturation on rows its block replays, so it illustrates growth rather than testing a controller. Births off period boundaries cost the freeze-time construction only 0.01 to 0.03, but several blocks holding the same classes remain unsolved by every rule we tested except Expert Gate.

10.2 Analytic and prototype routers

Analytic, prototype and ensemble routers

Per-block frozen state is one way to meet safeguard 3; a shared router with an exact closed-form solution is another. We test three further router families on the same PNN bank (Table 18; local CPU, shared trace).

An analytic RLS router (R21) maps the frozen input through a fixed random ReLU projection, as RanPAC does, and accumulates one shared Gram matrix and, per block, the cross-correlation with an indicator of the block that trained on each row (the declared use (i) of period information, Section 3); routing weights are solved in closed form. It reached 0.946 on Split-MNIST, 0.612 on CIFAR-100 and 0.901 on CLINC150, the highest task-free result on each stream and 0.10 to 0.14 above R19, and $\lambda \in \{1, 100, 1000\}$ gave identical accuracies. Unlike the gradient-trained shared scorer (R8a, 0.199 / 0.162 / 0.187), it needs no replay and does not collapse to the newest block. The comparison is not matched in information or memory: R21 accumulates every training row and keeps 55 to 146 times R19's floats (Section 9.10), while R19 writes a block's record from its last period's rows and its 500-row buffer. Stage F confirmed R21>R19 in 50 of 50 independent runs (Table 9), as stage D had found in its 50 dependent runs. Exact shared statistics thus avoid the shared scorer's collapse; stage G tests them at equal budget.

Budget and information

Stage G separates R21's two advantages, the rows it learns from and the floats it keeps, on the stage-F banks (Table 19). E4's pre-declared budget was the floats R19's implementation allocates, of which R19 fills 13 to 21 %; there R21 stays above R19 on all five benchmarks (H7a), on four with R19's record rows only (H7b), and with both limits only on CLINC150 (H7c). E4K repeats the test at the floats R19 keeps (M of 245 to 399): R21 stays above R19 in every run on Split- and Rotated-MNIST, CIFAR-100 and CLINC150 (H10a, +0.024 to +0.061) and ties on Permuted-MNIST (−0.001); with R19's record rows

Table 18 Additional routers on the shared-trace benches (local CPU, memory 0). R21: recursive-least-squares shared router over a fixed random ReLU projection of the input (M=3000, λ=100; Any-SSR/REAR approach). R22: cosine similarity to per-block class-mean prototypes in the frozen input space (τ=0.05). R23: reciprocal rank fusion of R19, R14 and R11 (k=10; k=60 gave 0.762, 0.520 and 0.710). Existing comparators are re-run on the same platform for a fair within-table comparison; absolute values are not directly comparable to Table 10, which used a different platform. The Δ_{500} columns show the change from memory 0 to memory 500 (CLINC150 and CIFAR-100 only, five sizes from 0 to 500). Highest task-free entry per column in bold. Accuracy columns: 3 exploratory seeds. R21>R19 is confirmed separately in stage F (Table 9).

	Final accuracy, M=0			Δ_{500}	
	Split	CIFAR-100	CLINC150	CIFAR	CLINC
R0 newest block	0.199	0.162	0.180	.000	.000
R8a shared scorer	0.199	0.162	0.187	+.148	+.631
R14 max-softmax	0.722	0.503	0.694	.000	.000
R11 input density	0.552	0.505	0.786	–	–
R19	0.809	0.494	0.803	+.014	.000
R21 RLS shared	**0.946**	**0.612**	**0.901**	−.006	−.001
R22 cosine proto	0.851	0.593	0.893	.000	+.001
R23 rank fusion	0.780	0.537	0.762	.000	.000

Table 19 Stage G, E4 and E4K: R21 under R19's budgets (the stage-F banks, seeds 20–29, local CPU, replay 500, memory 0; final accuracy, mean over ten seeds). R21-mem: projection width M chosen so that the floats R21 keeps fit the floats R19's implementation allocates (158 to 584 thousand), 5 to 8 times what R19 keeps. R21-kept: M fitted to the floats R19 keeps, shown in brackets (Section 9.10). R21-rec: R21 fed only the rows R19's records are written from. R21-rm, R21-rk: both limits. Marks on the differences to R19: $^{+}$holds (10 of 10 or 9 of 10 runs, Holm $p = 0.0059$), $^{\circ}$holds below $\delta = 0.01$, $^{-}$below R19 in 9 or 10 of 10 runs; unmarked: not shown.

	Split	Permuted	Rotated	CIFAR-100	CLINC150	CLEAR-10
R19 (thousand floats kept)	.823 (31)	.948 (35)	.887 (35)	.487 (81)	.804 (73)	.959 (82)
R21, every row, $M = 3000$	.947	.950	.940	.617	.899	.959
R21-mem, R19's allocation	$.907^{+}$	$.949^{\circ}$	$.928^{+}$	$.581^{+}$	$.890^{+}$	.959
R21-rec, R19's rows	.822	$.949^{\circ}$	$.929^{+}$	$.506^{+}$	$.886^{+}$	.960
R21-rm, both	$.739^{-}$	$.943^{-}$	$.873^{-}$	$.475^{-}$	$.872^{+}$	.959
R21-kept, R19's kept floats	$.872^{+}$	.946	$.910^{+}$	$.530^{+}$	$.865^{+}$	.959
R21-rk, both	$.703^{-}$	$.932^{-}$	$.859^{-}$	$.443^{-}$	$.833^{+}$	.959
M: R21-mem, -rm / -kept, -rk	558/245	579/261	579/261	1063/397	1075/377	1068/399

as well it falls below R19 in 9 or 10 of 10 runs on the three MNIST streams and CIFAR-100 (−0.016 to −0.120) and stays ahead only on CLINC150 (+0.029, H10b). At equal memory, then, the rows R21 learns from, not its memory, carry its lead: given the same rows and floats, R19 is the better router on four of the five benchmarks. R19 and R21 also differ in their scores, so this does not show that per-block state beats shared state as such.

On the matched comparison (Table 20) R21 beats the nearest-class-mean bar, and RanPAC beats R21, on all ten seeds of all three streams, as on seeds 10–14. On CLEAR-10 the newest block alone beats NCM and R21 is 0.004 below RanPAC: one block covers the stream (Section 9.3).

A cosine-prototype router (R22), with per-block class means in the frozen input space, comes within 0.02 of R21 on CIFAR-100 and CLINC150 (0.593, 0.893) but falls 0.10 behind on Split-MNIST (0.851), where pixel-space prototypes discriminate poorly. A reciprocal-rank-fusion ensemble (R23) of R19, R14 and R11 beat R19 only on CIFAR-100 (0.537 against 0.494).

Table 20 Matched comparison on the frozen-feature streams: the same ten seeds (20–29), one platform (local CPU), mean ± s.d. Routers: replay-500 bank, shared trace, memory 0. Ceiling: oracle of the same seeds' replay-500 banks. NCM and joint ridge are deterministic. Descriptive; no hypothesis was declared about NCM or RanPAC.

	CIFAR-100	CLINC150	CLEAR-10
$P(H{=}1)$ (bank ceiling)	0.804	0.918	0.980
R0 newest block	0.160±.007	0.181±.003	0.950±.003
R14 max-softmax	0.507±.012	0.705±.017	0.959±.001
R19	0.487±.008	0.804±.008	0.959±.001
R21 RLS shared	0.617±.009	0.899±.006	0.959±.001
NCM, no growth	0.581	0.873	0.859
Joint ridge, no growth	0.711	0.941	0.957
RanPAC, no growth	0.752±.007	0.960±.004	0.963±.001

Memory sensitivity

Over replay memories of 0 to 500, R21 and R22 moved by at most 0.006 on CLINC150 and CIFAR-100 (Table 18); the gradient-trained shared scorer gains most (0.187 to 0.818 on CLINC150, 0.162 to 0.310 on CIFAR-100) yet stays below R21. R21's exact statistics keep every frozen block's evidence without per-block copies, though not its routing of earlier periods; without an exact solution, as for R8a, per-block state prevents collapse.

10.3 Limitations

Architecture and regime

All column banks are PNN columns whose 100-unit hidden layers compress frozen 2048-dimensional features, so a router's failure is a selection failure. Here growth itself does not earn its cost, a negative result about PNN columns on frozen features, not about routing. Whether the safeguards bind as strongly in richer blocks is untested beyond the T5 chain; a regime where growth pays is separate work.

Adaptations and absent comparators

Expert Gate, MoCL, L2R, SupSup, MORE and TPL are adapted to complete-output routing, each departure recorded. Routers over frozen encoders (SEMA (Wang et al. 2025a), MaRS (Yan 2026), Any-SSR (Tong et al. 2025), whose router we test as R21) and prompt- and adapter-based growers (L2P (Wang et al. 2022b), DualPrompt (Wang et al. 2022a), CODA-Prompt (Smith et al. 2023), EASE (Zhou et al. 2024)) bring their own growth and protocol; comparing them needs a full growing system.

11 Conclusion

When blocks outlive tasks, a task-free router over the replay-based growers tested here relies on three safeguards: the active block replays its earlier periods until it freezes, its records are written at freeze, and later training never overwrites the router's evidence about a frozen block. A pre-declared confirmation on ten independent seeds held 19 of 20 tests, each with a 95% interval above 0.01. With the safeguards in place selection remains a large loss, its share depending on router and stream, and RanPAC (McDonnell et al. 2023; Peng et al. 2025) outperforms every routed bank on every frozen-feature stream under each stream's per-period test. The safeguards keep routing from failing; closing the selection gap they leave is what would make growth pay.

Reproducibility

Each experiment is one resumable command: `bash experiments/output_routing/run_local_sweeps.sh` for the local-CPU sweeps, `run_e1_e3.sh` (stage F), `run_e4_e6.sh` and `run_e4k.sh` (stage G), pre-declared in `docs/`; the non-growing learners are `experiments/analysis/ncm_reference.py` and `ranpac_`

`reference.py`. Avalanche 0.6's non-blocking device copies corrupted data on Apple MPS in our tests; the released code copies blocking except on CUDA.

Acknowledgements. Not applicable.

Author contributions. **Nishant Kumar:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft. **Saurabh Srivastava:** Supervision, Writing – review & editing.

Funding. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability. All datasets are public: MNIST (LeCun et al. 1998), CIFAR-100 (Krizhevsky 2009), CLINC150 (Larson et al. 2019) and CLEAR-10 (Lin et al. 2021).

Code availability. Code and result files will be made available upon publication and are available on request from the corresponding author until then.

Declarations

Conflict of interest. The authors declare that they have no conflict of interest.

Ethical approval. Not applicable: no studies with human participants or animals.

Informed consent. Not applicable.

Use of generative AI. During the preparation of this work, the authors used Claude (Anthropic) to assist with drafting, editing, and formatting the manuscript text. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

ORCID Nishant Kumar https://orcid.org/0009-0009-0156-555X · Saurabh Srivastava https://orcid.org/0009-0006-8020-7491

References

Aljundi R, Chakravarty P, Tuytelaars T (2017) Expert gate: Lifelong learning with a network of experts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3366–3375

Araujo V, Moens MF, Tuytelaars T (2024) Learning to route for dynamic adapter composition in continual learning with language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024. Association for Computational Linguistics, Miami, Florida, USA, pp 687–696, https://doi.org/10.18653/v1/2024.findings-emnlp.38

Carta A, Pellegrini L, Cossu A, et al (2023) Avalanche: A PyTorch library for deep continual learning. Journal of Machine Learning Research 24(363):1–6

Chung HW, Hou L, Longpre S, et al (2024) Scaling instruction-finetuned language models. Journal of Machine Learning Research 25(70):1–53

De Lange M, Aljundi R, Masana M, et al (2022) A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(7):3366–3385

Feng T, Shen Y, You J (2025) GraphRouter: A graph-based router for LLM selections. In: International Conference on Learning Representations (ICLR)

Giannini F, Ziffer G, Della Valle E (2023) cPNN: Continuous progressive neural networks for evolving streaming time series. In: Advances in Knowledge Discovery and Data Mining (PAKDD). Springer, Lecture Notes in Computer Science, pp 328–340, https://doi.org/10.1007/978-3-031-33383-5_26

Giannini F, Ziffer G, Della Valle E (2026) Dynamic continuous progressive neural networks for evolving streaming time series. Data Mining and Knowledge Discovery 40(4):49. https://doi.org/10.1007/s10618-026-01213-y

Guo Z, Cheng Q, Zhou DW, et al (2026) Stable routing for mixture-of-experts in class-incremental learning. Preprint, arXiv:2605.17571

He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 770–778

Holm S (1979) A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2):65–70

Jiang J, He L, Fang Z (2026) CERA-MoA: Co-evolving routing mechanisms with continually learning LLM agents. Preprint, arXiv:2609.18779

Jitkrittum W, Narasimhan H, Rawat AS, et al (2026) Universal model routing for efficient LLM inference. In: International Conference on Learning Representations (ICLR), arXiv:2502.08773

Kim G, Liu B, Ke Z (2022a) A multi-head model for continual learning via out-of-distribution replay. In: Proceedings of The 1st Conference on Lifelong Learning Agents, pp 548–563

Kim G, Xiao C, Konishi T, et al (2022b) A theoretical study on solving continual learning. In: Advances in Neural Information Processing Systems

Kolter JZ, Maloof MA (2007) Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8:2755–2790

Krizhevsky A (2009) Learning multiple layers of features from tiny images. Tech. rep., University of Toronto

Larson S, Mahendran A, Peper JJ, et al (2019) An evaluation dataset for intent classification and out-of-scope prediction. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, Hong Kong, China, pp 1311–1316, https://doi.org/10.18653/v1/D19-1131

LeCun Y, Bottou L, Bengio Y, et al (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324

Lee K, Lee K, Lee H, et al (2018) A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Advances in Neural Information Processing Systems, pp 7167–7177

Lee S, Ha J, Zhang D, et al (2020) A neural Dirichlet process mixture model for task-free continual learning. In: International Conference on Learning Representations (ICLR)

Lee S, Kim D, Yoo S, et al (2026) CARE-MoE: Conflict-aware routing-guided expert mixture-of-experts for continual learning. Mathematics 14(19):3557. https://doi.org/10.3390/math14193557

Lin H, Shao Y, Qian W, et al (2024) Class incremental learning via likelihood ratio based task prediction. In: International Conference on Learning Representations

Lin Z, Shi J, Pathak D, et al (2021) The CLEAR benchmark: Continual LEArning on real-world imagery. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/2838023a778dfaecdc212708f721b788-Abstract-round2.html

Madotto A, Lin Z, Zhou Z, et al (2021) Continual learning in task-oriented dialogue systems. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp 7452–7467

McDonnell MD, Gong D, Parvaneh A, et al (2023) RanPAC: Random projections and pre-trained models for continual learning. In: Advances in Neural Information Processing Systems

Mensink T, Verbeek J, Perronnin F, et al (2013) Distance-based image classification: Generalizing to new classes at near-zero cost. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(11):2624–2637. https://doi.org/10.1109/TPAMI.2013.83

Peng L, Elenter J, Agterberg J, et al (2025) LoRanPAC: Low-rank random features and pre-trained models for bridging theory and practice in continual learning. In: International Conference on Learning Representations

Raffel C, Shazeer N, Roberts A, et al (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research 21(140):1–67

Rusu AA, Rabinowitz NC, Desjardins G, et al (2016) Progressive neural networks. Preprint, arXiv:1606.04671

Rypeść G, Cygert S, Khan V, et al (2024) Divide and not forget: Ensemble of selectively trained experts in continual learning. In: International Conference on Learning Representations (ICLR)

Shihab IF, Al Ahsan ASAMMI, Swaqeeb MN (2026) Opportunity is not realizability: Selection-valid diagnostics for multi-LLM routing. Preprint, arXiv:2608.08265

Smith JS, Karlinsky L, Gutta V, et al (2023) CODA-Prompt: COntinual decomposed attention-based prompting for rehearsal-free continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 11909–11919

Tong K, Pan K, Zhang X, et al (2025) Any-SSR: How recursive least squares works in continual learning of large language model. In: IEEE/CVF International Conference on Computer Vision (ICCV), pp 3047–3057

Vitter JS (1985) Random sampling with a reservoir. ACM Transactions on Mathematical Software 11(1):37–57

Wang C, Li H, Zhang Y, et al (2026) ICL-Router: In-context learned model representations for LLM routing. In: AAAI Conference on Artificial Intelligence, arXiv:2510.09719

Wang H, Lu H, Yao L, et al (2025a) Self-expansion of pre-trained models with mixture of adapters for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 10087–10098

Wang L, Zhang X, Su H, et al (2024a) A comprehensive survey of continual learning: Theory, method and application. IEEE Transactions on Pattern Analysis and Machine Intelligence 46(8):5362–5383

Wang M, Adel H, Lange L, et al (2024b) Rehearsal-free modular and compositional continual learning for language models. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers). Association for Computational Linguistics, https://doi.org/10.18653/v1/2024.naacl-short.39

Wang Z, Zhang Z, Ebrahimi S, et al (2022a) DualPrompt: Complementary prompting for rehearsal-free continual learning. In: European Conference on Computer Vision (ECCV), pp 631–648

Wang Z, Zhang Z, Lee CY, et al (2022b) Learning to prompt for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 139–149

Wang Z, Li Y, Qu X, et al (2025b) SEE: Continual fine-tuning with sequential ensemble of experts. In: Findings of the Association for Computational Linguistics: ACL 2025, pp 7418–7432

Wortsman M, Ramanujan V, Liu R, et al (2020) Supermasks in superposition. In: Advances in Neural Information Processing Systems, pp 15173–15184

Wu C, Gan Y, Ge Y, et al (2024) LLaMA Pro: Progressive LLaMA with block expansion. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 6518–6537

Yan G (2026) MaRS: Memory-adaptive routing for reliable capacity expansion and knowledge retention. In: International Conference on Learning Representations (ICLR)

Yan H, Sun G, Zhou K, et al (2026) FlyPrompt: Brain-inspired random-expanded routing with temporal-ensemble experts for general continual learning. In: International Conference on Learning Representations (ICLR), arXiv:2602.01976

Yan S, Xie J, He X (2021) DER: Dynamically expandable representation for class incremental learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Yoon Y, Wang S, Chen W, et al (2026) When are experts misrouted? counterfactual routing analysis in mixture-of-experts language models. Preprint, arXiv:2605.07260

Zhou DW, Sun HL, Ye HJ, et al (2024) Expandable subspace ensemble for pre-trained model-based class-incremental learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition

Zhou DW, Cai ZW, Ye HJ, et al (2025) Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. International Journal of Computer Vision 133(3):1012–1032

Zhu H, Majzoubi M, Jain A, et al (2024) TAME: Task agnostic continual learning using multiple experts. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)

Zhuang H, Weng Z, Wei H, et al (2022) ACIL: Analytic class-incremental learning with absolute memorization and privacy protection. In: Advances in Neural Information Processing Systems