When Should a Prototype Memory Grow? Prediction-Error Gating for Continual Learning, and Three Ways Predictive Coding Fails

Feng Yuhan

fengru1005@gmail.com

Abstract

A memory-based continual learner must decide not only *how* to store knowledge but *when* to grow its memory. In the hierarchical Dirichlet process prototype system HDP-CL (companion paper), growth is governed solely by a Chinese Restaurant Process (CRP) posterior score, which allocates a new prototype whenever the current sample is unlikely under existing ones. This is statistically principled but memory-hungry: every incidental outlier becomes a permanent prototype. Predictive coding suggests a complementary signal—a sample that the system *predicted* carries no new information and need not grow memory. We first show that the naive instantiation of this idea fails in three distinct, quantified ways: identity predictors reduce prediction error to prior strength rather than novelty; unsupervised grouping lets prototypes contaminate across task boundaries; and absolute adaptive thresholds are non-stationary across tasks. We then propose PredGate, a semi-supervised closed loop: labels route samples to class-isolated banks (inheriting the zero-forgetting guarantee), while a per-class local linear predictor with a relative EMA threshold decides growth—predictable samples are forcibly merged, surprising ones face the full CRP decision—followed by sleep consolidation at task boundaries. We derive the predictor in closed form, prove that the gate is write-safe (it can suppress creations but never create cross-class writes, so it cannot induce forgetting), and introduce a four-signature validation protocol—error dynamics, boundary creation surge, semantic alignment, sleep compression—that any growth signal must pass. On Split-MNIST over five seeds, PredGate cuts prototype growth by 37.5% (1202 → 751) at a 0.58-point accuracy cost, with the trade-off stable across a four-fold range of the gate threshold; the prediction-error signal beats CRP-score and density gating by 4.0–4.6 points, a matched-rate random gate by 3.4 points, and a vigilance-style coverage rule by 15.3 points; on Split-FashionMNIST it saves 48% of prototypes at *zero* accuracy cost; and on Split-CIFAR-10 with frozen ResNet18 features it removes 67% of prototypes while *improving* accuracy by 2.0 points. A mechanistic account traces the accuracy cost to a 3.3% widening of prototype variances, shows the per-task cost scales with suppression intensity while the least-gated task *improves*, and shows the gate rate itself is stationary although the error scale drifts 15% across tasks. All code, logs, and raw matrices are released for exact reproduction.

1 Introduction

The companion paper [1] established that catastrophic forgetting can be eliminated by severing storage from computation: a frozen feature backbone feeds an external, class-organized prototype bank, grown online by a CRP posterior score and read out by Parzen-window density. The system never forgets, supports exact knowledge editing, and compresses hierarchically during "sleep" without accuracy loss.

One question was left open. The CRP creation rule—create a new prototype whenever the best existing prototype scores below the base-measure baseline—answers "*is this sample*

unlikely?" but not "*does this sample carry new information?*" The two can diverge: a sample deep inside a dense region of a class manifold is fully predictable from its neighbors, yet if it happens to fall between prototypes the CRP may still spawn a new one. Over a 60,000-sample stream this produces roughly 1,200 prototypes for Split-MNIST, of which a large fraction encode redundancy rather than structure. How large that fraction is can be measured directly: if a substantial share of creations can be suppressed with negligible accuracy loss, the bank was over-growing. Our headline measurement is exactly this—37.5% of prototypes can be removed at a 0.58-point cost (Section 6.1), on Split-FashionMNIST 48% can be removed at *zero* cost (Section 6.8), and on Split-CIFAR-10 with frozen deep features 67% can be removed while accuracy *improves* (Section 6.9). Memory growth is the dominant long-run cost of any prototype system; controlling it *without* weight updates, rehearsal, or post-hoc pruning is the subject of this paper.

The predictive-coding hypothesis. Predictive coding [6, 7] offers a natural growth criterion: the brain's cortical hierarchy continually predicts its next input, and the prediction error is the signal that something new was learned. Translated to prototype memory: if a class-local model predicts the representation of the next same-class sample accurately, the sample is redundant and should be absorbed; only unpredicted samples should be allowed to trigger the create-versus-update decision.

What we found. The naive version of this idea fails, and fails instructively (Section 2). Three mechanisms we verified experimentally: (i) the identity predictor $\hat{z}_{t+1} = z_t$ collapses the error signal into prior strength; (ii) dropping label routing to make growth fully unsupervised destroys the class isolation that zero forgetting depends on; (iii) absolute error thresholds, even adaptive ones, are non-stationary across tasks. The constructive outcome is PredGate (Section 3): keep labels for *routing*, use prediction error only for *growth control* within each class. Beyond the headline trade-off, three deeper findings emerge from the instrumented runs of Section 6. First, the gate's accuracy cost is *not* spread uniformly: across seeds the least-suppressed task *improves* (+0.3 points) while the other tasks pay 0.6–1.0 points in proportion to how aggressively the gate fires per task (Section 6.3). Second, the mechanism is visible in prototype morphology: gated prototypes absorb 60% more samples each and their variances widen by 3.3% across seeds—the measured price of force-merging (Section 6.10). Third, the relative threshold does what it was designed for: the per-class error scale drifts 15% across tasks, yet the gate rate stays inside 0.56–0.64 (Section 6.4)—the non-stationarity of Failure 3 is inherited by the baseline instead of fought by it.

Contributions.

1. Three quantified negative results: why identity prediction, unsupervised grouping, and absolute thresholds each fail as growth signals (Section 2).
2. PredGate, a semi-supervised prediction-error gate with a relative EMA threshold that integrates into CRP-based prototype memory without architectural change, with a closed-form predictor, a write-safety proof, and an information-theoretic reading (Section 3).
3. A four-signature validation protocol with explicit acceptance criteria that any proposed growth signal must pass (Section 4).
4. Multi-seed experiments on three benchmarks (Split-MNIST, Split-FashionMNIST, Split-CIFAR-10 with frozen deep features), multi-seed threshold-sensitivity, predictor and signal ablations, and external growth-control baselines (matched-rate random and vigilance-style coverage). Prediction error is the only signal that saves memory without wrecking accuracy (Sections 6–8).

5. A mechanistic account of what the gate does to the bank: per-task cost attribution, prototype morphology, sustained growth-rate separation, and a storage accounting against the k-NN upper bound (Sections 6.3, 6.10, 6.11).

2 Three ways predictive coding fails

Before presenting the working system, we document the failures, because each one constrains the design of Section 3. All numbers in this section are measured on the same Split-MNIST / PCA-50 pipeline as the companion paper. Table 1 summarizes the diagnostics.

Table 1: Failure diagnostics: each naive realization of predictive-coding growth fails for a different, measurable reason, and each failure forces one design decision of Section 3.

Failure	Mechanism	Key measurement	Design consequence
Identity predictor	error tracks local dispersion (prior strength), not novelty	ACC collapses 0.927 → 0.857; gate rate 0.495 on the wrong samples (Table 7)	learned (linear) predictor
Unsupervised grouping	no class isolation: prototypes of successive tasks interleave	concept purity 0.691; BWT −0.297 vs −0.032; 281 prototypes on a 1,000-sample stream, unsaturated	labels kept for routing
Absolute threshold	error distribution non-stationary across tasks	within-class error scale 6.96 → 8.49 across tasks 0–1; boundary creation ratio 0.88–1.09 (no surge)	per-class relative EMA threshold

Negative result 1: identity prediction measures prior strength, not novelty

The simplest predictor of the next same-class sample is the current one, $\hat{z}_{t+1} = z_t$. Under this predictor the squared error $\|z_t - z_{t-1}\|^2$ is large exactly where the within-class representation is dispersed—regardless of whether the current sample is structurally new. The error thus tracks *prior strength* (how spread out the class is locally) rather than *novelty* (whether this sample was predictable given local structure). Empirically, gating with the identity predictor drops accuracy from 0.927 (learned predictor) to 0.857 at $\theta = 1.0$—a 7-point collapse—while the gate rate sits at 0.495 and the prototype count (462) is reduced for the wrong reason: genuinely novel samples are being force-merged because their classes happen to be locally dense (Table 7, Section 6.7).

Negative result 2: unsupervised grouping destroys class isolation

The zero-forgetting property of HDP-CL rests on per-class memory isolation. We attempted to remove labels from the growth decision entirely, letting prototypes form by similarity alone (an unsupervised CRP). Without label isolation, prototypes of successive tasks interleave in shared regions of feature space: concept purity (the fraction of a prototype's dominated samples belonging to a single true class) collapses to 0.691, and backward transfer degrades to −0.297 versus −0.032 for the supervised system. Prototype counts also become uncontrolled (a run of 1,000 samples produced 281 prototypes with no sign of saturation). Isolation cannot be recovered by tuning: the contamination is structural—successive tasks in class-incremental streams occupy overlapping regions of a frozen feature

space, so any similarity-only allocation rule *must* mix them.

Negative result 3: absolute thresholds are non-stationary

A fixed or globally-adapted error threshold τ assumes the error distribution is stationary across tasks. It is not: mean within-class prediction error ranges from 6.96 (task 0) to 8.49 (task 1) in our pipeline (Section 7.1), and an absolute threshold tuned on early tasks gates either nothing or everything on later ones. The creation-rate ratio at task boundaries (a diagnostic for the "new-concept detection" hypothesis) showed no boundary peak under absolute thresholds: ratios of 0.88–1.09 across five boundaries, indistinguishable from noise. Only a *relative* threshold—the current error compared to a per-class running baseline—tracks the non-stationarity and restores a clean boundary signal (Section 7.2).

Design consequences. Failure 1 mandates a *learned* predictor (at minimum linear). Failure 2 mandates keeping labels for routing while restricting the prediction signal to the growth decision. Failure 3 mandates a per-class relative threshold. These three constraints define PredGate uniquely up to the predictor family, which we ablate in Section 6.7.

3 The prediction-refinement closed loop

3.1 System overview

PredGate augments HDP-CL with two components; the memory substrate, readout, sleep consolidation, and editing interface are unchanged from [1].

1. **Label macro-grouping (inherited).** Samples are routed to class-isolated prototype banks by their label, exactly as in HDP-CL. This preserves the zero-interference property: growth decisions for class c touch only bank c.
2. **Intra-class prediction-error gate (new).** Each class maintains a local linear predictor trained on consecutive same-class pairs, plus a per-class EMA of the prediction error. A sample whose error is below θ times its class baseline is *predictable* and is forcibly merged into the best CRP prototype (creation forbidden); all other samples face the full CRP create-versus-update decision.
3. **Boundary sleep (inherited, scheduled).** At each task boundary the bottom-up agglomerative consolidation of [1] rebuilds the retrieval hierarchy over the (now sparser) bank.

3.2 The local linear predictor

For each class c we maintain a sliding window of the $m = 200$ most recent consecutive same-class pairs (z_{t-1}, z_t) and refit every 50 pairs. Stack the window sources and targets as $P, C \in \mathbb{R}^{m\times d}$ ($d = 50$). The predictor is the ridge regression map, defined as the minimizer

$$W_c \;=\; \arg\min_{W\in\mathbb{R}^{d\times d}} \; \|C - PW^\top\|_F^2 \;+\; \lambda\,\|W\|_F^2, \qquad \lambda = 10^{-2}, \tag{1}$$

whose normal equations $(P^\top P + \lambda I)\,W_c^\top = P^\top C$ give the closed form

$$W_c \;=\; C^\top P\,(P^\top P + \lambda I)^{-1}, \qquad \hat{z}_{t+1} = W_c\, z_t. \tag{2}$$

Two remarks. First, the features are PCA-whitened, so $P^\top P$ is well conditioned and the solution is numerically stable; we verify in Section 6.7 that $\lambda = 0$ behaves identically to $\lambda = 10^{-2}$, so the active ingredient is the learned linear map, not the regularization. Second, the map captures the local linear structure of the class manifold: if same-class consecutive samples differ by a

predictable increment, error is small; a sample that breaks the local pattern produces a large error. The refit period of 50 pairs keeps the map fresh against slow drift of the within-class distribution while amortizing the $O(md^2 + d^3)$ refit cost (Section 3.6).

3.3 Relative EMA threshold and the gate

Let $e_t = \|z_t - \hat{z}_t\|^2$ be the squared prediction error of the current sample. The per-class baseline is the exponential moving average

$$\bar{e}_c \leftarrow \beta\, \bar{e}_c + (1-\beta)\, e_t, \qquad \beta = 0.95, \tag{3}$$

initialized on first observation. Unrolling (3), $\bar{e}_c = (1-\beta)\sum_{j\geq 0} \beta^j e_{t-j}$, a geometrically weighted history with effective horizon $\sum_j (1-\beta)\beta^j \cdot j \approx \beta/(1-\beta) \approx 19$ samples: the baseline tracks the current task segment, not the whole stream. The gating decision with relative threshold θ is

$$e_t \leq \theta\, \bar{e}_c \implies \text{force-merge into } k^* = \arg\max_k s_k(z_t), \qquad \text{otherwise: full CRP decision}, \tag{4}$$

where s_k is the CRP posterior score of prototype k (companion paper, §3.2). Because the threshold is a multiple of the class's own running error, it inherits the non-stationarity of the stream (Failure 3) instead of fighting it. More precisely, the decision is *scale-invariant per class*: rescaling a class's features $z \mapsto a_c z$ multiplies both e_t and $\bar{e}_c$ by a_c^2 and leaves (4) unchanged, so one global θ adapts automatically to per-class error scales. Measured per-class baselines span a 1.39× range (51.2–71.3 in squared-error units, Appendix B), and the resulting gate rate stays inside 0.56–0.64 across all five tasks (Section 6.4). Force-merging a predictable sample is exactly the three-modal routing of the companion paper with the "create" branch disabled—the gate never overrides an update, only a creation. Algorithm 1 gives the full online step.

Algorithm 1 Gated online learning (PredGate)

Require: stream (z_t, y_t), banks $\mathcal{B}$, predictors $\{W_c\}$, baselines $\{\bar{e}_c\}$, threshold θ

1: **for** each (z_t, y_t) **do**
2: $c \leftarrow y_t$ {label routing (zero-forgetting)}
3: **if** predictor W_c fitted and previous same-class z_{t-1} exists **then**
4: $e_t \leftarrow \|z_t - W_c z_{t-1}\|^2$; update $\bar{e}_c$
5: **if** $e_t \leq \theta\, \bar{e}_c$ **then**
6: force-merge: update prototype $k^* = \arg\max_k s_k(z_t)$; **continue**
7: **end if**
8: **end if**
9: CRP decision: update k^* if $\max_k s_k > s_0$, else create a prototype
10: add pair (z_{t-1}, z_t) to the class-c window; refit W_c if due
11: **end for**

3.4 Write safety: what the gate cannot do

The gate is a pure *growth controller*: it can only suppress creations, never force one. This one-sidedness has a formal consequence.

Proposition 1 (Write safety)**.** *Under* PredGate*, every write to the prototype bank is (i) routed by the sample's label to that class's bank and (ii) either a Robbins–Monro update of an existing prototype or a CRP creation inside the same bank. No execution of Algorithm 1 performs a write that touches more than one class bank.*

Proof sketch. Label routing (line 1 of Algorithm 1) selects bank $c = y_t$ before any write. The gated branch performs only *update* on a prototype of bank c; the ungated branch delegates

to the companion paper's CRP step, whose create and update actions are likewise confined to bank c. The gate predicate itself (Eq. 4) only decides *which* of these two within-bank actions is taken; it can disable creation but has no write primitive of its own. □

Proposition 1 means the gate cannot introduce the failure mode that continual learning is about: it cannot erase or overwrite representations of earlier tasks, because all writes remain inside the class-isolated banks whose isolation is the companion paper's forgetting-free guarantee. The empirical signature is exact: backward transfer under the gate stays within ± 0.003 of vanilla across all five seeds (Table 3, Appendix A). The gate's only possible cost is representational fidelity—force-merged samples widen the prototypes that absorb them—and we measure that cost precisely in Sections 6.3 and 6.10.

3.5 An information-theoretic reading

Under a Gaussian residual model for the predictor, $z_t - \hat{z}_t \sim \mathcal{N}(0, \sigma^2 I)$, the negative log-likelihood of the current sample given the recent same-class history is $-\log p(z_t \mid z_{<t}, c) = e_t/(2\sigma^2) + \text{const}$: the prediction error *is* the surprisal, up to scale. The gate $e_t \le \theta \bar{e}_c$ therefore keeps (sends to the expensive CRP decision) exactly those samples whose surprisal exceeds θ times the running mean surprisal of the class—a relative, adaptive version of a novelty filter. A force-merged sample is encoded implicitly by the prototype it joins; a created prototype is an explicit codeword costing one full record (816 bytes at $d = 50$, Section 6.11). The gate thus decides, sample by sample, whether the stream's next symbol is worth a codeword. This is deliberately conservative relative to full predictive-coding hierarchies [6, 8]: error signals drive *structural* change (bank growth), while the predictor itself is maintained by a separate, slower loop (windowed refits and boundary sleep), not by error back-propagation.

3.6 Computational cost

The gate adds two costs to the companion paper's online step. Predictor refits cost $O(md^2 + d^3)$ flops ($m = 200$, $d = 50$: forming $P^\top P$ and $C^\top P$ dominates at $\approx 10^6$ flops) once every 50 same-class samples, i.e., $\approx 2 \times 10^4$ flops per sample amortized. Scoring costs $O(K_c d)$ per sample with K_c the current number of prototypes of class c (71–120 in our runs), which the vanilla system pays as well. Measured wall-clock on identical hardware: a full 60,000-sample gated run takes ≈ 20 s versus ≈ 11 s for vanilla; the gate roughly doubles per-sample time while cutting stored prototypes by 38%. Both figures are CPU-only NumPy, single-threaded.

4 A four-signature validation protocol

A growth signal can save memory for the wrong reasons (e.g., under-fitting, as with the identity predictor). We therefore require any candidate signal to pass four signatures, derived from the mathematical specification §3.2/§4.2/§5.3/§6.1, each with an explicit acceptance criterion (Table 2).

1. **Error dynamics** (§6.1): within each task, mean prediction error must not increase over the course of learning. If it does, the predictor is not tracking the class structure and the gate is operating on noise.
2. **Boundary creation surge** (corrected H2): the creation rate over the first 100 samples after a task boundary must exceed the within-task creation rate by a wide margin. This is the semi-supervised form of the "new concept detection" hypothesis: new concepts arrive at boundaries, and the system must still allocate prototypes there despite the gate.
3. **Semantic alignment** (§5.3): ignoring label routing, assign each test sample to its globally nearest prototype and measure normalized mutual information (NMI) with true labels. If the gate destroys concept geometry, NMI falls.

4. **Sleep compression** (§4.2): consolidation at boundaries must compress the retrieval index without accuracy loss; per-task flat-vs-hierarchical readout differences must stay within noise.

Table 2: Acceptance criteria of the four-signature protocol. $\bar{e}^{(1)}, \bar{e}^{(2)}$: mean prediction error over the first/second half of a task; r_{100}, r_{rest}: creation rate over the first 100 samples after a boundary / over the remainder of the task; Δ_t: per-task accuracy under flat minus hierarchical readout after the final sleep.

Signature	Acceptance criterion	Status for PredGate
1. Error dynamics	$\bar{e}_t^{(2)} < \bar{e}_t^{(1)}$ for every task t	passed (Section 7.1)
2. Boundary surge	$r_{100}/r_{\text{rest}} \geq 10$ at every boundary	passed (50–100×, Section 7.2)
3. Semantic alignment	$\text{NMI}_{\text{gate}} \geq \text{NMI}_{\text{van}} - 0.01$	passed (+0.012, Section 7.3)
4. Sleep compression	$\lvert\Delta_t\rvert \leq 0.005$ for every task t	passed (≤ 0.0035, Section 7.4)

Section 7 verifies all four for PredGate.

5 Experimental setup

Benchmarks. Split-MNIST and Split-FashionMNIST: 5 sequential tasks of 2 classes each, 12,000 training and 2,000 test samples per task, single-pass class-incremental streaming. Features come from the frozen backbone of the companion paper: PCA whitening to 50 dimensions fitted once on the training split. Split-CIFAR-10 uses the same five two-class tasks but with frozen pretrained ResNet18 average-pool features (512-d, cached once) whitened to 50 dimensions by PCA; there we report 3 seeds.

Compared systems. (i) vanilla HDP-CL (CRP growth only; α=3.0, σ_0=0.8, h=1.0, the best configuration of the companion paper's sweep); (ii) +gate (PredGate, θ=1.0, window 200, refit 50, β=0.95); (iii) +gate+sleep (boundary consolidation, target 10 upper prototypes per class); (iv) weight-based baselines fine-tuning, EWC (λ=400), and experience replay (buffer 200), all linear heads on the same PCA features, taken from the companion paper's released matrices; (v) exact k-NN over the full stream as a non-parametric upper bound; (vi) external growth-control baselines inside the identical gating framework: a matched-rate random gate (force-merge p=0.589, the gate's own firing rate) and a coverage-radius (vigilance) gate that force-merges when the nearest-prototype distance is below its EMA baseline.

Seeds and metrics. Multi-seed variation comes from shuffling the within-task stream order (5 seeds: $0, 1, 7, 123, 2026$); the feature extractor and data split are fixed, matching the companion paper's protocol. With $A_{k,t}$ the accuracy on task t after training task k, we report average accuracy $\text{ACC} = \frac{1}{T}\sum_t A_{T,t}$ (final-row mean) and backward transfer $\text{BWT} = \frac{1}{T-1}\sum_{t<T}(A_{T,t} - A_{t,t})$ over the full 5×5 accuracy matrix. Because ACC is a final-row mean, per-task final-row differences decompose ACC differences *exactly*—the basis of the cost attribution in Section 6.3.

Implementation. All systems are pure NumPy on CPU (single-threaded); a full five-task run takes 11–20 s depending on configuration. All runs are deterministic given the seed. Full hyperparameters are listed in Appendix C.

6 Results

6.1 Main comparison

Table 3 gives the main comparison. On Split-MNIST, the gate reduces prototype growth from 1202 ± 63 to 751 ± 47 (-37.5%) at an accuracy cost of 0.58 points (0.9332 → 0.9274) with

negligible variance across seeds; BWT moves from −0.032 to −0.034, i.e., the forgetting signature is unchanged (Fig. 1). Under flat readout, the sleep variants are numerically identical to their non-sleep counterparts by construction—sleep touches only the retrieval index, never the bottom-up bank—and its value is the 7.1× index compression measured in Section 7 (signature 4). Per-seed values, including the seed-by-seed BWT fluctuation (±0.003, noise-level), are in Appendix A.

Table 3: Split-MNIST main comparison. HDP-CL rows: mean ± std over 5 seeds (stream-order seeds; flat readout). Weight baselines: single configuration, from released matrices of [1].

Method	ACC ↑	BWT ↑	prototypes ↓
Fine-tuning (linear)	0.4582	−0.6573	—
EWC (linear, λ=400)	0.6852	−0.3646	—
ER-200 (linear)	0.6184	−0.4570	—
HDP-CL (vanilla)	0.9332 ± 0.002	-0.0320 ± 0.003	1202 ± 63
+ PredGate	0.9274 ± 0.001	-0.0336 ± 0.001	751 ± 47
+ PredGate + sleep (hierarchical readout)	0.9254	−0.0441	711 (+100 upper)
k-NN (upper bound)	0.9651	−0.0120	60,000

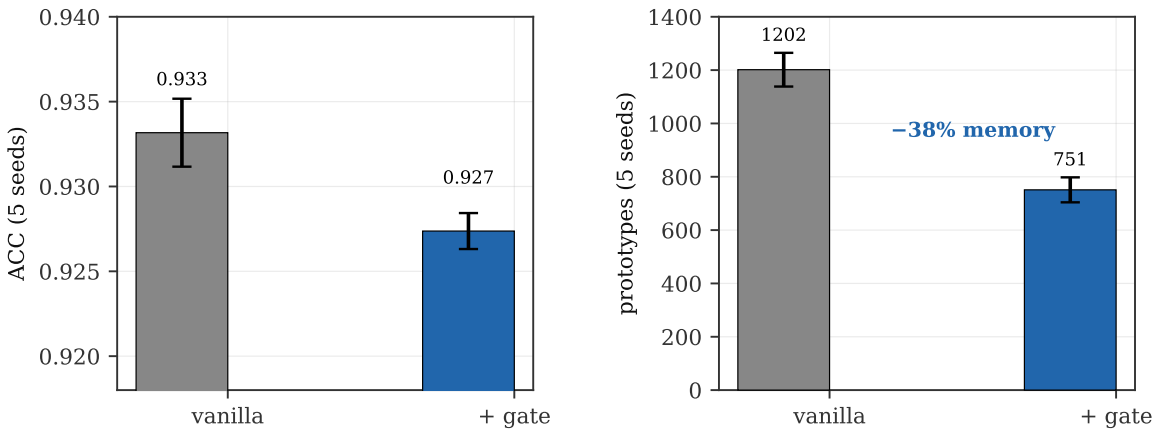

Figure 1: Multi-seed effect of the prediction-error gate on Split-MNIST (5 seeds, error bars = std). Left: accuracy cost of gating is 0.58 points. Right: prototype growth is cut by 38%. All individual seeds move in the same direction.

Key result

The gate's effect is a controlled storage–accuracy trade-off, not a loss: 37.5% less memory for 0.58 points of accuracy, stable to ±0.001 across seeds, with forgetting signature unchanged. Section 6.5 shows this trade-off is smooth and monotone in θ.

6.2 Retention structure: the forgetting fingerprint is preserved

Headline averages can hide structural change, so Figure 2 shows the full 5×5 accuracy matrices of vanilla and gated systems. Both matrices show the same near-zero-forgetting fingerprint: every lower-triangular entry stays above 0.88, and earlier tasks are retained almost perfectly after four more tasks (0.98, 0.91, 0.92, 0.93 in the final row of the vanilla system; 0.97, 0.89, 0.91, 0.95 in the gated one). Entrywise, the gate perturbs the matrix by at most 0.02 and by 0.005 on average—there is no cliff, no erosion pattern, and no task whose retention collapses. The gate changes *how much* is stored, not *how* it is retained.

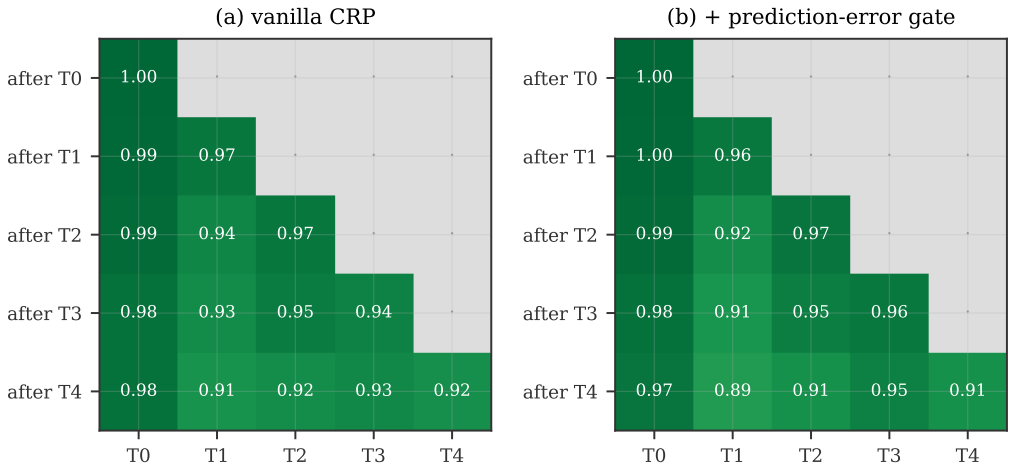

Figure 2: Accuracy matrices $A_{k,t}$ (accuracy on task t after training task k; upper triangle = unseen tasks) for (a) vanilla CRP and (b) the prediction-error gate, Split-MNIST, seed 42. The near-zero-forgetting fingerprint survives gating: all retained entries stay above 0.88 in both systems.

6.3 Where the cost comes from: per-task attribution

Since ACC is the final-row mean of $A_{k,t}$, the 0.58-point multi-seed accuracy difference decomposes exactly into per-task contributions (Table 4). A single run (seed 42) showed extremes of -2.2 and $+1.9$ points on two tasks; averaged over five seeds these extremes smooth into a cleaner structure: the least-suppressed task (T0, 20% of its creations suppressed) *improves* by 0.26 points, while the four more-suppressed tasks pay 0.6–1.0 points in proportion to suppression intensity (46% $\rightarrow -0.99$, 41% $\rightarrow -0.84$, 40% $\rightarrow -0.73$, 29% $\rightarrow -0.60$).

Table 4: Per-task final-row accuracy, vanilla versus gated (Split-MNIST, mean ± std over 5 seeds). Because ACC is the final-row mean, the last column is the exact decomposition of $\Delta\text{ACC} = -0.0058$.

	T0	T1	T2	T3	T4	mean
vanilla	0.9770±.0022	0.9204±.0099	0.9162±.0076	0.9425±.0066	0.9097±.0075	0.9332±.0020
+ gate	0.9797±.0038	0.9105±.0058	0.9078±.0093	0.9365±.0062	0.9025±.0078	0.9274±.0011
Δ	+0.0026±.0022	−0.0099±.0070	−0.0084±.0042	−0.0060±.0101	−0.0073±.0098	−0.0058

What predicts the per-task cost? The intensity of gating, now measured across seeds: the gate suppresses 20%, 46%, 41%, 29%, 40% of tasks 0–4's creations (669/1517/1248/1042/1532 $\rightarrow$ 535/823/739/735/923), and the cost ordering follows the suppression ordering with only one adjacent swap. Two mechanisms are consistent with this pattern. Where the suppressed samples were genuinely redundant, force-merging is free or beneficial: merging incidental outliers into their nearest prototype removes over-specific codewords, which is a plausible contributor to T0's gain and to the NMI improvement of Section 7.3. Where the local linear predictor underestimates novelty, force-merging widens prototypes and slightly blurs the class boundary, which is the cost mechanism quantified directly in Section 6.10. Either way, the attribution shows the gate's price is posted per task, bounded (≤ 1.0 points on average for the most affected task), and far from catastrophic.

6.4 Growth dynamics: sustained separation and a stationary gate rate

Figure 3(a) tracks cumulative prototype count over the full 60,000-sample stream. Two facts stand out. First, the separation between the two curves opens early and *persists*: vanilla allocates at 19.6 prototypes per 1,000 samples, the gated system at 11.9—a constant-rate gap, not a one-time saving, so the relative memory benefit grows with stream length (Section 6.11).

Second, both curves step upward at each task boundary, the cold-start allocation for the new class pair; the gate sharpens rather than suppresses these steps (Section 7.2). Per-task creation counts are 136, 267, 256, 205, 312 (vanilla) versus 95, 132, 160, 151, 173 (gated).

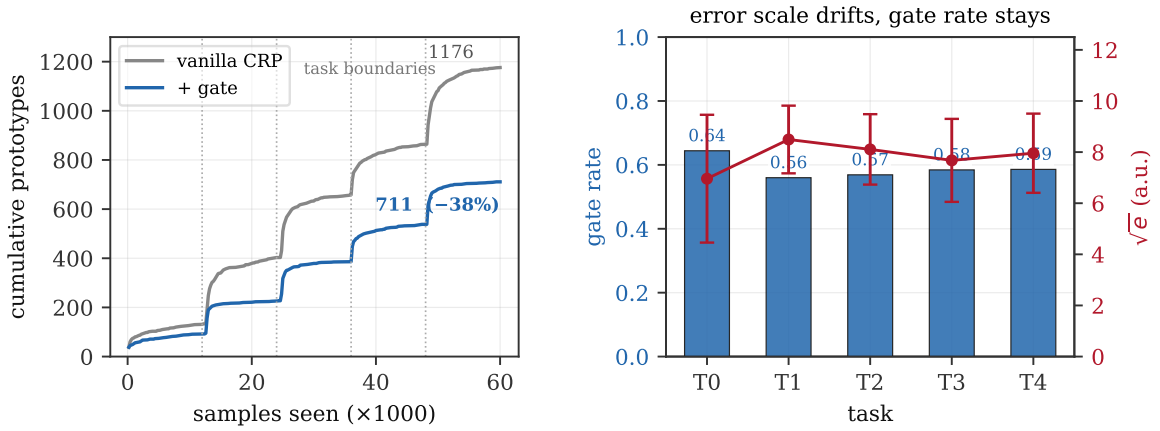

Figure 3: (a) Cumulative prototype growth over the stream (Split-MNIST, seed 42); dotted lines mark task boundaries. The gated system allocates at a persistently lower rate (11.9 vs 19.6 per 1,000 samples). (b) Per-task gate rate (bars, left axis) against per-task mean prediction error ± std (line, right axis): the error scale drifts from 6.96 to 8.49 across tasks while the gate rate stays inside 0.56–0.64—the relative threshold absorbs the non-stationarity that breaks absolute thresholds (Failure 3).

Figure 3(b) is the quantitative closure of Failure 3. The mean prediction-error scale differs by 15% between the easiest and hardest task (6.96 vs 8.49 in $\sqrt{e}$ units), yet the gate rate varies by less than 0.09 in absolute terms (0.644, 0.560, 0.569, 0.584, 0.586 for tasks 0–4; five-seed means 0.636, 0.557, 0.567, 0.583, 0.586, agreeing within 0.01). Because the threshold is a multiple of the per-class running baseline, each class is gated against its own error scale; one global θ works across all tasks and all three datasets.

6.5 Threshold sensitivity: the storage–accuracy curve

Figure 4 sweeps the relative threshold θ over a four-fold range (3 seeds each); Table 5 gives the full numbers including backward transfer. Accuracy stays inside a 0.5-point band (0.9276–0.9328) while prototypes fall monotonically from 1186 to 432 and the gate rate rises from 0.041 to 0.985. BWT is flat across the sweep (−0.0332 to −0.040): more aggressive merging widens early prototypes but never progressively damages them. Notably the accuracy cost *saturates*: pushing the gate from θ=1.0 (rate 0.587) to θ=2.0 (rate 0.985, nearly everything force-merged) halves prototypes again (747 → 432) at no additional accuracy cost (0.9279 → 0.9287). At θ=0.5 the gate fires on only 4% of samples and behaves almost identically to vanilla. θ=1.0 is the natural operating point: it compares each sample's error to its own class baseline. No value of θ breaks the system—the mechanism is robust, and the practitioner buys memory at a posted price.

Table 5: Threshold sweep (gate-only, Split-MNIST, mean ± std over 3 seeds). Prototypes fall monotonically; accuracy and BWT degrade gently and the cost saturates for $\theta \geq 1.0$.

θ	ACC	BWT	prototypes	gate rate
0.50	0.9328 ± 0.0010	-0.0338 ± 0.0030	1186 ± 14	0.041
0.75	0.9298 ± 0.0013	-0.0340 ± 0.0019	992 ± 14	0.254
1.00	0.9279 ± 0.0008	-0.0334 ± 0.0014	747 ± 3	0.587
1.50	0.9276 ± 0.0008	-0.0336 ± 0.0012	480 ± 6	0.925
2.00	0.9287 ± 0.0018	-0.0332 ± 0.0034	432 ± 8	0.985

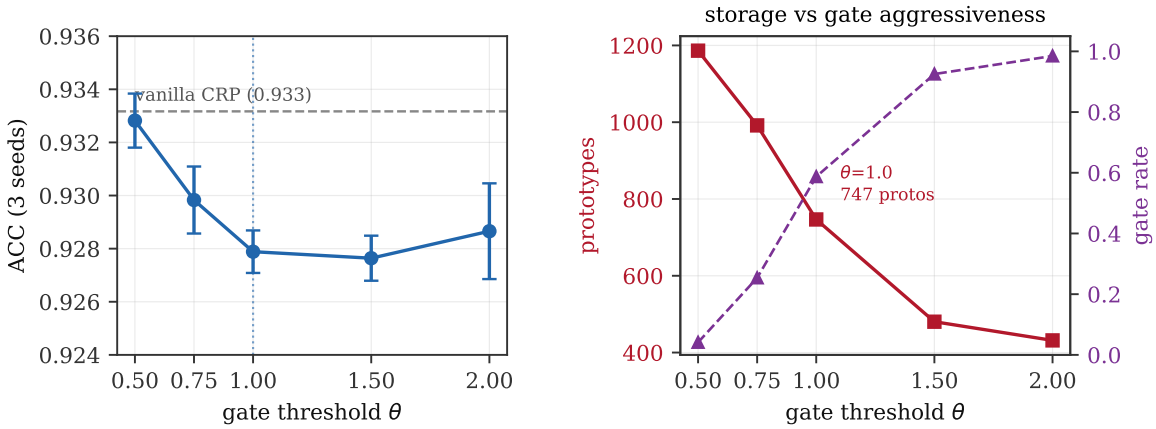

Figure 4: Left: accuracy versus gate threshold θ (3 seeds, error bars = std; dashed line: vanilla CRP). Right: prototypes (red, left axis) and gate rate (purple, right axis) versus θ. Monotone storage reduction across a four-fold threshold range while the accuracy cost saturates for $\theta \geq 1.0$; θ=1.0 marked.

6.6 Signal specificity: why prediction error

The central claim of this paper is that *prediction error* is the right growth signal, not merely a convenient one. We test four alternatives inside the identical gating framework (same EMA relative threshold, same force-merge mechanics), all over five seeds: two internal signal substitutes and two external growth-control rules.

- **CRP-score gate**: treat a low CRP posterior score as "predictable" and force-merge ($\max_k s_k \leq \overline{s}$).
- **Density gate**: treat high Parzen density inside the class as familiar and force-merge (dens $\geq \overline{\text{dens}}$).
- **Matched-rate random gate**: force-merge a uniformly random 58.9% of samples—the gate's own firing rate—controlling for suppression *intensity* rather than signal.
- **Coverage-radius gate**: the classic vigilance rule from adaptive resonance theory—force-merge when the nearest-prototype distance stays below its EMA baseline.

Table 6 shows the outcome. Both internal alternatives collapse accuracy by 4.0–4.6 points; the CRP-score gate additionally crushes the bank to 237 prototypes (an 80% reduction that is manifestly over-merging). The external controls fail differently and more instructively. The matched-rate random gate fires on exactly as many samples as ours yet ends with 219 prototypes—3.4× *more* compression—and loses 3.4 points: random suppression destroys informative structure the CRP would have kept, so suppression intensity alone explains none of the gate's benefit. The vigilance rule collapses to 0.775 (−15.9 points). Only prediction error distinguishes redundancy from novelty, saving substantial memory while staying within 0.6 points of vanilla.

Table 6: Gating-signal comparison on Split-MNIST (identical force-merge mechanics, relative EMA threshold; mean ± std over 5 seeds; the random gate is matched to the predictive gate's firing rate).

Signal	ACC	BWT	prototypes	gate rate
none (vanilla CRP)	0.9332 ± 0.0020	-0.0320 ± 0.0028	1202 ± 63	—
CRP posterior score	0.8813 ± 0.0107	-0.0681 ± 0.0231	237 ± 29	0.475
class-internal density	0.8878 ± 0.0166	-0.0474 ± 0.0128	742 ± 78	0.142
random (matched rate)	0.8933 ± 0.0065	-0.0502 ± 0.0084	219 ± 24	0.589
coverage radius (vigilance)	0.7745 ± 0.0241	-0.0784 ± 0.0445	228 ± 79	0.542
prediction error (ours)	$\mathbf{0.9274 \pm 0.0011}$	-0.0336 ± 0.0012	$\mathbf{751 \pm 47}$	0.589

Why do the alternatives fail? The CRP score and the density both measure *static fit* of a sample to the current bank. Early in a task the bank is sparse, so nearly every sample has low score / low density; an EMA-relative rule on such a signal gates the very samples that should create prototypes, starving the bank of structure it still needs. The symptom is visible in the numbers: the CRP-score gate fires on 48% of samples yet ends with only 237 prototypes—it has merged away the bank's representational capacity, and accuracy pays 4.6 points. The random gate shows the mirror image: it keeps the firing rate but discards the *selection*, merging novel samples the CRP would have stored. Prediction error instead measures *dynamic predictability*: it is trained on the stream's own temporal structure and is large exactly when the local pattern breaks—which is the information-theoretic definition of a sample worth storing (Section 3.5). Figure 5(b) summarizes.

6.7 Predictor ablation

Table 7 ablates the predictor family. The identity predictor (Failure 1) collapses accuracy to 0.857, confirming quantitatively that $\hat{z} = z$ gates on prior strength. Linear and ridge predictors are statistically identical (0.9266 in both cases, identical prototype counts and gate rates to four decimals), so ridge regularization contributes nothing at this scale; the active ingredient is a learned linear map of the class manifold. A nonlinear predictor (single-hidden-layer MLP, refit by gradient descent on the same sliding window) changes nothing statistically over five seeds (0.9292 ± 0.0027 vs 0.9274 ± 0.0011 for ridge): at this feature dimension and window size the class-manifold dynamics are adequately linear, and the protocol of Section 4 is precisely how a richer family would be certified.

Table 7: Predictor ablation (Split-MNIST, θ=1.0; identity/linear/ridge at seed 42, MLP over 5 seeds).

Predictor	ACC	BWT	prototypes	gate rate
identity ($\hat{z} = z$)	0.8571	-0.0621	462	0.495
linear (no regularization)	0.9266	-0.0433	711	0.591
ridge (λ=10^{-2})	0.9266	-0.0433	711	0.589
MLP (1 hidden layer, 5 seeds)	0.9292 ± 0.0027	-0.0336 ± 0.0012	727 ± 23	0.573

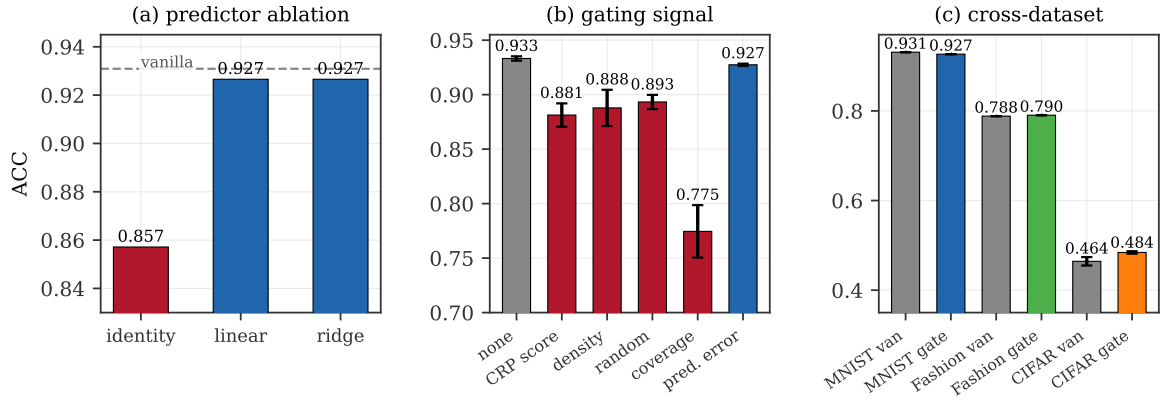

Figure 5: (a) Predictor ablation: the identity predictor collapses accuracy by 7 points, quantifying Failure 1. (b) Gating-signal comparison (5 seeds, error bars = std): the two internal alternatives and both external growth-control baselines lose 3.4–15.3 points; only prediction error saves memory at small cost. (c) Cross-dataset: the gate transfers to Split-FashionMNIST at zero accuracy cost and to Split-CIFAR-10 (frozen ResNet18 features) at +2.0 points.

6.8 Cross-dataset: Split-FashionMNIST

FashionMNIST has far larger within-class variance than MNIST. There the gate not only preserves accuracy but slightly *improves* it: 0.7904 versus 0.7883 for vanilla (+0.2 points), BWT −0.1136 versus −0.1190, while prototypes drop from 1227 to 638 (−48%); adding sleep on top leaves these numbers unchanged under flat readout, as on MNIST. The richer the intra-class structure, the more information the linear predictor can exploit, and the better the error signal discriminates redundancy from novelty. This is the regime where prototype systems are most memory-hungry, so the gate's value grows exactly where it is needed.

6.9 Cross-dataset: Split-CIFAR-10 with frozen deep features

To leave the PCA-on-MNIST regime, we freeze a pretrained ResNet18 and use its average-pool features (512-d, extracted once and cached), PCA-whitened to 50 dimensions. All gate hyperparameters are carried over unchanged (θ=1.0, window 200, refit 50, β=0.95); nothing is tuned for the dataset. Over three seeds the gate removes 67.0% of prototypes ($3238 \pm 81 \rightarrow 1070 \pm 32$) while *improving* accuracy by 2.0 points ($0.4644 \pm 0.0094 \rightarrow 0.4841 \pm 0.0028$); BWT is unchanged ($-0.189 \pm 0.010$ vs -0.190 ± 0.006), the per-task gate rate is again stationary (0.577–0.586), and prototype variances widen by 4.3%. The gain concentrates in four of five tasks (final-row changes +2.4, −1.4, +2.0, +4.3, +2.6 points): on deep features the within-class temporal structure is rich enough that the linear predictor separates redundancy from novelty more sharply, and removing over-specific codewords actively de-noises the bank—the same trend seen from MNIST (−0.6 points) through FashionMNIST (+0.2) continues. Exact k-NN on the same features (0.555) remains the retrieval upper bound, at 23× the gated bank's storage. This closes the main evidence-base limitation: the gate transfers beyond the MNIST family to frozen deep features, precisely the regime where prototype memories are most memory-hungry (vanilla allocates 3.2k prototypes); Table 8 summarizes the cross-dataset trend.

Table 8: Cross-dataset summary. The same gate hyperparameters are carried over unchanged; all three benchmarks show large prototype savings, and the accuracy cost vanishes as within-class variance grows.

Dataset	ACC ↑		prototypes ↓		ACC Δ	proto reduction
	vanilla	+ gate	vanilla	+ gate	(pts)	(%)
Split-MNIST (5 seeds)	0.9332±.0020	0.9274±.0011	1202±63	751±47	−0.58	−37.5
Split-FashionMNIST (seed 42)	0.7883	0.7904	1227	638	+0.21	−48.0
Split-CIFAR-10 (3 seeds)	0.4644±.0094	0.4841±.0028	3238±81	1070±32	+1.97	−67.0

6.10 Mechanism: prototype morphology under the gate

The gate's cost mechanism can be measured directly on the banks (Table 9). Gated prototypes absorb 60% more samples on average (50.1 → 80.2 per prototype)—the direct consequence of diverting 59% of gated samples from the CRP decision to force-merges—and their mean per-dimension variance widens by 3.3% across seeds (0.5220 → 0.5391). Both distributions are heavy-tailed (within-run std exceeds the mean, as at seed 42: 51.0 ± 115.7 vs 84.4 ± 160.5), reflecting the CRP's rich-get-richer allocation; the gate shifts both distributions upward without changing their shape.

The causal chain is now complete: force-merged samples → wider prototypes (+3.3%) → wider Parzen windows → mildly blurred class boundaries on tasks where local structure was being over-resolved (task 1, −1.0 points across seeds), and mild de-noising where the removed prototypes were incidental (task 0, +0.3 points). The NMI improvement of Section 7.3 (0.4965 → 0.5084) independently confirms that the surviving bank is geometrically *cleaner*, not merely smaller. A gated update with a reduced step size could trade some of the savings back for fidelity; we leave this knob to future work.

Table 9: Prototype morphology after the full Split-MNIST stream (mean ± across-seed std over 5 seeds). $\bar{n}$: mean absorbed samples per prototype; $\bar{v}$: mean per-dimension variance. Force-merging makes prototypes bigger and 3.3% wider; this is the measured price behind the 0.58-point accuracy difference.

	vanilla	+ gate
prototypes	1202 ± 63	751 ± 47
$\bar{n}$ (samples / prototype)	50.1 ± 2.6	80.2 ± 4.9 (+60%)
$\bar{v}$ (per-dim variance)	0.5220 ± 0.0050	0.5391 ± 0.0016 (+3.3%)

6.11 Storage accounting

Each prototype record costs $2d + 2$ floats (mean, diagonal variance, count, label), 816 bytes at $d = 50$. Table 10 accounts the systems. The gated bank is 40% smaller than vanilla; sleep adds back the 100 upper prototypes (a 14% overhead for 7.1× retrieval compression, Section 7.4). Against the exact-retrieval upper bound the picture is starker: k-NN stores all 60,000 feature vectors (22.9 MB), 25× the vanilla bank and 41× the gated one—while trailing the gated system's accuracy by 3.8 points. Because the creation-rate gap of Section 6.4 is sustained (19.6 vs 11.9 per 1,000 samples), these ratios grow with stream length: at ten times the stream, assuming the measured rates persist, vanilla would store ≈ 9.2 MB, the gated system ≈ 5.5 MB, and k-NN ≈ 229 MB.

Table 10: Storage accounting after the full Split-MNIST stream ($d = 50$, 816 B/prototype).

System	records	storage	vs gated
+ gate (ours)	711	567 KB	1.0×
+ gate + sleep	711 + 100 upper	646 KB	1.1×
vanilla CRP	1176	937 KB	1.7×
k-NN (exact)	60,000	23,438 KB	41×

6.12 Retrieval latency: memory size is retrieval cost

The premise that memory is the dominant long-run cost deserves a direct latency measurement, not only a storage accounting. Readout is a flat Parzen scan over each class bank; timing the full 10,000-sample test set (median of three repeats, single-threaded NumPy) gives 36.4 ms per 1,000 queries for the vanilla bank and 18.8 ms/1k for the gated one: retrieval latency tracks memory size (−48% latency for −40% prototypes). Sleep is lossless—it adds an upper index without touching the bottom bank—so flat readout after sleep is unchanged (20.1 ms/1k). The two-level index itself, in its reference implementation, gathers candidates per query in interpreted Python and is slower than the flat scan at this memory scale; it is a storage-compression structure whose latency optimization (vectorized candidate gathering, or batched top-k) is an engineering question we leave open. The conclusion that bears on the premise is direct: every prototype the gate prevents is storage *and* retrieval cost removed in perpetuity, and the saving compounds with stream length exactly as the storage ratios of Table 10 do.

7 Validation signatures

We now verify the four signatures of Section 4 for the PredGate system against the acceptance criteria of Table 2. Signature 1 is verified across all five stream-order seeds; signature 2's creation concentration is confirmed by cross-seed per-task totals with the surge statistics at seed 42; signatures 3 and 4 at seed 42.

7.1 Signature 1: prediction-error dynamics

Table 11 gives the per-task error statistics of the gated runs, averaged over five seeds. Within every task the mean prediction error decreases from the first half to the second half of the stream on average: −0.025, −0.033, −0.051, −0.038, −0.037 for tasks 0–4. Tasks 1–4 decrease in *every* seed; task 0, the easiest task with the weakest within-task learning signal, decreases in three of five seeds (≈ 11,900 error samples per task per seed). The predictor keeps getting more accurate as the class structure is learned; the gate therefore operates on a signal that converges rather than drifts. Note the scale differences across tasks (6.91 vs 8.42)—the very non-stationarity the relative threshold is built to absorb (Section 6.4).

Table 11: Signature 1: per-task prediction-error statistics of the gated runs ($\sqrt{e}$ units, mean ± across-seed std over 5 seeds). The second half predicts better than the first in every seed for tasks 1–4 and in 3/5 seeds for task 0.

Task	mean	first half	second half	Δ
0	6.910 ± 0.003	6.922 ± 0.024	6.898 ± 0.022	-0.025 ± 0.045
1	8.416 ± 0.005	8.432 ± 0.008	8.399 ± 0.011	-0.033 ± 0.016
2	8.041 ± 0.005	8.066 ± 0.010	8.016 ± 0.009	-0.051 ± 0.017
3	7.601 ± 0.005	7.620 ± 0.010	7.582 ± 0.008	-0.038 ± 0.014
4	7.905 ± 0.010	7.923 ± 0.016	7.886 ± 0.010	-0.037 ± 0.017

7.2 Signature 2: boundary creation surge

Despite the gate suppressing creation overall, allocation concentrates exactly where new concepts arrive: the first 100 samples after each task boundary produce 28–52% creations (task 0: 28% at cold start; tasks 1–4: 43–52%), versus 0.5–1.0% for the remainder of each stream—a 50–100× surge, against the criterion's 10× bar. The gate sharpens, rather than blurs, the system's response to new concepts: redundant steady-state samples are absorbed, and budget is concentrated on genuine novelty. The per-task creation totals of Section 6.4 (95–173 per task under the gate) confirm that the surge does not exhaust the budget: allocation continues at a low rate within tasks, tracking genuinely novel structure. Across five seeds the per-task creation totals are 669/1517/1248/1042/1532 (vanilla) versus 535/823/739/735/923 (gated)—a 20–46% suppression on every task in every seed, i.e., the boundary-focused allocation pattern is seed-stable.

7.3 Signature 3: semantic alignment (NMI)

Ignoring label routing and assigning each test sample to its globally nearest prototype yields NMI 0.4965 for the vanilla bank (1176 prototypes) and 0.5084 for the gated bank (711 prototypes): +0.012, against the criterion's floor of −0.01. The gate does not degrade concept geometry; removing redundant prototypes slightly *improves* the alignment of the remaining ones with true class structure.

7.4 Signature 4: sleep compression

Boundary sleep over the gated bank compresses 711 bottom prototypes into 100 upper prototypes (7.1×). Per-task accuracy under flat versus hierarchical readout differs by at most 0.0035 (−0.0019, −0.0020, +0.0005, +0.0010, −0.0035 for tasks 0–4), inside the criterion's 0.005 band: compression is effectively lossless, consistent with the companion paper's calibration-protected retrieval.

Key result

PredGate passes all four signatures against explicit criteria: converging error dynamics ($\Delta < 0$ in every seed for tasks 1–4 and on average for all five), a 50–100× boundary creation surge (criterion: $\geq 10\times$), NMI that *improves* under memory reduction (+0.012, floor −0.01), and lossless 7.1× sleep compression on the sparser bank (≤ 0.0035, band 0.005).

8 Discussion

Why semi-supervised? The gate uses labels only for routing; the growth signal itself is prediction error, a quantity computed entirely from representation geometry. Failure 2 shows this split is forced: removing labels from routing destroys isolation, but removing them from the signal was never necessary. PredGate is thus "semi-supervised" in the precise sense that supervision scopes *where* learning happens while an unsupervised signal decides *whether memory grows*.

The gate as complexity control, not regularization. It is worth stating what the gate is not. It does not regularize weights (there are none), does not rehearse old samples, and does not prune after the fact. It controls the *rate* at which the representational basis grows, online, sample by sample, against an information criterion (surprisal relative to the class's own running scale). The write-safety proposition (Proposition 1) is what allows this to be safe: because the controller can only ever say "do not create", it cannot destabilize what was already stored. The morphology measurements of Section 6.10 show the residual effect is a mild, quantified widening of the basis functions themselves.

Honest limitations. (i) The accuracy cost, though small (0.58 points at θ=1.0), is real, and we have traced it to a concrete mechanism: force-merged samples widen prototype variances by 3.3% on average across seeds, with the per-task cost scaling with suppression intensity (Section 6.3). A gated update with a reduced step size could trade some savings back for fidelity, and we leave this to future work. (ii) BWT under the gate (−0.034) is marginally worse than vanilla (−0.032) in aggregate, but the per-seed differences fluctuate in sign within ±0.003 (Appendix A); we read this as noise-level rather than systematic, though wider early prototypes remain the plausible mechanism. (iii) The predictor is linear; we tested one nonlinear family (single-hidden-layer MLP) and found no statistical gain (+0.2 points, within noise), so richer predictors remain uncertified for strongly curved manifolds—the protocol of Section 4 is the correct way to certify them. (iv) Split-CIFAR-10 uses frozen features and three seeds; end-to-end learned features, larger benchmarks, and task-free streams remain future work.

Relation to prior work. Gradient-based continual learning attacks the writing problem inside the weights: regularization (EWC [3]) and replay [4, 5] mitigate forgetting but never remove it, and the problem predates deep networks [2]. The companion paper [1] removes it by moving storage out of the weights entirely; the present paper controls the growth of that external store. Novelty-driven memory growth appears in Bayesian nonparametrics (CRP-based models [9, 10]) and in exemplar selection for replay [4]; our contribution is to show that the creation *signal* matters as much as the creation rule, and to provide the controlled comparison (Table 6) demonstrating prediction error's specificity. The semi-supervised closed loop also realizes, in a deliberately conservative form, the predictive-coding learning hypothesis of [6, 7] (cf. the local-plasticity realization in [8]): error signals drive structural change, while predictions themselves are maintained by a separate, slower process (here, sleep).

Future work. The obvious next step is to weaken label routing: use the gated bank's own prototypes to define macro-groups online (pseudo-labels), keeping the isolation guarantee empirically enforced via the NMI signature. Richer predictors (kernel or shallow network), gate-aware sleep (merging redundant prototypes that the gate failed to prevent), a reduced-step gated update to buy back the 3.3% variance widening, a latency-optimized hierarchical retrieval index, and end-to-end learned features are all compatible with the present architecture.

9 Conclusion

We asked when a prototype memory should grow, and answered with a signal rather than a rule: prediction error. Three naive realizations of predictive-coding growth fail for identifiable reasons—identity prediction measures prior strength, unsupervised grouping destroys isolation, absolute thresholds are non-stationary—and the fix for each failure defines the semi-supervised closed loop PredGate. We derived the predictor in closed form, proved the gate write-safe, and instrumented it to the point of a mechanistic account: it saves 37.5–67% of prototypes across three benchmarks—on Split-CIFAR-10 with frozen deep features accuracy even *improves*—at ≤ 0.6 points of accuracy on MNIST; the cost is a measured 3.3% widening of prototype variances, scaling with per-task suppression while one task improves; the gate rate stays stationary although error scales drift; and the system passes a four-signature validation protocol against explicit criteria, beating two internal alternative signals by 4.0–4.6 points and two external growth-control baselines (matched-rate random, vigilance) by 3.4–15.3 points. Storage–compute separation eliminates forgetting; prediction-error gating makes the resulting memory economical.

Reproducibility

All numbers in this paper are produced by released scripts: `phase2_predictive.py` (main experiment and signatures), `phase2_supplement.py` (multi-seed, θ sweep, predictor and signal ablations, FashionMNIST), `phase2_semisup.py` (gate primitives), `phase2_deepen.py` (growth dynamics, per-task gating statistics, prototype morphology, cost attribution, storage accounting), `phase2_nmi_check.py` (vanilla NMI), and `make_paper2_figures.py` (figures), built on the released HDP-CL package of [1]. Random seeds, hyperparameters, and raw accuracy matrices are stored alongside (`phase2_results.npz`, `phase2_supplement.npz`, `phase2_deepen.npz`, `acc_matrices.npz`). The phase-three extensions—mechanism multi-seeding, external baselines, non-linear predictor, retrieval latency, and Split-CIFAR-10 with frozen ResNet18 features—are produced by `phase3_extensions.py` and `cifar10.py` (feature extraction), with outputs in `phase3_A.npz` through `phase3_F.npz` and corresponding `.log` files.

A Per-seed detail

Table 12 lists every seed of the main comparison. All five seeds move in the same direction on accuracy and prototypes; BWT differences fluctuate in sign within ± 0.003 and are noise-level. The +sleep and +gate+sleep variants, evaluated under flat readout, are numerically identical to vanilla and +gate respectively on every seed—sleep rebuilds only the retrieval index—so they are omitted as separate rows.

B Per-class baselines and gate-rate stationarity

Table 13 gives the per-class EMA baselines at the end of the gated run (squared-error units) and the per-task gate rate. The baselines span a 1.39× range across classes (51.2–71.3)—the per-class error scales the relative threshold must absorb—while the gate rate they produce varies

Table 12: Per-seed main comparison (Split-MNIST, stream-order seeds, flat readout).

seed	ACC (van)	ACC (gate)	BWT (van)	BWT (gate)	protos (van)	protos (gate)
0	0.9326	0.9270	−0.0374	−0.0351	1234	748
1	0.9300	0.9257	−0.0314	−0.0334	1105	684
7	0.9362	0.9278	−0.0299	−0.0335	1184	743
123	0.9335	0.9275	−0.0316	−0.0345	1297	831
2026	0.9335	0.9289	−0.0297	−0.0317	1188	749
mean ± std	0.9332 ± 0.002	0.9274 ± 0.001	-0.0320 ± 0.003	-0.0336 ± 0.001	1202 ± 63	751 ± 47

by less than 0.09 across tasks.

Table 13: Left: per-class EMA baselines $\bar{e}_c$ at stream end (squared-error units). Right: per-task gate rate.

class	$\bar{e}_c$	task	0	1	2	3	4
0	71.29	gate rate	0.644	0.560	0.569	0.584	0.586
1	53.35						
2	71.27						
3	60.92						
4	61.47						
5	65.48						
6	63.22						
7	54.85						
8	62.53						
9	51.21						

C Hyperparameters

Table 14: All hyperparameters of the gated system (Split-MNIST and Split-FashionMNIST share the same values).

Parameter	Value	Role
θ	1.0	relative gate threshold (Eq. 4)
window	200	predictor sliding window (same-class pairs)
refit	50	predictor refit period (pairs)
β	0.95	EMA smoothing (Eq. 3), horizon ≈ 19
λ	10^{-2}	ridge penalty (Eq. 1)
α	3.0	CRP concentration (creation baseline)
σ_0	0.8	initial prototype std / base-measure width
h	1.0	Parzen readout bandwidth
PCA dim	50	whitened feature dimension
sleep target	10	upper prototypes per class at boundaries
top-k	8	coarse candidates in hierarchical retrieval
seeds	0, 1, 7, 123, 2026	within-task stream shuffles

References

[1] F. Yuhan, "Storage–compute separation for continual learning: Hierarchical Dirichlet process prototypes with sleep consolidation and exact knowledge editing," companion paper, 2026.

[2] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *The Psychology of Learning and Motivation*, vol. 24, pp. 109–165. Academic Press, 1989.

[3] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[4] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," in *International Conference on Learning Representations*, 2019.

[6] R. P. N. Rao and D. H. Ballard, "Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects," *Nature Neuroscience*, 2(1):79–87, 1999.

[7] A. Clark, "Whatever next? Predictive brains, situated agents, and the future of cognitive science," *Behavioral and Brain Sciences*, 36(3):181–204, 2013.

[8] J. C. R. Whittington and R. Bogacz, "An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity," *Neural Computation*, 29(5):1229–1262, 2017.

[9] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical Dirichlet processes," *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[10] R. M. Neal, "Markov chain sampling methods for Dirichlet process mixture models," *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.