ENGRAM: ASSOCIATIVE RECALL FOR PROLONGED TEST-TIME ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time adaptation (TTA) must cope with conditions that disappear and return. Updating a single model can overwrite useful adaptations, while test-time mixtures of experts maintain separate parameter states and require input-dependent routing. A visual descriptor of each test batch retrieves a linear combination of shared parameter-adjustment directions. After adaptation, an error-correcting write updates memory. A compensating residual retains adjustments not reproduced by the updated memory read, preserving the adapted parameters exactly. Our analysis examines how descriptor similarity and limited memory capacity affect recall. The experiments span image corruptions (ImageNet-C and CIFAR-10/100-C), object-level style shifts (DomainNet-126 and PACS), and semantic segmentation (Cityscapes→ACDC), under recurring and evolving conditions. In recurring corruption tests, ENGRAM improves on expert-based baselines using the same TTA host in 26 of 30 comparisons and ties the remainder; it attains the lowest final error among compared methods in all six recurring corruption settings. With LoRA-based adaptation, it improves ACDC segmentation by 2.7 mIoU points over the corresponding expert-based baseline at comparable computation and peak GPU memory. Long-horizon tests show sustained recall, and held-out evaluations demonstrate transfer to unseen corruptions. Ablation studies show how the different components help preserve useful adaptations and recall them when conditions return.

1 INTRODUCTION

Deployment conditions do not simply move away from the training distribution: they also return. A vehicle encounters fog on successive journeys, and a perception system alternates between daylight and night. Test-time adaptation (TTA) updates a source-trained model from unlabelled test batches (Wang et al., 2021; Niu et al., 2022). Over a prolonged stream, it must both adapt to the current condition and retain adjustments that may be useful again. Stability alone does not solve this retention problem. Regularisation, restoration, and weight ensembling can limit error accumulation and collapse (Niu et al., 2022; Wang et al., 2022; Marsden et al., 2024; Press et al., 2023; Hoang et al., 2024), but do not explicitly retrieve a previously adapted state. Test-time mixtures of experts maintain specialised parameter states and use input-dependent routing to select or combine them (Lee et al., 2024; Vray et al., 2025). Related conditions may require adjustments along common parameter directions, yet storing separate expert states does not explicitly capture this shared structure. Can one working model recall condition-dependent adaptations as combinations of learned, shared adjustment directions?

ENGRAM (Figure 1) learns a memory of parameter adjustments alongside an existing TTA method, the *host*. A test-batch style descriptor queries a linear map to retrieve a weighted combination of shared adaptation directions. Before the host updates, the *working model*, the single model that the host adapts, replaces the adjustment recalled for the previous descriptor with the one recalled for the current descriptor; the rest of its recent adaptation travels with it as a *working residual*. After the update, an error-correcting write stores part of the completed adjustment and the residual is recomputed to hold the remainder, so the adapted parameters are unchanged. An append-only basis adds shared directions up to a fixed rank budget, without domain labels or a separate parameter state for each condition.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

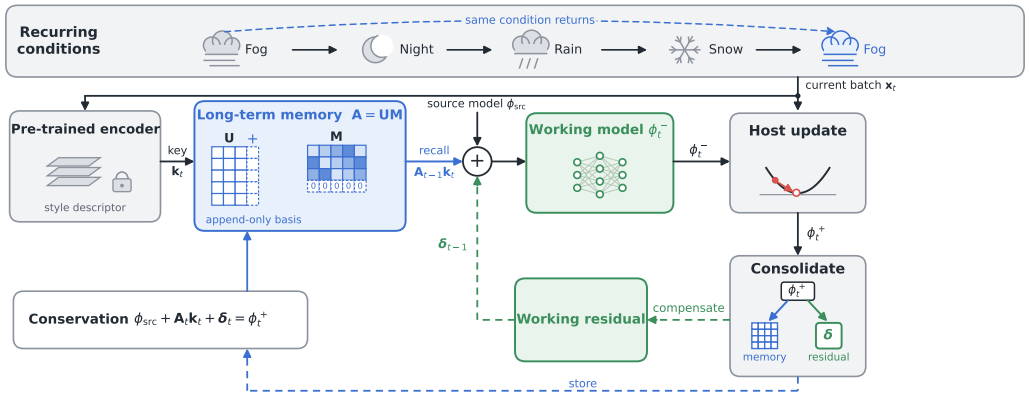


Figure 1: **ENGRAM.** A fixed pre-trained encoder supplies the memory key. The memory read and working residual initialise the host model. After adaptation, consolidation updates memory and recomputes the residual while preserving the adapted parameters.

Prior work constrains adaptation gradients to a low-dimensional subspace (Duan et al., 2025) or stores spectral coefficients for later retrieval (Baek et al., 2026). ENGRAM learns shared directions from the host's completed parameter adjustments and uses visual style descriptors to retrieve combinations of these directions. The memory write follows the established delta rule (Widrow & Hoff, 1960; Schlag et al., 2021). Our contribution is its coupling to the host through residual compensation that preserves the adapted parameters during memory writes, together with basis growth that leaves existing memory reads unchanged.

Contributions. (1) Recall without domain experts. ENGRAM gives one working model an associative memory of its own adaptations. A visual style descriptor of each test batch retrieves a combination of shared adjustment directions, learned online from the host's completed updates, so a returning condition is recalled without domain labels, discrete expert routing, or per-condition parameter copies, and the host objective is untouched. **(2) Memory updates that preserve adaptation.** A compensating residual preserves the adapted parameters during every memory update. With a fixed descriptor and matched conditions, ENGRAM follows the underlying TTA trajectory. We quantify how descriptor overlap affects interference and bound the parameter error when conditions return, accounting for retrieval error, incomplete consolidation, and limited capacity. **(3) Durable, transferable gains.** ENGRAM improves on matched expert-based baselines in 26 of 30 recurring corruption comparisons and ties the rest, with gains also in style adaptation and segmentation. Long-horizon tests show sustained recall over 100 recurrences; held-out tests show transfer to unseen corruptions.

Related work. Appendix F reviews continual TTA, approaches for reusable adaptation, and associative memory, with a detailed discussion of how ENGRAM differs.

2 METHOD

Host interface. A source-trained model receives unlabelled batches $\mathbf{x}_1, \mathbf{x}_2, \ldots$ without domain identities or boundaries. Its test-time adaptation method, the *host*, updates a vector of coordinates $\phi \in \mathbb{R}^d$ whose source value is ϕ_{src}; these may be normalisation parameters, prompts, or adapter parameters. Its complete batch transition is

$$(\phi_t^+, \mathcal{S}_t, \widehat{\mathbf{p}}_t) = \mathcal{H}(\phi_t^-, \mathcal{S}_{t-1}; \mathbf{x}_t), \tag{1}$$

where ϕ_t^- and ϕ_t^+ denote the parameters before and after the host update, and $\widehat{\mathbf{p}}_t$ denotes its predictions. The auxiliary state $\mathcal{S}_t$ includes optimiser statistics, teacher parameters (Wang et al., 2022), or buffered samples (Yuan et al., 2023), depending on the host. ENGRAM supplies the input parameters and records the completed update. The host follows its original adaptation procedure, and predicts using its original model and timing. Auxiliary states evolve under the host's rules; ENGRAM does not store or restore them separately for each domain.

Algorithm 1 ENGRAM: online cycle

Require: Source ϕ_{src}; host $\mathcal{H}, \mathcal{S}_0$; unit-key map g; memory settings

Initialise empty $\mathbf{U}, \mathbf{M}$, zero $\boldsymbol{\delta}$, and growth statistics
for each unlabelled batch $\mathbf{x}_t$ **do**
  Read $\mathbf{k}_t \leftarrow g(\mathbf{x}_t)$
  $\phi_t^- \leftarrow \phi_{\text{src}} + \mathbf{U}\mathbf{M}\mathbf{k}_t + \boldsymbol{\delta}$
  Adapt run the complete host step:
  $(\phi_t^+, \mathcal{S}_t, \widehat{\mathbf{p}}_t) \leftarrow \mathcal{H}(\phi_t^-, \mathcal{S}_{t-1}; \mathbf{x}_t)$
  Emit $\widehat{\mathbf{p}}_t$; $\mathbf{y}_t \leftarrow \phi_t^+ - \phi_{\text{src}}$
  Write $\mathbf{e}_t \leftarrow \mathbf{U}^\top \mathbf{y}_t - \mathbf{M}\mathbf{k}_t$
  $\mathbf{M} \leftarrow \mathbf{M} + \beta \mathbf{e}_t \mathbf{k}_t^\top$
  Compensate $\boldsymbol{\delta} \leftarrow \mathbf{y}_t - \mathbf{U}\mathbf{M}\mathbf{k}_t$
  Grow update statistics; append if triggered (Appendix A)
end for

What each phase changes

Read	continues the last adapted model, applying only the change in recalled adjustment (Proposition 1).
Adapt	the host adapts as usual, with its own objective.
Write	memory absorbs the adjustment; the model is unchanged (Proposition 1).
Grow	adds a direction; existing reads are unchanged (Equation (4)).

Cost. One encoder pass, $\mathcal{O}(r(d+m)+d)$ arithmetic per batch, $r(d+m)+2d+m$ stored scalars (Appendix C).

Shared memory. We compute a unit-norm batch key $\mathbf{k}_t \in \mathbb{R}^m$ from channel-wise log-variances of features from the early layers of a fixed pre-trained VGG-19 encoder (Simonyan & Zisserman, 2014). Standardisation uses source images allowed by the reference protocol (Vray et al., 2025). Appendix C specifies the layers and key construction. Memory factorises as $\mathbf{A} = \mathbf{U}\mathbf{M}$, where $\mathbf{U} \in \mathbb{R}^{d\times r}$ contains orthonormal adjustment directions and $\mathbf{M} \in \mathbb{R}^{r\times m}$ maps keys to their coefficients. The read $\mathbf{A}\mathbf{k} = \mathbf{U}(\mathbf{M}\mathbf{k})$ combines these directions. The working residual $\boldsymbol{\delta} \in \mathbb{R}^d$ stores the difference between the completed adjustment and the memory read. Keys may vary within domains and overlap across them; no domain labels or domain count are required.

Read and adapt. Starting with empty memory and zero residual, the first batch uses the source parameters. Subsequent initialisations combine the current read with the previous residual:

$$\begin{aligned} \phi_t^- &= \phi_{\text{src}} + \mathbf{A}_{t-1}\mathbf{k}_t + \boldsymbol{\delta}_{t-1}, \\ &= \phi_{t-1}^+ + \mathbf{A}_{t-1}(\mathbf{k}_t - \mathbf{k}_{t-1}) \qquad (t \geq 2). \end{aligned} \tag{2}$$

Thus recall applies a *difference of reads* to the last adapted model, rather than replacing that model with memory alone. The correction vanishes for a fixed key. The host then executes Equation (1).

Write and compensate. Let $\mathbf{y}_t = \phi_t^+ - \phi_{\text{src}}$ be the completed adjustment and $\mathbf{e}_t = \mathbf{U}^\top \mathbf{y}_t - \mathbf{M}\mathbf{k}_t$ its coefficient error. A delta-rule write with rate $0 < \beta \leq 1$ is followed by exact residual recomputation:

$$\mathbf{M} \leftarrow \mathbf{M} + \beta \mathbf{e}_t \mathbf{k}_t^\top, \qquad \boldsymbol{\delta}_t = \mathbf{y}_t - \mathbf{U}\mathbf{M}\mathbf{k}_t. \tag{3}$$

The target includes all operations completed by the host. The write changes only what memory will recall next; the model that just adapted is untouched, and the residual has no objective or learning rate of its own. Memory is never decayed globally.

Append-only basis growth. After each write, we maintain a running average of the part of the residual that the basis cannot represent (Algorithm 2). After the required number of batches, we add a direction if capacity remains and the average's bias-corrected norm exceeds a fixed fraction of $\|\mathbf{y}_t\|$. We re-orthogonalise and normalise this candidate to obtain the new direction $\mathbf{u}$:

$$\mathbf{U} \leftarrow [\mathbf{U}\ \mathbf{u}], \qquad \mathbf{M} \leftarrow \begin{bmatrix} \mathbf{M} \\ \mathbf{0}_{1\times m} \end{bmatrix}. \tag{4}$$

The zero row leaves all existing reads unchanged; later writes learn coefficients for the new direction. Directions are never removed. When the basis is full, adjustments it cannot represent remain in the residual and cannot be recalled from memory. Appendix A and Table 10 give the full growth rule and default settings.

3 GUARANTEES AND LIMITS OF RECALL

The read–adapt–write cycle satisfies exact identities that hold for any host and any write rate. We state these first, then the additional assumptions needed to analyse repeated visits. After batch t, let

$\mathbf{A}_t = \mathbf{U}_t\mathbf{M}_t$ and $\mathbf{P}_t = \mathbf{U}_t\mathbf{U}_t^\top$ be the memory and its basis projector, and let $\mathbf{v}_t = \mathbf{y}_t - \mathbf{A}_{t-1}\mathbf{k}_t$ be the pre-write error. Proofs are in Appendix B.

Proposition 1 (conservation and host equivalence). *For every batch, $\boldsymbol{\phi}_{\mathrm{src}} + \mathbf{A}_t\mathbf{k}_t + \boldsymbol{\delta}_t = \boldsymbol{\phi}_t^+$, and $\|\boldsymbol{\phi}_{t+1}^- - \boldsymbol{\phi}_t^+\| \le \|\mathbf{A}_t\|_2\|\mathbf{k}_{t+1} - \mathbf{k}_t\|$. While the key stays constant,* ENGRAM *and the host produce the same parameter trajectory from the same starting state and inputs, including any randomness the host uses, for any host, any $\beta \in (0, 1]$, and across growth events.*

A write preserves the adapted parameters through residual compensation, so memory can be updated after every batch without undoing the host's adaptation. Its auxiliary state $\mathcal{S}_t$ follows the host's update rules; it is not restored from an earlier visit to the condition. The bound controls sensitivity to descriptor variation, which can occur even within one condition: the next initialisation differs from the last adapted parameters by at most the norm of the key change multiplied by $\|\mathbf{A}_t\|_2$.

Proposition 2 (consolidation is a filter). *With a fixed key $\mathbf{k}$ and basis, the stored read $\mathbf{q}_t = \mathbf{A}_t\mathbf{k}$ obeys*

$$\mathbf{q}_{t+1} = (1 - \beta)\mathbf{q}_t + \beta\mathbf{P}\mathbf{y}_{t+1}, \tag{5}$$

and if $\mathbf{y}_t \to \mathbf{y}^*$ then $\mathbf{q}_t \to \mathbf{P}\mathbf{y}^*$ and $\boldsymbol{\delta}_t \to (\mathbf{I} - \mathbf{P})\mathbf{y}^*$.

Memory exponentially averages completed host adjustments within the basis. For a fixed key and basis, the write rate controls consolidation without changing the matched host trajectory. If the host converges, the read approaches its projected adjustment without additional shrinkage; components outside the basis remain in the residual.

Proposition 3 (interference is weighted by overlap). *For a fixed query $\mathbf{k}_\star$ and batches $s < T$,*

$$(\mathbf{A}_T - \mathbf{A}_s)\mathbf{k}_\star = \beta \sum_{t=s+1}^{T} \langle \mathbf{k}_t, \mathbf{k}_\star \rangle \, \mathbf{P}_{t-1}\mathbf{v}_t. \tag{6}$$

A query orthogonal to every intervening key is preserved exactly, including across growth; under a global decay γ it would instead shrink to $\gamma^{T-s}\mathbf{A}_s\mathbf{k}_\star$.

A write changes another read in proportion to how much the two keys overlap and to the write error projected onto the basis, where the projector is the one in place before growth because the write precedes expansion. Overlapping reads can therefore change while consolidation is incomplete, even for a stationary host target. Omitting global decay removes an additional source of forgetting, but does not eliminate this interference.

Theorem 1 (what determines re-entry). *Let a batch with key $\mathbf{k}_i$ follow a batch with key $\mathbf{k}_j$ and completed adjustment $\mathbf{y}_j$, let $\mathbf{A}, \mathbf{P}$ be the current memory and projector, $\mathbf{P}^\perp = \mathbf{I} - \mathbf{P}$, and let $\mathbf{y}_i^*$ be a reference adjustment for the returning condition. Under full carry,*

$$\|\boldsymbol{\phi}_i^- - \boldsymbol{\phi}_{\mathrm{src}} - \mathbf{y}_i^*\| \le \underbrace{\|\mathbf{A}\mathbf{k}_i - \mathbf{P}\mathbf{y}_i^*\|}_{\text{retrieval error}} + \underbrace{\|\mathbf{P}\mathbf{y}_j - \mathbf{A}\mathbf{k}_j\|}_{\text{consolidation lag}} + \underbrace{\|\mathbf{P}^\perp(\mathbf{y}_j - \mathbf{y}_i^*)\|}_{\text{unrepresented mismatch}}. \tag{7}$$

Small retrieval error, consolidation lag, and unrepresented mismatch together ensure accurate parameter initialisation. The retrieval and consolidation error vectors can cancel, so neither must be small individually. The bound concerns parameters, not prediction error. Appendix B separates the last term into reference mismatch and carried adaptation error and covers gated carry.

Beyond identities. Proposition 4 idealises the recurring stream. It assumes stationary, linearly independent unit keys, a fixed basis, empty initial memory, and that each visit contracts the error on its own key towards a fixed target through a fixed linear map. Without disturbances, independent uniform visits give mean-square convergence, with a rate bound involving the smallest eigenvalue of the key Gram matrix. Under a fixed cyclic order, the memory-map error contracts over each complete cycle. Bounded disturbances yield corresponding error bounds. Proposition 5 gives the smallest representation error attainable with rank at most r. This error has two parts: key–target relationships that no linear map can satisfy, and insufficient rank. Independent keys remove the first limitation but not the second, while poorly conditioned keys can require a large-norm map. Neither result guarantees convergence of an arbitrary host or better predictions; Section 4.2 examines these mechanisms empirically.

Table 1: **Recurring CSC and CDC.** Classification error (%, ↓) at recurrences 1 and 20; severity 5, five-seed means. $\Delta = e_{20} - e_1$: green indicates improvement, red degradation. Bold marks the best final error per dataset and schedule, including ties.

	Recurring CSC									Recurring CDC								
	CIFAR-10-C			CIFAR-100-C			ImageNet-C			CIFAR-10-C			CIFAR-100-C			ImageNet-C		
Method	1	20	Δ	1	20	Δ	1	20	Δ	1	20	Δ	1	20	Δ	1	20	Δ
Source	43.5	43.5	0.0	46.5	46.5	0.0	82.0	82.0	0.0	43.5	43.5	0.0	46.5	46.5	0.0	82.0	82.0	0.0
TENT (Wang et al., 2021)	19.3	87.8	+68.5	61.4	99.0	+37.6	62.6	99.5	+36.9	20.5	87.0	+66.5	60.2	98.9	+38.7	62.0	99.5	+37.5
+ReservoirTTA (Vray et al., 2025)	18.3	17.6	−0.7	38.1	44.0	+5.9	62.6	58.2	−4.4	18.2	17.4	−0.8	33.9	39.7	+5.8	62.4	57.5	−4.9
+ENGRAM	19.3	17.1	−2.2	61.4	44.0	−17.4	62.6	56.7	−5.9	20.5	16.9	−3.6	60.2	39.7	−20.5	62.0	56.0	−6.0
ETA (Niu et al., 2022)	17.8	30.9	+13.1	32.0	37.6	+5.6	60.0	59.4	−0.6	17.9	33.5	+15.6	32.4	37.6	+5.2	59.3	60.1	+0.8
+ReservoirTTA (Vray et al., 2025)	17.5	16.4	−1.1	31.6	30.0	−1.6	59.8	53.1	−6.7	17.4	16.3	−1.1	30.9	29.6	−1.3	58.6	52.2	−6.4
+ENGRAM	17.8	**15.9**	−1.9	32.0	29.0	−3.0	60.0	51.6	−8.4	17.9	**15.8**	−2.1	32.4	28.6	−3.8	59.3	50.7	−8.6
SAR (Niu et al., 2023)	20.4	20.4	0.0	31.9	60.4	+28.5	61.9	67.1	+5.2	20.4	20.4	0.0	31.6	57.8	+26.2	61.5	66.2	+4.7
+ReservoirTTA (Vray et al., 2025)	20.4	20.4	0.0	31.9	30.5	−1.4	62.2	53.1	−9.1	20.4	20.4	0.0	31.7	29.8	−1.9	62.6	53.6	−9.0
+ENGRAM	20.4	20.4	0.0	31.9	29.5	−2.4	61.9	51.6	−10.3	20.4	20.4	0.0	31.6	28.8	−2.8	61.5	52.1	−9.4
CoTTA* (Wang et al., 2022)	18.8	22.4	+3.6	35.1	65.5	+30.4	67.6	61.7	−5.9	18.8	22.3	+3.5	35.1	65.1	+30.0	67.7	61.5	−6.2
RoTTA (Yuan et al., 2023)	19.4	18.4	−1.0	34.8	59.1	+24.3	67.3	99.4	+32.1	21.9	20.4	−1.5	36.8	73.8	+37.0	71.6	99.5	+27.9
RDumb (Press et al., 2023)	17.8	18.4	+0.6	32.0	32.9	+0.9	59.8	56.8	−3.0	17.9	18.1	+0.2	32.4	32.6	+0.2	59.6	59.5	−0.1
PeTTA (Hoang et al., 2024)	23.0	17.2	−5.8	39.4	32.9	−6.5	67.5	60.1	−7.4	27.2	20.8	−6.4	42.1	35.3	−6.8	71.6	69.5	−2.1
LCoTTA (Duan et al., 2025)	17.8	16.8	−1.0	32.2	31.8	−0.4	59.3	57.2	−2.1	17.9	17.0	−0.9	32.4	32.1	−0.3	59.3	57.6	−1.7
EATA (Niu et al., 2022)	17.5	17.8	+0.3	30.5	30.5	0.0	57.5	55.9	−1.6	17.7	17.9	+0.2	31.0	31.1	+0.1	58.5	57.0	−1.5
+ReservoirTTA (Vray et al., 2025)	17.5	16.4	−1.1	30.6	28.4	−2.2	58.0	51.0	−7.0	17.5	16.4	−1.1	30.4	28.4	−2.0	58.5	51.8	−6.7
+ENGRAM	17.5	**15.9**	−1.6	30.5	27.4	−3.1	57.5	**49.5**	−8.0	17.7	15.9	−1.8	31.0	27.4	−3.6	58.5	**50.3**	−8.2
ROID* (Marsden et al., 2024)	17.8	17.7	−0.1	29.5	29.3	−0.2	56.1	55.5	−0.6	18.0	18.1	+0.1	30.2	30.1	−0.1	58.7	58.3	−0.4
+ReservoirTTA (Vray et al., 2025)	17.8	16.8	−1.0	29.6	27.8	−1.8	56.4	52.1	−4.3	17.9	16.8	−1.1	29.6	27.8	−1.8	57.0	53.0	−4.0
+ENGRAM	17.8	16.3	−1.5	29.5	**26.8**	−2.7	56.1	50.6	−5.5	18.0	16.3	−1.7	30.2	**26.8**	−3.4	58.7	51.5	−7.2

4 EXPERIMENTS

4.1 RECURRING CORRUPTION BENCHMARKS

Protocol. Following Vray et al. (2025), we use CIFAR-10-C with WideResNet-28, CIFAR-100-C with ResNeXt-29, and ImageNet-C with ResNet-50-BN (Hendrycks & Dietterich, 2019; Croce et al., 2021). Target labels are used only for scoring, and only explicitly marked oracle controls receive domain information. Each stream repeats 15 severity-5 corruptions for 20 recurrences. Recurring continual structure change (CSC) repeats a fixed corruption order; recurring continual dynamic change (CDC) varies domain order and visit duration. We wrap TENT (Wang et al., 2021), ETA and EATA (Niu et al., 2022), SAR (Niu et al., 2023), and ROID* (Marsden et al., 2024), retaining the reference host configurations and source-data allowance (Appendices A and C), and evaluate LCoTTA (Duan et al., 2025) under the same protocol. CoTTA* adapts only normalisation affine parameters, while ROID* omits the augmentation-consistency loss. Table 1 reports five-seed mean errors at recurrences 1 and 20 and their difference Δ; baseline values other than LCoTTA are taken from Vray et al. (2025). Separate paired CIFAR-100-C studies report 95% confidence intervals for recall and residual-carry comparisons (Appendices D and G.2).

Results and interpretation. ENGRAM achieves the lowest final error among compared methods in all six recurring corruption settings and improves on ReservoirTTA (Vray et al., 2025) with the same host in 26 of 30 comparisons, tying the remaining four at the reported precision. Under CSC, TENT's ImageNet-C error rises to 99.5%, whereas TENT + ENGRAM ends at 56.7%, compared with 58.2% for TENT + ReservoirTTA. Gains also occur when the standalone host's error remains stable: on CIFAR-100-C, EATA gives 30.5% at both recurrences 1 and 20, while EATA + ENGRAM finishes at 27.4% and ROID* + ENGRAM at 26.8%. On ImageNet-C, EATA + ENGRAM improves on EATA + ReservoirTTA by 1.5 points under both schedules. Together, these results support recall as a complement to stable adaptation: avoiding deterioration alone does not ensure that later visits benefit from earlier ones. Final accuracy must nevertheless be distinguished from improvement over the first pass. Under CSC on CIFAR-100-C, TENT + ENGRAM matches the pool's 44.0% endpoint; its large negative Δ reflects recovery from a worse first pass, rather than better final performance. The endpoint and change columns should therefore be read together.

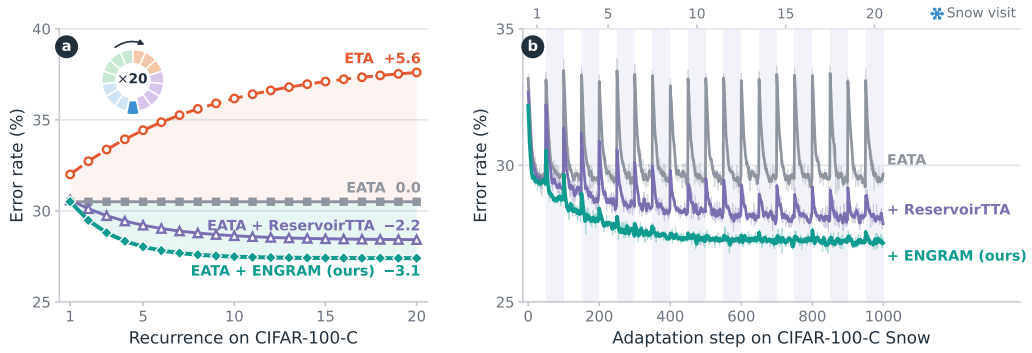


Figure 2: **Recurring CIFAR-100-C.** (a) Mean error per recurrence over the 15 corruptions. (b) Batch errors within successive Snow visits, with other corruptions omitted from the display.

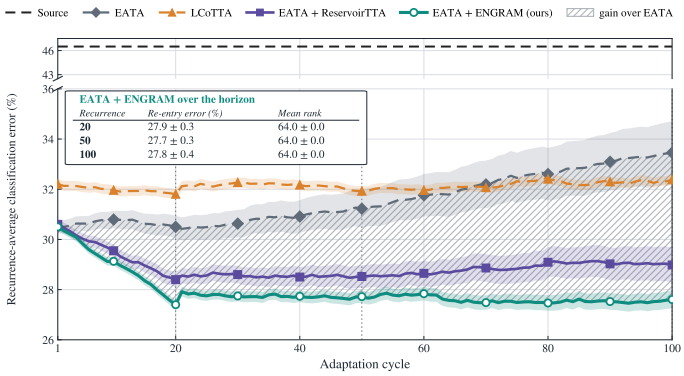


Figure 3: **Long-horizon recall.** CIFAR-100-C, five paired seeds, ± 1 SD bands. The inset gives re-entry error and rank; the basis is capped from recurrence 20.

Table 2: **Memory components.** CIFAR-100-C, recurrence 20. RE: re-entry; Avg: visit mean error (%). Five seeds.

Variant	RE	Avg
ENGRAM	**27.9**	**27.5**
No compensation	34.6	32.8
Additive write	88.4	85.2
Global decay	33.2	30.5
Stop writes after 1	31.8	29.4
$\beta = 1$	28.2	27.7
No carry	32.4	30.1

Re-entry and long horizons. Figure 2 separates improvement across recurrences from adaptation within a visit. Memory lowers re-adaptation error, with ENGRAM finishing below the matched expert pool. Its lower first-batch errors on returning Snow visits show that the benefit is present from the first scored batch. Extending CIFAR-100-C CSC to 100 recurrences without resetting model, host, or memory state preserves the advantage over EATA, its expert pool, and LCoTTA (Figure 3). The basis reaches its 64-direction cap by recurrence 20, while re-entry error remains similar at recurrences 20 and 100 (27.9% and 27.8%). These results show sustained recall after basis growth stops. Figure 8 and Table 15 report first-batch and visit-average errors; the figure also shows individual seeds. Section 4.2 examines how the memory components contribute to recall.

4.2 MEMORY MECHANISMS

The controlled experiments below use CIFAR-100-C with the EATA host. Full settings, including seed counts, are given in Appendices D.3 and G.2. Re-entry error is measured on the first returning batch, and visit-average error over the full visit.

Does compensation preserve adaptation? Removing compensation raises re-entry error by 6.7 points (Table 2). Compensation preserves the adapted parameters (Proposition 1); numerical checks confirm reconstruction and fixed-key host equivalence (Appendix D.8). Full-target additive writes perform much worse because they repeatedly add completed adjustments rather than correct retrieval error. Stopping writes after the first recurrence weakens recall, whereas immediate writes ($\beta = 1$) remain close to the default. This matches Equation (5): β controls consolidation speed, while

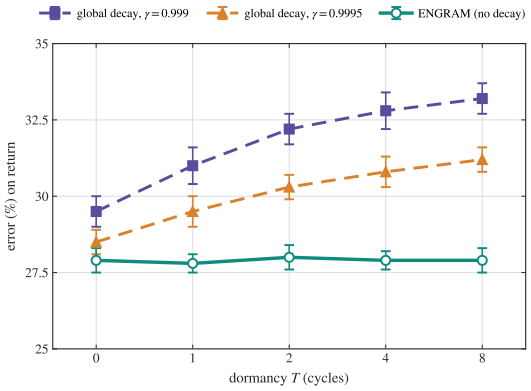


Figure 4: **Retention during absence.** CIFAR-100-C re-entry error after T cycles without a learned corruption; the default uses no global decay.

Table 3: **Address quality.** CIFAR-100-C re-entry error (%), recurrence 20, five-seed mean ± SD; † uses domain identities.

Address / reference	Error
Oracle pool†	**26.54** ± 0.41
Orthogonal keys†	27.02 ± 0.35
VGG keys (default)	27.88 ± 0.38
Correlated keys	36.81 ± 0.65

Table 4: **Preserving directions.** CIFAR-100-C visit-average error (%), recurrence 20; full ablation in Table 12.

Rank cap $r_{\max}$	16	32	64
Append-only	**29.8**	**28.2**	**27.5**
Replace least-energy	32.1	29.4	27.7

Table 5: **Cross-task summary.** Error (%, ↓), except ACDC mIoU (↑). Both rows use ETA for ViT-B/16 (recurrence 20), EATA for CCC (ResNet-50, stream mean) and styles (mean over 20 cycles and source splits), and LoRA for ACDC (recurrence 10). Protocols: Appendix E.

	ImageNet-C / ViT-B/16		ImageNet-C
Method	CSC	CDC	CCC
ReservoirTTA	31.9	32.9	58.4
ENGRAM	**30.4**	**31.4**	**56.8**
	DomainNet-126	PACS	ACDC↑
ReservoirTTA	35.13	28.68	60.0
ENGRAM	**33.65**	**27.20**	**62.7**

Table 6: **Unseen corruptions.** Mean error (%, ↓) on ten held-out ImageNet-C corruptions; ViT-B/16, severity 5. Baselines from Zhang et al. (2026b); per-corruption results in Table 25.

Method	Mean error
Source	53.3
TENT	51.6
CoTTA	52.6
ViDA	43.7
MoASE	41.0
MoASE++	39.8
EATA + ENGRAM	**38.5**

even immediate writes leave components outside the basis in the residual. Sensitivity and source-anchored host controls appear in Figures 9 and 14.

How does key overlap affect recall? Writes affect other reads in proportion to key overlap and projected write error (Equation (6)). With the same rank cap, re-entry error increases from 27.02% with orthogonal keys to 27.88% with VGG keys and 36.81% with deliberately correlated keys (Table 3). This addressing penalty persists at the larger tested rank cap (Table 11). These comparisons cover complete adaptation runs: changing keys also affects subsequent updates and basis growth. Within-domain key variation separately perturbs model initialisation (Equation (2)); Figure 10 measures both variation and overlap. Global decay weakens stored reads even without key overlap (Proposition 3). In Figure 4, return error increases with the length of absence under decay but remains stable without it. Omitting decay avoids this loss, although overlapping writes can still interfere.

How do residual carry and capacity affect recall? The residual preserves adjustments not recovered by memory, but can carry mismatched adjustments between conditions (Equation (7)). In the paired study under the main benchmark protocol (batch size 200), full carry lowers visit-average error by 0.82 points versus the expert pool and 0.75 versus similarity gating (Table 9 and Figure 16). These compare complete policies: gating also changes subsequent updates and basis growth. With a shared offline basis, full carry and oracle reset perform similarly (Figure 11), suggesting that basis learning contributes to their difference under independent growth.

At rank 16, append-only growth achieves 29.8% visit-average error, compared with 32.1% for least-energy replacement; the gap narrows at rank 64 (Table 4). Replacement accommodates new directions by discarding existing ones. Later-arriving corruptions expose the same capacity limit (Figure 12): once the basis is full, adjustments outside it remain available through the residual but cannot

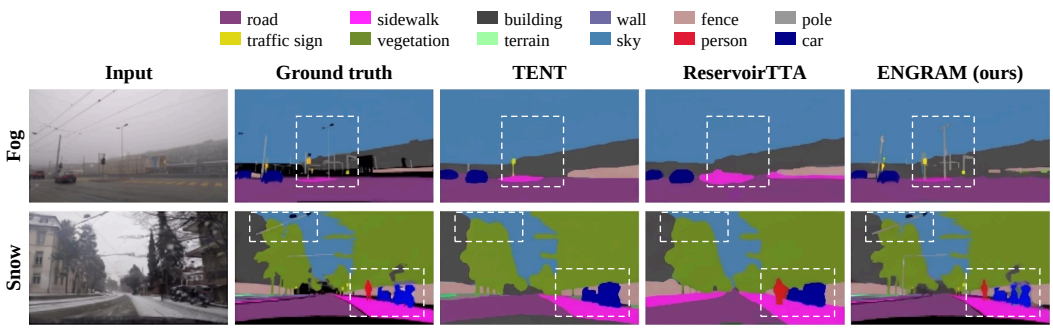


Figure 5: **Qualitative Cityscapes→ACDC results.** ReservoirTTA and ENGRAM both use BECoTTA as the host. Highlighted regions show boundaries and small structures. Colours denote semantic classes.

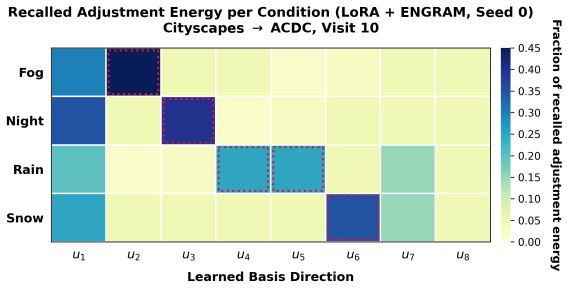


Figure 6: **Shared directions.** LoRA + ENGRAM, ACDC, seed 0, recurrence 10. Cells are per-condition squared coefficients, normalised over the displayed directions; outlined cells mark row maxima.

Table 7: **ACDC by condition.** SegFormer-B5 (Xie et al., 2021), recurrence-10 mIoU (%, ↑). Baselines from Vray et al. (2025); full results in Table 19.

Method	Fog	Night	Rain	Snow
Source	69.1	40.3	59.7	57.8
TENT	65.3	34.1	56.7	52.0
+ReservoirTTA	67.6	38.5	58.0	54.7
+ENGRAM	68.8	39.8	59.5	56.4
CoTTA*	71.2	39.9	63.3	59.8
+ReservoirTTA	72.7	40.2	64.7	60.0
+ENGRAM	**74.9**	42.0	66.6	61.8
BECoTTA	72.5	40.8	64.8	60.3
LoRA + ReservoirTTA	72.4	41.4	64.9	61.2
LoRA + ENGRAM	**74.9**	**43.7**	**67.9**	**64.2**

be stored for recall. This corresponds to the unrepresented mismatch in Equation (7), which bounds parameter error without decomposing prediction error.

4.3 ACROSS ARCHITECTURES AND TASKS

Architectures and evolving streams. Gains over same-host ReservoirTTA extend to ViT-B/16, continuously changing corruptions (CCC), and other tasks (Table 5). Benefits also hold for visual prompt tuning (VPT) (Jia et al., 2022) (Table 16) and under gradual-severity and class-correlated shifts (Table 24). However, outperforming the corresponding domain expert-based baseline does not always mean improving on the host alone (Table 22).

Transfer to unseen conditions. Following Zhang et al. (2026b), we adapt a ViT-B/16 with EATA + ENGRAM on five ImageNet-C corruptions, then evaluate on ten held-out corruptions. Adaptation, memory writes, and basis growth are disabled, and the residual is set to zero. Each batch uses the source parameters plus the adjustment retrieved from memory. ENGRAM achieves 38.5% mean error versus 39.8% for MoASE++, with the lowest error on every held-out corruption among the compared methods (Tables 6 and 25).

Object-level styles and segmentation. The DomainNet-126 (Peng et al., 2019) and PACS (Li et al., 2017) gains hold over 20 cycles and every source split (Table 23). On Cityscapes→ACDC (Sakaridis et al., 2021), LoRA + ENGRAM reaches 62.7% mIoU against 60.0% for its matched pool; CoTTA* gains as well, while TENT remains below Source in every condition at recurrence 10 (Table 7). Figure 5 provides qualitative examples of these boundary and small-structure differences.

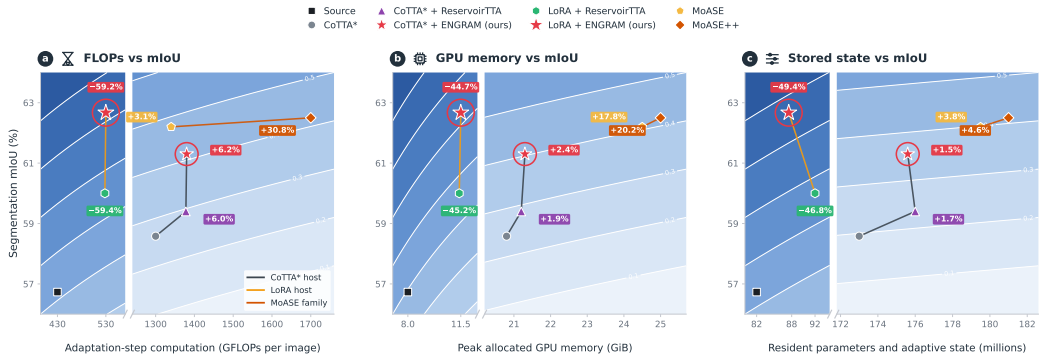


Figure 7: **Segmentation accuracy and resource cost.** SegFormer-B5 (Xie et al., 2021), batch one: (a) adaptation-step computation, (b) peak GPU allocation, (c) resident model and adaptive-state scalars. Deltas use CoTTA* as the cost reference; shading only marks the direction of better trade-offs (higher mIoU at lower cost). Interpret memory comparisons within the same host.

Shared directions and dense prediction. The learned basis is shared across conditions, not partitioned by them. In Figure 6, u_1 carries energy under all four conditions and Rain and Snow additionally share u_7; the display shows the eight highest-energy directions, not the full basis. The segmentation ablations show that rank alone does not explain recall (Table 8). Removing compensation grows a larger basis yet lowers mIoU, and global decay leaves the rank almost unchanged yet weakens early-return performance.

Computational and memory cost. Relative to LoRA + ReservoirTTA, LoRA + ENGRAM uses approximately 0.4% more computation and 0.9% more peak GPU memory, with 4.9% fewer stored scalars (Figure 7). ViT runtimes round to $3.0\times$ source inference for both methods (Table 16). LoRA and CoTTA* use different adaptation procedures, so resource comparisons should be made within each host. Memory accounting must include both memory factors and working state (Appendix C).

Table 8: **Segmentation ablations.** LoRA on ACDC, five seeds; mIoU (%, ↑). Final: recurrence 10. Early-return: mIoU per ten-image window, averaged over conditions and recurrences 2–10. r: mean final rank. Details: Table 20.

Variant	Final	Early	r
LoRA host	58.6	56.5	–
LoRA + ReservoirTTA	60.0	59.4	–
No compensation	57.2	56.2	58.4
No residual carry	60.8	59.6	28.6
Global decay ($\gamma = .999$)	61.1	59.2	45.8
Fixed after recurrence 1	60.4	60.1	16.0
LoRA + ENGRAM	**62.7**	**61.8**	45.2

Table 9: **Residual carry.** CIFAR-100-C with EATA, batch 200. Visit-average error (%, ↓) at recurrence 20; five paired seeds, mean ± SD. Details: Appendix G.2.

Method	Error
Source	46.59 ± 0.14
EATA	30.63 ± 0.25
+ReservoirTTA	28.14 ± 0.28
+ENGRAM, gate	28.07 ± 0.26
+ENGRAM, full	$\mathbf{27.32} \pm 0.16$

5 CONCLUSION

ENGRAM enables a test-time adapter to reuse condition-dependent adaptations through shared parameter directions. Experiments demonstrate gains across corruption, style, and segmentation streams under recurring and evolving shifts. Long-horizon and held-out evaluations show sustained recall and transfer to unseen corruptions.

Limitations. Conditions with similar visual descriptors can interfere in memory. Limited capacity prevents some new adjustments from being stored for later recall. Memory can retain host errors and adds resource costs. Our theory describes parameter updates; it does not guarantee accuracy gains or convergence for every host.

Reproducibility statement

Section 2 and Appendices A and C describe the algorithm, memory settings, integration with existing TTA methods, and storage costs. Proofs are given in Appendix B. Protocols, ablations, seed counts, and paired statistics are provided in Appendices D, E and G.

Ethics statement

This work uses existing perception benchmarks under their original licences. An inaccurate memory read or host update can degrade predictions under shift, and although the method stores no samples, its stored adjustments reflect the data seen during adaptation.

Statement on the use of large language models

We used AI tools exclusively to polish the text, fix typos, improve wording, and check grammar. We did not use generative AI tools for methodology development, dataset creation, or reference compilation. The authors are fully responsible for verifying the manuscript, and experimental records.

References

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Sunghyun Baek, Jaemyung Yu, Seunghee Koh, Minsu Kim, Hyeonseong Jeon, and Junmo Kim. IMSE: Intrinsic mixture of spectral experts fine-tuning for test-time adaptation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eZO38vANPM.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025.

Guohao Chen, Shuaicheng Niu, Deyu Chen, Shuhai Zhang, Changsheng Li, Yuanqing Li, and Mingkui Tan. Cross-device collaborative test-time adaptation. *Advances in Neural Information Processing Systems*, 37:122917–122951, 2024.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

Dexin Duan, Rui Xu, Peilin Liu, and Fei Wen. Lifelong test-time adaptation via online learning in tracked low-dimensional subspace. In *Advances in Neural Information Processing Systems*, 2025.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Trung-Hieu Hoang, Duc Minh Vo, and Minh N Do. Persistent test-time adaptation in recurring testing scenarios. In *Advances in Neural Information Processing Systems*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pp. 709–727. Springer, 2022.

Daeun Lee, Jaehong Yoon, and Sung Ju Hwang. BECoTTA: Input-dependent online blending of experts for continual test-time adaptation. In *International Conference on Machine Learning*, 2024.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5543–5551. Ieee, 2017.

Jiaming Liu, Senqiao Yang, Peidong Jia, Renrui Zhang, Ming Lu, Yandong Guo, Wei Xue, and Shanghang Zhang. ViDA: Homeostatic visual domain adapter for continual test time adaptation. In *International Conference on Learning Representations*, 2024.

Robert A Marsden, Mario Döbler, and Bin Yang. Universal test-time adaptation through weight ensembling, diversity weighting, and prior correction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2555–2565, 2024.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1406–1415. IEEE, 2019.

Ori Press, Steffen Schneider, Matthias Kümmerer, and Matthias Bethge. RDumb: A simple approach that questions our progress in continual test-time adaptation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10765–10775, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366, 2021.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278, 2009.

Guanglong Sun, Shuang Cui, Bo Lei, Liyuan Wang, Zihan Zhai, Hongwei Yan, Hang Su, Jun Zhu, and Yi Zhong. ComMem: Complementary memory systems for test-time adaptation of vision-language models. *arXiv preprint arXiv:2606.28719*, 2026.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 57503–57522. PMLR, 2025.

Guillaume Vray, Devavrat Tomar, Xufeng Gao, Jean-Philippe Thiran, Evan Shelhamer, and Behzad Bozorgtabar. ReservoirTTA: Prolonged test-time adaptation for evolving and recurring domains. In *Advances in Neural Information Processing Systems*, 2025.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: A unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022.

Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. In *IRE WESCON Convention Record*, volume 4, pp. 96–104, 1960.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

Jiaming Yi, Ruirui Pan, Jishen Yang, and Xiulong Yang. MINT: Memory-infused prompt tuning at test-time for CLIP. *arXiv preprint arXiv:2506.03190*, 2025.

Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15922–15932, 2023.

Rongyu Zhang, Aosong Cheng, Yulin Luo, Gaole Dai, Huanrui Yang, Jiaming Liu, Ran Xu, Li Du, Dan Wang, and Yuan Du. Decomposing the neurons: Activation sparsity via mixture of experts for continual test time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 36057–36065, 2026a.

Ronyu Zhang, Aosong Cheng, Gaole Dai, Yulin Luo, Jiaming Liu, Li Du, Huanrui Yang, Dan Wang, Leyuan Fang, Yuan Du, and Shanghang Zhang. MoASE++: Mixture of activation sparsity experts with domain-adaptive on-policy distillation for continual test time adaptation. *arXiv preprint arXiv:2605.17743*, 2026b.

Yunbei Zhang, Akshay Mehra, Shuaicheng Niu, and Jihun Hamm. DPCore: Dynamic prompt coreset for continual test-time adaptation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 75757–75778. PMLR, 2025.

A Algorithm and Implementation of Memory Updates

Algorithm 2 specifies the default implementation used for the reported experiments and state accounting, with explicit residual storage and full carry-over. The host's auxiliary state $\mathcal{S}_t$ contains quantities maintained in addition to the adapted coordinates ϕ, such as optimiser statistics, teacher parameters, and sample buffers when present. Predictions $\widehat{\mathbf{p}}_t$ follow the host's original prediction model and timing. The completed parameters ϕ_t^+ are recorded after all host operations, including any restoration or source-weight averaging.

Algorithm 2 ENGRAM: full carry-over and append-only growth

Require: Source parameters ϕ_{src}; configured host $\mathcal{H}$ and initial state $\mathcal{S}_0$; fixed key map g with $\|g(\mathbf{x})\|_2 = 1$; settings from Table 10

Initialise $\mathbf{U} \in \mathbb{R}^{d\times 0}$, $\mathbf{M} \in \mathbb{R}^{0\times m}$, $\boldsymbol{\delta} = \bar{\boldsymbol{\rho}} = \mathbf{0}$, $n = 0$, $r = 0$
for each unlabelled batch $\mathbf{x}_t$ **do**
Read — Retrieve the current adjustment
  $\mathbf{k}_t \leftarrow g(\mathbf{x}_t)$
  $\mathbf{a}_t \leftarrow \mathbf{U}(\mathbf{M}\mathbf{k}_t)$; $\phi_t^- \leftarrow \phi_{\mathrm{src}} + \mathbf{a}_t + \boldsymbol{\delta}$
  Install ϕ_t^- in the existing parameter tensors
Adapt — Keep the host's update and prediction rules
  $(\phi_t^+, \mathcal{S}_t, \widehat{\mathbf{p}}_t) \leftarrow \mathcal{H}(\phi_t^-, \mathcal{S}_{t-1}; \mathbf{x}_t)$; emit $\widehat{\mathbf{p}}_t$
Write and compensate — Store without changing the completed model
  $\mathbf{y}_t \leftarrow \phi_t^+ - \phi_{\mathrm{src}}$; $\mathbf{e}_t \leftarrow \mathbf{U}^\top \mathbf{y}_t - \mathbf{M}\mathbf{k}_t$
  $\mathbf{M} \leftarrow \mathbf{M} + \beta \mathbf{e}_t \mathbf{k}_t^\top$
  $\boldsymbol{\delta} \leftarrow \mathbf{y}_t - \mathbf{U}(\mathbf{M}\mathbf{k}_t)$
Grow — Append directions without changing existing reads
  $\mathbf{b}_t \leftarrow \boldsymbol{\delta} - \mathbf{U}(\mathbf{U}^\top \boldsymbol{\delta})$
  $\bar{\boldsymbol{\rho}} \leftarrow \lambda \bar{\boldsymbol{\rho}} + (1-\lambda)\mathbf{b}_t$; $n \leftarrow n+1$
  $\mathbf{z} \leftarrow [\bar{\boldsymbol{\rho}} - \mathbf{U}(\mathbf{U}^\top \bar{\boldsymbol{\rho}})]/(1-\lambda^n)$
  if $n \geq n_{\min}$ and $r < \min(r_{\max}, d)$ and $\|\mathbf{z}\| > \epsilon \|\mathbf{y}_t\|$ **then**
    Re-orthogonalise $\mathbf{z}$ against $\mathbf{U}$
    if $\|\mathbf{z}\|$ is numerically nonzero **then**
      $\mathbf{u} \leftarrow \mathbf{z}/\|\mathbf{z}\|$
      $\mathbf{U} \leftarrow [\mathbf{U}\ \mathbf{u}]$; $\mathbf{M} \leftarrow \begin{bmatrix} \mathbf{M} \\ \mathbf{0}_{1\times m} \end{bmatrix}$
      $r \leftarrow r+1$; $\bar{\boldsymbol{\rho}} \leftarrow \mathbf{0}$; $n \leftarrow 0$
    end if
  end if
  Retain $\mathbf{k}_t$ for diagnostics and the optional carry ablation
end for

Growth statistics. Between growth events the basis is fixed. We maintain a smoothed adjustment outside this basis and normalise its averaging weights:

$$\begin{aligned} \bar{\boldsymbol{\rho}} &\leftarrow \lambda \bar{\boldsymbol{\rho}} + (1-\lambda)\mathbf{P}^\perp \boldsymbol{\delta}_t, \qquad n \leftarrow n+1, \\ \mathbf{z} &= \frac{\mathbf{P}^\perp \bar{\boldsymbol{\rho}}}{1-\lambda^n}, \qquad \mathbf{P}^\perp = \mathbf{I} - \mathbf{U}\mathbf{U}^\top. \end{aligned} \tag{8}$$

Here n counts batches since initialisation or the last appended direction. The default minimum spacing is $n_{\min} = 10$ batches. The other growth settings are $\lambda = 0.9$, $\epsilon = 0.25$, and $r_{\max} = 64$; the write rate is $\beta = 0.1$. Table 10 collects these values and their roles. A direction is appended only when three conditions hold at once: at least $n_{\min}$ batches have passed since the initialisation or the last append, the rank is below its cap, and the bias-corrected candidate has norm above $\epsilon\|\mathbf{y}_t\|$. After an append, the running average $\bar{\boldsymbol{\rho}}$ and the counter n are reset; otherwise their updated values are retained for the next batch. A candidate whose norm is numerically zero is discarded rather than normalised. Every projection outside the basis is computed as $\mathbf{v} - \mathbf{U}(\mathbf{U}^\top \mathbf{v})$, so no dense $d \times d$ projector is required.

Equivalent difference of reads. The residual is recomputed directly rather than optimised separately: $\boldsymbol{\delta}_t = \boldsymbol{\phi}_t^+ - \boldsymbol{\phi}_{\mathrm{src}} - \mathbf{A}_t\mathbf{k}_t$. Substituting this identity into the next read gives the second line of Equation (2). An equivalent full-carry implementation can therefore update the previous working parameters using $\mathbf{A}_{t-1}(\mathbf{k}_t - \mathbf{k}_{t-1})$ and compute growth from $\mathbf{P}^\perp\mathbf{y}_t$, because $\mathbf{P}^\perp\boldsymbol{\delta}_t = \mathbf{P}^\perp\mathbf{y}_t$. For a uniform expression starting at $t = 1$, set $\boldsymbol{\phi}_0^+ = \boldsymbol{\phi}_{\mathrm{src}}$, $\mathbf{A}_0 = \mathbf{0}$, and $\mathbf{k}_0 = \mathbf{0}$; the last is an initialisation sentinel, not a batch key. This equivalence holds in exact arithmetic; changing the order of floating-point operations can introduce rounding differences. The reported storage budgets count the explicit-residual implementation, including its saved key, as specified in Appendix C.

Optional similarity gating. The carry ablation replaces $\boldsymbol{\delta}_{t-1}$ in the read by $h_t\boldsymbol{\delta}_{t-1}$, where $h_1 = 1$ and $h_t = \max(0, \langle \mathbf{k}_{t-1}, \mathbf{k}_t\rangle)^2$ for $t \geq 2$. Equivalently,

$$\boldsymbol{\phi}_t^- = \boldsymbol{\phi}_{t-1}^+ + \mathbf{A}_{t-1}(\mathbf{k}_t - \mathbf{k}_{t-1}) - (1 - h_t)\boldsymbol{\delta}_{t-1} \qquad (t \geq 2). \tag{9}$$

The write, compensation, and growth rules remain unchanged. The gate reduces residual carry when consecutive descriptors are less similar, including differences caused by within-domain noise. Figure 11 and Appendix G.2 report the diagnostic and main-protocol carry comparisons, respectively.

Host state. Retrieved parameters are written into the host's existing tensors, so they keep their optimiser associations. The host then updates its own auxiliary state, such as optimiser moments and teacher parameters, by its own rules; ENGRAM never restores a domain-specific copy of that state. The write target uses the completed parameters $\boldsymbol{\phi}_t^+$, recorded after all host operations, including any restoration or averaging towards the source weights. The memory algorithm uses no target labels, domain identifiers, or stream boundaries.

B ADDITIONAL THEORY AND PROOFS

We first establish properties of the implemented read–adapt–write cycle, then analyse recurrence under additional assumptions and derive the memory's representational limits. All identities are stated in exact arithmetic. The source reference and adapted coordinate system remain fixed throughout each run. Unless stated otherwise, the write rate is constant with $0 < \beta \leq 1$.

Vector norms are Euclidean, matrix norms with subscript 2 are spectral, and $\|\cdot\|_F$ denotes the Frobenius norm. Keys have unit norm, the columns of $\mathbf{U}_t$ are orthonormal, and $\mathbf{A}_t = \mathbf{U}_t\mathbf{M}_t$ and $\mathbf{P}_t = \mathbf{U}_t\mathbf{U}_t^\top$ denote the memory map and projector after batch t.

The write precedes basis growth. Since growth appends a zero row to the coefficient map, it leaves $\mathbf{A}$ unchanged. Consequently, including at growth events, the memory increment is

$$\mathbf{A}_t - \mathbf{A}_{t-1} = \beta\mathbf{P}_{t-1}\mathbf{v}_t\mathbf{k}_t^\top, \qquad \mathbf{v}_t = \mathbf{y}_t - \mathbf{A}_{t-1}\mathbf{k}_t. \tag{10}$$

This identity follows by multiplying the coefficient update by $\mathbf{U}_{t-1}$.

B.1 CONSERVATION AND CONTINUITY

Proposition 1 (Conservation and continuity)**.** *For every batch,*

$$\boldsymbol{\phi}_{\mathrm{src}} + \mathbf{A}_t\mathbf{k}_t + \boldsymbol{\delta}_t = \boldsymbol{\phi}_t^+. \tag{11}$$

For a carry factor $h_{t+1} \in [0, 1]$, let $\eta_t = \|\mathbf{k}_{t+1} - \mathbf{k}_t\|$. Then

$$\boldsymbol{\phi}_{t+1}^- - \boldsymbol{\phi}_t^+ = \mathbf{A}_t(\mathbf{k}_{t+1} - \mathbf{k}_t) + (h_{t+1} - 1)\boldsymbol{\delta}_t, \tag{12}$$

$$\|\boldsymbol{\phi}_{t+1}^- - \boldsymbol{\phi}_t^+\| \leq \|\mathbf{A}_t\|_2\eta_t + (1 - h_{t+1})\|\boldsymbol{\delta}_t\|. \tag{13}$$

For the similarity gate in Equation (9), $h_{t+1} = \max(0, \langle \mathbf{k}_t, \mathbf{k}_{t+1}\rangle)^2$, and $1 - h_{t+1} \leq \eta_t^2$. For full carry and the specified similarity gate, a constant-key interval reproduces the host's trajectory when entry parameters, auxiliary state, batches, and random draws are matched. This holds for any host, any $\beta \in [0, 1]$, and across basis-growth events.

Proof. The residual is defined by $\boldsymbol{\delta}_t = \mathbf{y}_t - \mathbf{A}_t\mathbf{k}_t$, where $\mathbf{y}_t = \boldsymbol{\phi}_t^+ - \boldsymbol{\phi}_{\mathrm{src}}$, proving Equation (11). Subtracting this identity from the next read proves Equation (12); the triangle inequality gives Equation (13).

For the gate, set $c = \langle \mathbf{k}_t, \mathbf{k}_{t+1} \rangle$. Unit norms imply $\eta_t^2 = 2(1-c)$. If $c \geq 0$, $1 - h_{t+1} = 1 - c^2 \leq 2(1-c)$; if $c < 0$, $1 - h_{t+1} = 1 < 2(1-c)$. On a constant key, both policies give $h_{t+1} = 1$ and hence $\boldsymbol{\phi}_{t+1}^- = \boldsymbol{\phi}_t^+$. Applying the unchanged host transition with matched auxiliary state and random draws then proves trajectory equivalence by induction. Growth changes neither the map nor the residual, so it does not affect this argument. □

This is equivalence to a host started from the same state at entry to the interval. It does not assert equivalence to a host that experienced a different history before entry, or restoration of a returning domain's earlier optimiser or teacher state.

B.2 Consolidation as a filtered host trajectory

Proposition 2 (The memory is a filtered host trajectory)**.** *Consider a constant-key interval with a fixed basis and constant write rate $0 < \beta \leq 1$. Let $\mathbf{P} = \mathbf{U}\mathbf{U}^\top$, $\rho = 1 - \beta$, and $\mathbf{q}_t = \mathbf{A}_t \mathbf{k}$. Index batches locally from $t = 1$, with $\mathbf{q}_0$ the read at interval entry. Then*

$$\mathbf{q}_{t+1} = \rho \mathbf{q}_t + \beta \mathbf{P} \mathbf{y}_{t+1}, \tag{14}$$

$$\mathbf{q}_t = \rho^t \mathbf{q}_0 + \beta \sum_{u=1}^{t} \rho^{t-u} \mathbf{P} \mathbf{y}_u. \tag{15}$$

If $\mathbf{y}_t \to \mathbf{y}^*$, then $\mathbf{q}_t \to \mathbf{P}\mathbf{y}^*$ and $\boldsymbol{\delta}_t \to (\mathbf{I} - \mathbf{P})\mathbf{y}^*$.

If the host additionally satisfies $\|\mathbf{y}_u - \mathbf{y}^*\| \leq a^u \|\mathbf{y}_0 - \mathbf{y}^*\|$ for $0 < a < 1$, where $\mathbf{y}_0$ is its displacement at interval entry, then after S batches

$$\begin{aligned} \|\mathbf{q}_S - \mathbf{P}\mathbf{y}^*\| \leq \rho^S \|\mathbf{q}_0 - \mathbf{P}\mathbf{y}^*\| \\ + \beta \left(\sum_{u=1}^{S} \rho^{S-u} a^u \right) \|\mathbf{y}_0 - \mathbf{y}^*\|. \end{aligned} \tag{16}$$

The geometric factor equals $a(\rho^S - a^S)/(\rho - a)$ when $a \neq \rho$, and $S a^S$ when $a = \rho$.

Proof. Multiplying Equation (10) by the constant unit key gives

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \beta \mathbf{P}(\mathbf{y}_{t+1} - \mathbf{q}_t) = \rho \mathbf{q}_t + \beta \mathbf{P} \mathbf{y}_{t+1},$$

because $\mathbf{P}\mathbf{q}_t = \mathbf{q}_t$. Iteration gives Equation (15). After subtracting $\mathbf{P}\mathbf{y}^*$,

$$\mathbf{q}_S - \mathbf{P}\mathbf{y}^* = \rho^S (\mathbf{q}_0 - \mathbf{P}\mathbf{y}^*) + \beta \sum_{u=1}^{S} \rho^{S-u} \mathbf{P}(\mathbf{y}_u - \mathbf{y}^*). \tag{17}$$

If the host converges, split the sum at an index beyond which its error is arbitrarily small. The finite prefix tends to zero and the remaining weights sum to at most one. This proves convergence of the read; convergence of the residual follows from $\boldsymbol{\delta}_t = \mathbf{y}_t - \mathbf{q}_t$. For $\beta = 1$, the same conclusions follow directly from $\mathbf{q}_t = \mathbf{P}\mathbf{y}_t$. Finally, apply the triangle inequality, $\|\mathbf{P}\|_2 \leq 1$, and the assumed host bound to Equation (17). Evaluating the finite geometric sum gives the two stated cases. □

The bound retains separate entry errors for the host and the memory. Conservation does not make them equal. For example, if the host reaches a fixed representable target immediately but the entry read is zero, the unconsolidated fraction after S writes is $(1-\beta)^S$. The proposition does not assume or establish convergence of an arbitrary host; it describes what memory stores along its completed trajectory.

B.3 Retention and interference

Proposition 3 (Overlap-dependent retention)**.** *For a fixed unit query $\mathbf{k}_\star$ and batch indices $s < T$,*

$$(\mathbf{A}_T - \mathbf{A}_s)\mathbf{k}_\star = \beta \sum_{t=s+1}^{T} \langle \mathbf{k}_t, \mathbf{k}_\star \rangle \mathbf{P}_{t-1} \mathbf{v}_t, \tag{18}$$

$$\|(\mathbf{A}_T - \mathbf{A}_s)\mathbf{k}_\star\| \leq \beta \sum_{t=s+1}^{T} |\langle \mathbf{k}_t, \mathbf{k}_\star \rangle| \, \|\mathbf{P}_{t-1} \mathbf{v}_t\|. \tag{19}$$

In particular:

(a) Orthogonal queries. *If $\mathbf{k}_\star$ is orthogonal to every intervening write key, its read is preserved exactly, including across growth events.*

(b) Fixed-key visits. *Suppose $(s, T]$ is partitioned into visits $(s_\ell, e_\ell]$, $\ell = 1, \dots, L$, each using one fixed key $\mathbf{k}^{(\ell)}$. Then*

$$(\mathbf{A}_T - \mathbf{A}_s)\mathbf{k}_\star = \sum_{\ell=1}^{L} \langle \mathbf{k}^{(\ell)}, \mathbf{k}_\star \rangle (\mathbf{A}_{e_\ell} - \mathbf{A}_{s_\ell})\mathbf{k}^{(\ell)}. \tag{20}$$

Visits may use the same key repeatedly.

(c) Global decay. *For an alternative rule $\mathbf{A}_t = \gamma \mathbf{A}_{t-1} + \mathbf{b}_t \mathbf{k}_t^\top$, where $0 \le \gamma < 1$ and $\mathbf{b}_t \in \mathbb{R}^d$, a query orthogonal to every intervening key satisfies*

$$\mathbf{A}_T \mathbf{k}_\star = \gamma^{T-s} \mathbf{A}_s \mathbf{k}_\star, \qquad \|(\mathbf{A}_T - \mathbf{A}_s)\mathbf{k}_\star\| = (1 - \gamma^{T-s})\|\mathbf{A}_s \mathbf{k}_\star\|. \tag{21}$$

Proof. Sum Equation (10) and multiply by $\mathbf{k}_\star$ to obtain Equation (18). The triangle inequality gives Equation (19), and orthogonality gives part (a).

Within a fixed-key visit, every increment has the form $\mathbf{b}_t \mathbf{k}^\top$. Thus

$$(\mathbf{A}_e - \mathbf{A}_s)\mathbf{k}_\star = \langle \mathbf{k}, \mathbf{k}_\star \rangle \sum_{t=s+1}^{e} \mathbf{b}_t, \qquad (\mathbf{A}_e - \mathbf{A}_s)\mathbf{k} = \sum_{t=s+1}^{e} \mathbf{b}_t.$$

Summing over visits proves part (b). For part (c), multiplying the decayed update by an orthogonal query removes its write term at every step. Iteration then gives Equation (21). □

Part (b) concerns changes in the stored read. A stationary host target can still produce interference while memory is catching up. Part (c) quantifies loss relative to the read before dormancy; it is not an error bound against an unknown domain optimum. It applies specifically to global scaling combined with writes along the active key.

B.4 Proof and interpretation of the re-entry bound

Theorem 1 (Re-entry decomposition)**.** *Suppose the preceding batch has key $\mathbf{k}_j$ and completed displacement $\mathbf{y}_j$. Let $\mathbf{A}$ and $\mathbf{P}$ denote the memory map and basis projector after that batch. The next batch has key $\mathbf{k}_i$ and carry factor $h \in [0, 1]$. For reference displacements $\mathbf{y}_j^*$ and $\mathbf{y}_i^*$, with $\mathbf{P}^\perp = \mathbf{I} - \mathbf{P}$,*

$$\begin{aligned} \|\boldsymbol{\phi}_i^- - \boldsymbol{\phi}_{\mathrm{src}} - \mathbf{y}_i^*\| \le\ & \underbrace{\|\mathbf{A}\mathbf{k}_i - \mathbf{P}\mathbf{y}_i^*\|}_{\textit{retrieval error}} + \underbrace{\|\mathbf{P}^\perp(h\mathbf{y}_j^* - \mathbf{y}_i^*)\|}_{\textit{out-of-basis mismatch}} \\ & + \underbrace{h\|\mathbf{P}^\perp(\mathbf{y}_j - \mathbf{y}_j^*)\|}_{\textit{carried adaptation error}} + \underbrace{h\|\mathbf{P}\mathbf{y}_j - \mathbf{A}\mathbf{k}_j\|}_{\textit{consolidation lag}}. \end{aligned} \tag{22}$$

Proof of Theorem 1. Conservation on the preceding batch gives $\boldsymbol{\delta}_j = \mathbf{y}_j - \mathbf{A}\mathbf{k}_j$. The next initialisation is therefore $\boldsymbol{\phi}_i^- - \boldsymbol{\phi}_{\mathrm{src}} = \mathbf{A}\mathbf{k}_i + h\boldsymbol{\delta}_j$. Since $\mathbf{A}\mathbf{k}_j$ lies in the basis, the residual decomposes as

$$\boldsymbol{\delta}_j = \mathbf{P}^\perp \mathbf{y}_j + (\mathbf{P}\mathbf{y}_j - \mathbf{A}\mathbf{k}_j).$$

Subtract $\mathbf{y}_i^*$ and add and subtract $h\mathbf{P}^\perp \mathbf{y}_j^*$ to obtain the exact identity

$$\begin{aligned} \boldsymbol{\phi}_i^- - \boldsymbol{\phi}_{\mathrm{src}} - \mathbf{y}_i^* = & (\mathbf{A}\mathbf{k}_i - \mathbf{P}\mathbf{y}_i^*) + \mathbf{P}^\perp(h\mathbf{y}_j^* - \mathbf{y}_i^*) \\ & + h\mathbf{P}^\perp(\mathbf{y}_j - \mathbf{y}_j^*) + h(\mathbf{P}\mathbf{y}_j - \mathbf{A}\mathbf{k}_j). \end{aligned} \tag{23}$$

The triangle inequality proves the theorem. □

For the default full-carry policy, the same calculation can be written without a reference for the preceding condition:

$$\begin{aligned}\boldsymbol{\phi}_i^- - \boldsymbol{\phi}_{\mathrm{src}} - \mathbf{y}_i^* = (\mathbf{A}\mathbf{k}_i - \mathbf{P}\mathbf{y}_i^*) + (\mathbf{P}\mathbf{y}_j - \mathbf{A}\mathbf{k}_j) \\ + \mathbf{P}^{\perp}(\mathbf{y}_j - \mathbf{y}_i^*).\end{aligned} \tag{24}$$

The triangle inequality gives Equation (7). The first two vectors lie in the basis and the third is orthogonal to it. The four-term theorem expands the third vector around the preceding condition's reference displacement and additionally allows gated carry-over.

The consolidation-lag term in Equation (22) is the mismatch between the preceding host's completed displacement within the basis and its stored read. It vanishes when that displacement is fully consolidated, irrespective of whether the host has reached its reference state. The preceding condition j is the one immediately before the return; it need not be an earlier visit to condition i. The vector decomposition is exact. Error vectors in the same subspace can cancel, so summing their norms gives an upper bound on parameter error.

For fixed displacements, references, and carry factor, appending directions cannot increase either of the two orthogonally projected terms in Equation (22). However, a new direction initially has zero coefficients, so growth may increase retrieval error and within-basis consolidation lag even though it leaves the model and all memory reads unchanged. The theorem therefore does not assert monotonic improvement of the total bound under growth.

B.5 Recall across repeated visits

The next result studies an idealised process with stationary keys, a fixed basis, and contracting visit dynamics. It is separate from the identities above, which do not require the host or the induced visit map to be linear.

Proposition 4 (Idealised recurrence)**.** *Let $\mathbf{k}_1, \dots, \mathbf{k}_K \in \mathbb{R}^m$ be linearly independent unit keys, let $\mathbf{K} = [\mathbf{k}_1, \dots, \mathbf{k}_K]$ and $\mathbf{G} = \mathbf{K}^\top\mathbf{K}$, and fix the basis. Index completed visits by n, with $\mathbf{M}_0 = \mathbf{0}$. Each visit n uses one key $\mathbf{k}_{j_n}$ and changes the map only along that key: $\mathbf{M}_{n+1} - \mathbf{M}_n = \mathbf{g}_n\mathbf{k}_{j_n}^\top$. Assume stationary coefficient targets $\hat{\mathbf{c}}_j^* \in \mathbb{R}^r$ and visit dynamics*

$$\mathbf{M}_{n+1}\mathbf{k}_{j_n} - \hat{\mathbf{c}}_{j_n}^* = \mathbf{T}_{j_n}(\mathbf{M}_n\mathbf{k}_{j_n} - \hat{\mathbf{c}}_{j_n}^*) + \xi_n, \tag{25}$$

where $\mathbf{T}_j \in \mathbb{R}^{r\times r}$ are fixed, $\|\mathbf{T}_j\|_2 \le a < 1$, $a \ge 0$, and $\|\xi_n\| \le \bar{n}$. The disturbances may depend on the history and need not have zero mean. Their uniform bound is an assumption.

Let $\mathbf{M}^*$ be the minimum-Frobenius-norm map satisfying $\mathbf{M}^*\mathbf{k}_j = \hat{\mathbf{c}}_j^*$ for every j, and let $\mathbf{E}_n = \mathbf{M}_n - \mathbf{M}^*$.

(a) Independent uniform visits. *If each j_n is sampled uniformly and independently of the process history before visit n, then*

$$\begin{aligned}\mathbb{E}\|\mathbf{E}_n\|_F^2 \le \left(1 - \frac{(1-a^2)\lambda_{\min}(\mathbf{G})}{2K}\right)^n \|\mathbf{E}_0\|_F^2 \\ + \frac{2K(1+a^2)}{(1-a^2)^2\lambda_{\min}(\mathbf{G})}\bar{n}^2.\end{aligned} \tag{26}$$

(b) Fixed cyclic visits. *For a fixed order visiting every key once per cycle, there is a $\theta < 1$ such that, at cycle boundaries,*

$$\|\mathbf{E}_{\ell K}\|_F \le \theta^\ell\|\mathbf{E}_0\|_F + \frac{K\bar{n}(1-\theta^\ell)}{1-\theta}. \tag{27}$$

Without disturbance, the memory converges to $\mathbf{M}^*$ in mean square under (a) and in norm at cycle boundaries under (b). With disturbance, part (b) gives limiting upper bound $K\bar{n}/(1-\theta)$ at cycle boundaries.

Proof. Write $\mathbf{E} = \mathbf{M}_n - \mathbf{M}^*$ and $\mathbf{k} = \mathbf{k}_{j_n}$. The active-key update and unit norm give

$$\mathbf{g}_n = (\mathbf{T}_{j_n} - \mathbf{I})\mathbf{E}\mathbf{k} + \xi_n, \qquad \mathbf{E}_{n+1} = \mathbf{E}(\mathbf{I} - \mathbf{k}\mathbf{k}^\top) + (\mathbf{T}_{j_n}\mathbf{E}\mathbf{k} + \xi_n)\mathbf{k}^\top.$$

The two terms are orthogonal in the Frobenius inner product. Consequently,

$$\|\mathbf{E}_{n+1}\|_F^2 = \|\mathbf{E}\|_F^2 - \|\mathbf{E}\mathbf{k}\|^2 + \|\mathbf{T}_{j_n}\mathbf{E}\mathbf{k} + \xi_n\|^2. \tag{28}$$

For $a > 0$, Young's inequality with $\eta = (1-a^2)/(2a^2)$ gives

$$\|\mathbf{T}_{j_n}\mathbf{E}\mathbf{k} + \xi_n\|^2 \le \frac{1+a^2}{2}\|\mathbf{E}\mathbf{k}\|^2 + \frac{1+a^2}{1-a^2}\bar{n}^2.$$

Substitution yields

$$\|\mathbf{E}_{n+1}\|_F^2 \le \|\mathbf{E}\|_F^2 - \frac{1-a^2}{2}\|\mathbf{E}\mathbf{k}\|^2 + \frac{1+a^2}{1-a^2}\bar{n}^2. \tag{29}$$

For $a = 0$, Equation (28) directly gives a stronger bound and hence also Equation (29).

Let $\mathcal{S} = \mathrm{span}\{\mathbf{k}_1, \ldots, \mathbf{k}_K\}$. The rows of the minimum-norm map $\mathbf{M}^*$ lie in $\mathcal{S}$. Because $\mathbf{M}_0 = \mathbf{0}$ and each update is along a key, the rows of every $\mathbf{E}_n$ also lie in $\mathcal{S}$. For an independent uniform choice of j_n, conditionally on the past,

$$\mathbb{E}\big[\|\mathbf{E}_n\mathbf{k}_{j_n}\|^2 \mid \mathbf{E}_n\big] = \frac{1}{K}\|\mathbf{E}_n\mathbf{K}\|_F^2 \ge \frac{\lambda_{\min}(\mathbf{G})}{K}\|\mathbf{E}_n\|_F^2.$$

The inequality uses the positive spectrum of $\mathbf{K}\mathbf{K}^\top$ on $\mathcal{S}$, which equals that of $\mathbf{G}$. Taking expectations in Equation (29) therefore gives a scalar recursion with contraction factor $c = 1 - (1-a^2)\lambda_{\min}(\mathbf{G})/(2K)$ and additive term $B = (1+a^2)\bar{n}^2/(1-a^2)$. Iteration gives $\mathbb{E}\|\mathbf{E}_n\|_F^2 \le c^n\|\mathbf{E}_0\|_F^2 + B(1-c^n)/(1-c)$, which implies Equation (26). This is the same key-span argument used in randomized projection analyses (Strohmer & Vershynin, 2009).

For cyclic visits, define the noise-free operator

$$\mathcal{L}_j(\mathbf{E}) = \mathbf{E}(\mathbf{I} - \mathbf{k}_j\mathbf{k}_j^\top) + \mathbf{T}_j\mathbf{E}\mathbf{k}_j\mathbf{k}_j^\top.$$

The exact energy identity implies

$$\|\mathcal{L}_j(\mathbf{E})\|_F^2 \le \|\mathbf{E}\|_F^2 - (1-a^2)\|\mathbf{E}\mathbf{k}_j\|^2.$$

Every visit is non-expansive. Equality of input and output norms requires $\mathbf{E}\mathbf{k}_j = \mathbf{0}$, in which case the visit leaves $\mathbf{E}$ unchanged. Equality over an entire cycle would require orthogonality to every key. For matrices whose rows lie in $\mathcal{S}$, this is possible only for $\mathbf{E} = \mathbf{0}$. Compactness of the unit sphere in this finite-dimensional subspace gives a one-cycle operator norm $\theta < 1$. The zero-dimensional case is immediate. The K disturbances propagated through non-expansive visits have total norm at most $K\bar{n}$. Thus

$$\|\mathbf{E}_{(\ell+1)K}\|_F \le \theta\|\mathbf{E}_{\ell K}\|_F + K\bar{n}.$$

Iteration proves Equation (27). □

Connection to re-entry. If the targets are $\hat{\mathbf{c}}_j^* = \mathbf{U}^\top\mathbf{y}_j^*$, then

$$\|\mathbf{U}\mathbf{M}_n\mathbf{k}_j - \mathbf{P}\mathbf{y}_j^*\| = \|\mathbf{E}_n\mathbf{k}_j\| \le \|\mathbf{E}_n\|_F.$$

Under these assumptions, part (a) controls the retrieval term of Theorem 1 in mean square; part (b) controls it at cycle boundaries.

Scope of the result. The assumed visit map is a condition on the induced memory dynamics. Proposition 2 bounds memory error using separate host and memory entry errors; it does not establish linearity or contraction of the visit map. The zero initialisation keeps the rows of $\mathbf{M}_n$ in the stationary-key span. An arbitrary initial component orthogonal to that span persists without affecting those reads, so convergence of the entire map to $\mathbf{M}^*$ need not hold. Stationary keys and targets, a fixed basis, fixed visit operators, and bounded disturbances need not hold in deployment. In particular, variable visit lengths can change the visit operator, and a cyclic schedule does not satisfy the sampling assumption in part (a). The result explains how key conditioning and visit quality can affect recall under explicit idealisations. It does not guarantee lower classification error or a universal minimum visit length. Controlled diagnostics in Appendix D.1 examine departures from these assumptions.

B.6 Representational capacity

Let $\mathbf{K} = [\mathbf{k}_1, \ldots, \mathbf{k}_N] \in \mathbb{R}^{m \times N}$ contain the keys and $\mathbf{V} = [\mathbf{v}_1^*, \ldots, \mathbf{v}_N^*] \in \mathbb{R}^{d \times N}$ the desired displacements. Write $\mathbf{P_K} = \mathbf{K}^\dagger \mathbf{K}$ for the orthogonal projector onto the row space of $\mathbf{K}$, where $\mathbf{K}^\dagger$ is the Moore–Penrose pseudoinverse.

Proposition 5 (Exact limit of a rank-r map)**.** *For $0 \le r \le \min(d, m)$,*

$$\min_{\mathrm{rank}(\mathbf{A}) \le r} \|\mathbf{AK} - \mathbf{V}\|_F^2 = \|\mathbf{V}(\mathbf{I} - \mathbf{P_K})\|_F^2 + \sum_{j>r} \sigma_j(\mathbf{VP_K})^2, \tag{30}$$

where the singular values σ_j are ordered from largest to smallest. For a fixed orthonormal basis $\mathbf{U}$,

$$\min_{\mathbf{M}} \|\mathbf{UMK} - \mathbf{V}\|_F^2 = \|(\mathbf{I} - \mathbf{UU}^\top)\mathbf{V}\|_F^2 + \|\mathbf{UU}^\top \mathbf{V}(\mathbf{I} - \mathbf{P_K})\|_F^2. \tag{31}$$

Proof. Every attainable output $\mathbf{Y} = \mathbf{AK}$ satisfies $\mathbf{YP_K} = \mathbf{Y}$. Orthogonal decomposition on the right therefore gives

$$\|\mathbf{AK} - \mathbf{V}\|_F^2 = \|\mathbf{V}(\mathbf{I} - \mathbf{P_K})\|_F^2 + \|\mathbf{AK} - \mathbf{VP_K}\|_F^2.$$

Because $\mathrm{rank}(\mathbf{AK}) \le r$, the second term is bounded below by the squared singular-value tail in Equation (30). Let $\mathbf{Y}_r$ be the rank-at-most-r truncated singular value decomposition of $\mathbf{VP_K}$. Its row space lies in that of $\mathbf{K}$, so $\mathbf{Y}_r\mathbf{P_K} = \mathbf{Y}_r$. The map $\mathbf{A} = \mathbf{Y}_r\mathbf{K}^\dagger$ satisfies $\mathbf{AK} = \mathbf{Y}_r$ and $\mathrm{rank}(\mathbf{A}) \le \mathrm{rank}(\mathbf{Y}_r) \le r$, attaining the bound.

For a fixed basis, orthogonal decomposition on the left gives

$$\|\mathbf{UMK} - \mathbf{V}\|_F^2 = \|(\mathbf{I} - \mathbf{UU}^\top)\mathbf{V}\|_F^2 + \|\mathbf{MK} - \mathbf{U}^\top\mathbf{V}\|_F^2.$$

The least-squares minimum of the second term is $\|\mathbf{U}^\top\mathbf{V}(\mathbf{I} - \mathbf{P_K})\|_F^2$, attained by $\mathbf{M} = \mathbf{U}^\top\mathbf{VK}^\dagger$. Orthonormality of $\mathbf{U}$ gives $\|\mathbf{U}^\top\mathbf{V}(\mathbf{I}-\mathbf{P_K})\|_F^2 = \|\mathbf{UU}^\top\mathbf{V}(\mathbf{I}-\mathbf{P_K})\|_F^2$, proving Equation (31). □

The first result separates incompatible linear relationships between keys and target displacements from insufficient map rank. Identical keys requiring different displacements are one example of the first limitation. With linearly independent keys, $\mathbf{P_K} = \mathbf{I}$, so this term vanishes even when the keys are poorly conditioned. The fixed-basis result additionally exposes directions missing from the chosen basis.

These are best-achievable representation errors with no bound on the map norm. They do not imply that online writes or append-only growth attain the optimum. Poorly conditioned independent keys can weaken the convergence bound in Proposition 4 and require sensitive maps. For independent keys, exact interpolation additionally requires $\mathrm{rank}(\mathbf{V}) \le r$; with a fixed basis, the targets must lie in its span.

C Implementation Details

Resource measurements. We profile segmentation resource costs using a SegFormer-B5 backbone on an NVIDIA RTX PRO 6000 GPU, and Cityscapes→ACDC inputs at 960×540 resolution with batch size one.

C.1 Memory settings

Table 10 lists all five numerical settings of the default memory procedure. The write rate controls consolidation and the rank cap limits the number of shared directions; the other three decide when a smoothed adjustment is added to the basis and are choices of the growth heuristic. These defaults are shared across hosts, backbones, datasets, and tasks; budget and sensitivity experiments vary the indicated settings. The sensitivity sweeps there vary the write rate, growth threshold, averaging factor, and rank cap with the minimum spacing held at ten batches; they measure how much the CIFAR-100-C results depend on these settings, not whether the settings are optimal on other backbones or datasets.

Table 10: **Default memory settings.** Budget and sensitivity experiments override the indicated setting; the host retains its own published configuration.

Setting	Value	Role
Write rate β	0.1	Fraction of coefficient error written per batch
Rank cap $r_{\max}$	64	Maximum number of shared directions
Growth threshold ϵ	0.25	Required magnitude relative to $\|\mathbf{y}_t\|$
Averaging factor λ	0.9	Smoothing of out-of-basis adjustments
Minimum spacing $n_{\min}$	10	Batches before the next possible growth event

C.2 STYLE DESCRIPTORS

Following ReservoirTTA (Vray et al., 2025), we use the ImageNet-pretrained VGG-19 encoder without batch normalisation, kept fixed throughout adaptation. Features at zero-based indices $[2, 5, 7]$ have 64, 128, and 128 channels, respectively, giving $m = 320$. Each descriptor coordinate is the logarithm of a channel variance pooled over the batch and spatial positions. Coordinate means and standard deviations are estimated once from 1,000 descriptors drawn under that protocol's 2,000 source-image allowance. The descriptor is standardised coordinate-wise and normalised to unit length. ReservoirTTA (Vray et al., 2025) and ENGRAM use the identical descriptor implementation, including its numerical floors and preprocessing. The segmentation configuration uses the descriptor of the current image, since it runs at batch size one.

C.3 HOST INTEGRATION AND OPTIMISATION

Our experiments use CNN and transformer backbones, adapting normalisation scales and biases, prompt tokens, or low-rank adapter parameters according to the host. The recalled initialisation is copied into the host's existing parameter tensors to preserve their optimiser associations. The host then completes its adaptation step, including any regularisation, sharpness-aware passes, restoration, or source-weight averaging, before memory records the updated parameters. Optimiser moments, teacher parameters, sample buffers, and filters follow the host's update rules and are not stored or restored separately for each domain. Each method retains the host's prediction model and timing, including teacher-based prediction where applicable, while using its own memory retrieval and parameter-combination procedure.

For recurring classification, host settings follow Vray et al. (2025, Table 10). CIFAR-10-C and CIFAR-100-C use batch size 200 and learning rate 10^{-3} with Adam, except for SAR's SGD optimiser. ImageNet-C with ResNet-50 uses SGD, learning rate 2.5×10^{-4}, and batch size 64. Source models follow the reference protocol and RobustBench (Croce et al., 2021).

For the additional protocols in Appendix E, ViT-B/16 uses SGD with learning rate 2.5×10^{-4} (10^{-4} for CCC and SAR) and batch size 64; DomainNet-126 uses SGD with learning rate 2.5×10^{-4} and batch size 128; and PACS uses Adam with learning rate 10^{-3} and batch size 64. The CCC stream follows Press et al. (2023).

LoRA adaptation. On Cityscapes→ACDC, SegFormer-B5 uses Adam and batch size 1. The LoRA rows use the BECoTTA configuration adopted by Vray et al. (2025), with learning rate 6×10^{-7} and stochastic restoration disabled; the other segmentation configurations use 7.5×10^{-6}. ReservoirTTA and ENGRAM use the same BECoTTA adaptation loss and adapter configuration, including adapter rank, target modules, scaling, initialisation, and dropout. The adapter rank r_{LoRA} is distinct from the memory rank cap $r_{\max}$, which limits the number of shared parameter-adjustment directions stored by ENGRAM.

Prompt adaptation. The VPT rows use ReservoirTTA's visual-prompt adaptation configuration for ViT-B/16, following the prompt parameterisation used by DPCore (Vray et al., 2025). Both memory methods retain this configuration's prompt count, placement, initialisation, trainable parameters, and adaptation objective without modification. The corresponding host prediction model and timing are also retained.

C.4 State and computation

The explicit-residual implementation stores $\mathbf{U}$ (dr values), $\mathbf{M}$ (rm), the residual and growth average ($2d$), and a saved key (m). Three logical int64 counters contribute 24 bytes. With b bytes per floating-point value, the counted state occupies

$$b\,[r(d+m)+2d+m]+24$$

bytes. Here $b = 4$ for float32 and $b = 8$ for float64; mixed-precision allocations are counted separately. At the same precision, this is smaller than K stored d-dimensional parameter vectors exactly when

$$b\,[r(d+m)+2d+m]+24 < bKd.$$

For $m = d$, the condition reduces to $r < (K-3)/2 - 12/(bd)$. Storage therefore depends on the coefficient map and working state as well as the basis. The rank and pool-size settings in Table 13 specify capacity limits.

These counts include array payload and logical counters. They exclude the source and working backbones, host state (including sample buffers), the fixed style encoder and source calibration, routing metadata, temporary workspace, and object overhead. ENGRAM adds no exemplar buffer.

Excluding the host and style encoder, memory operations require $\mathcal{O}(r(d+m)+d)$ arithmetic per batch, including residual updates and growth statistics. Appending a direction requires $\mathcal{O}(dr)$ orthogonalisation. The style encoder adds one forward pass per batch.

D Benchmark Diagnostics and Additional Results

D.1 Retention and recall on recurring corruptions

We use CIFAR-100-C to examine how the memory operations affect recall, interference, and adaptation. The primary diagnostic combines the ResNeXt-29 backbone with EATA and repeats the 15 severity-5 corruptions for 20 recurrences. Methods receive identical batches within each run. These experiments complement the benchmark comparison in Table 1; the diagnostic stream and seed assignments are specified below.

Recall improves across visits. Figure 8 separates first-batch re-entry error from visit-average error and shows the five seed trajectories. At recurrence 20, EATA has a re-entry error of 35.9%, compared with 30.9% for ReservoirTTA and 27.9% for ENGRAM. Their corresponding visit-average errors are 30.5%, 28.4%, and 27.5%. Thus, the benefit extends to the first prediction after a domain returns. The paired error differences are reported in Table 14; Table 15 provides analogous comparisons on CIFAR-10-C and ImageNet-C.

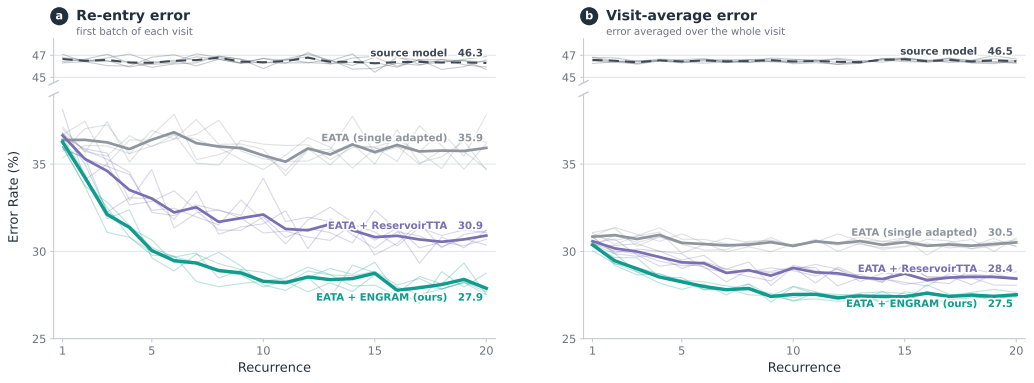


Figure 8: **Recall on CIFAR-100-C.** EATA with ResNeXt-29 under recurring severity-5 corruptions: (a) first-batch re-entry error and (b) visit-average error. Thin lines show five paired seeds; bold lines show their means.

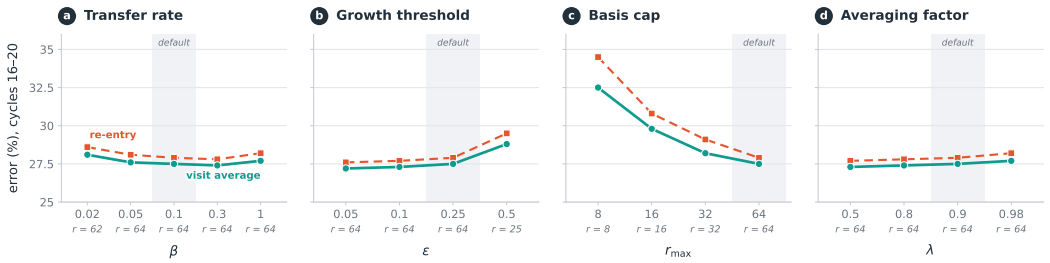


Figure 9: **Memory-setting sensitivity.** CIFAR-100-C, two paired seeds; errors average recurrences 16–20. Four settings vary; growth spacing remains ten batches. Grey annotations give mean learned rank.

Recall depends on keys and capacity. Table 11 varies the rank cap and address geometry. Increasing capacity improves recall with orthogonal or natural VGG keys, but highly correlated keys remain a limitation even at the larger cap. Figure 12 examines how finite capacity affects later-arriving domains. The budget comparison in Table 13 and the separate key intervention in Appendix G.1 distinguish these two limitations.

D.2 Component ablations and sensitivity

Table 12 evaluates the memory components on CIFAR-100-C. Removing residual compensation raises re-entry error from 27.9% to 34.6%. An additive full-target write raises it to 88.4%: repeatedly adding completed displacements accumulates stored values instead of correcting the current retrieval error. Freezing writes after the first recurrence gives 31.8%, showing the value of continued consolidation. Eliminating residual carry-over gives 32.4%, whereas immediate transfer ($\beta = 1$) gives 28.2%. The comparison of append-only growth and least-energy replacement also favours preserving existing directions at the tested budgets.

Global memory decay increases re-entry error after a domain is withheld (Figure 4). The key-noise experiment in Figure 11 favours full carry-over over similarity gating in the tested streams. Figure 9 varies the four memory settings: changing β from 0.02 to 1 changes late-stream error by less than one percentage point, while the growth threshold and rank cap affect both error and the number of stored directions.

D.3 Diagnostic protocol

Data, host, and stream. The source model is a ResNeXt-29 trained on clean CIFAR-100. The adapted vector contains the affine scale and shift parameters of its normalisation layers. Keys use the fixed pre-trained VGG-19 encoder and source calibration described in Appendix C. The EATA host uses unlabelled test batches, reliable-sample filtering, and its source-anchored regularisation; labels are used only to score predictions. For these diagnostics, recurring CSC uses 50 batches per corruption and 20 recurrences, giving 15,000 adaptation steps at batch size 64. Recurring CDC samples corruption identities uniformly and visit lengths uniformly from 25 to 75 batches, for 300 visits. Unless a caption specifies another endpoint, results are evaluated at recurrence 20. Sensitivity and key-noise curves average recurrences 16–20; Table 11 averages re-entry errors from recurrences 18–20.

Paired runs. The main diagnostic comparisons, component ablations, additional benchmarks, and weight-ensembled-host experiment use seeds $\{0, \ldots, 4\}$. Carry-policy curves and sensitivity sweeps use the two paired seeds $\{0, 1\}$. Seeds control within-domain sampling and, for CDC, domain order and duration. Within a seed, methods receive identical streams. Standard deviations use denominator $n - 1$; confidence intervals are formed from per-seed paired differences. Batch-level observations are not treated as independent replicates. These repeats measure sensitivity to stream sampling; the comparisons across architectures and datasets appear in Table 15.

Storage comparisons. The pool in Table 13 is limited to K adapted parameter states, whereas ENGRAM is limited to $r_{\max}$ shared directions. The pool's parameter stores require Kd scalars; the memory factors require $r_{\max}(d+m)$, in addition to the working state detailed in Appendix C. Setting $K = r_{\max}$ therefore compares similar nominal capacities when $d \gg m$, but does not by itself impose identical byte budgets. The table should be interpreted as a capacity comparison. The oracle pool in Table 11 receives domain identities and serves as a reference with privileged routing information.

Table 11: **Capacity and addressing.** CIFAR-100-C re-entry error (%), recurrences 18–20. The oracle pool uses domain identities. The 15 terminal reference displacements have numerical rank 15, not the trajectory's rank. "Captured" averages their individually normalised energy in the learned basis. Norm, coverage, and final rank refer to $r_{\max} = 8$.

			pool	ENGRAM re-entry (%)		ENGRAM, $r_{\max}$=8		
Keys	Condition	Reference rank	re-entry	$r_{\max}$=8	$r_{\max}$=64	$\|\mathbf{A}\|_2$	captured (%)	Final Rank
Oracle (One-Hot)	Orthogonal	15	26.5	**33.8**	**27.0**	15.2	53.3	8
VGG Descriptors	Natural Overlap	15	26.5	34.2	27.9	12.4	51.0	8
Collinear (Blended)	Highly Correlated	15	26.5	38.5	36.8	7.6	32.1	8

D.4 RETENTION, ADDRESS GEOMETRY, AND CAPACITY

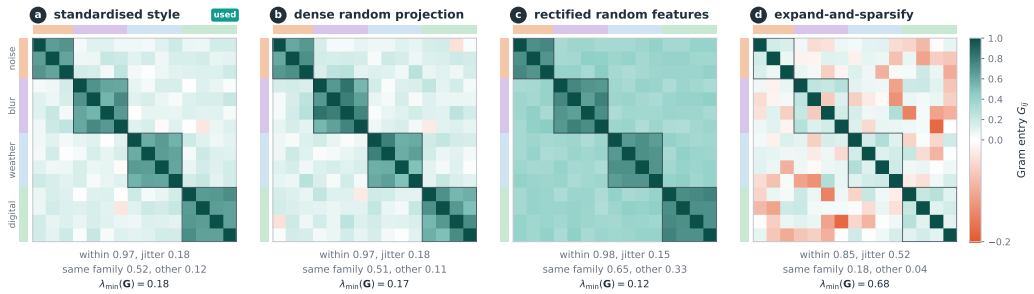


Figure 10: **Geometry of CIFAR-100-C keys.** Gram matrices of the 15 corruption keys under four encodings, grouped by corruption family. Summaries report within-domain cosine similarity, consecutive-batch jitter, within- and across-family overlap, and the minimum Gram eigenvalue. The encodings trade address separation against consistency across batches; VGG descriptors are used in the main evaluation.

Figure 4 measures return error after withholding a corruption. The default avoids decay applied to the entire memory; interference from writes with overlapping keys remains possible (Proposition 3). Table 12 reports the component comparisons, and Figure 10 measures the overlap and batch-to-batch variation of alternative key encodings.

Table 12: **Memory components and basis growth.** CIFAR-100-C, ResNeXt-29, five seeds; error (%) and learned rank r at recurrence 20. Blue identifies the complete method or append-only growth. Bold compares errors within each rank-budget block. † uses domain boundaries.

Variant	re-entry (%)	visit-average (%)	r
ENGRAM (default)	**27.9**	**27.5**	64
no residual compensation	34.6	32.8	42
full-target additive write	88.4	85.2	8
global decay γ=0.999	33.2	30.5	64
writes frozen after cycle 1	31.8	29.4	48
dense random keys	28.4	27.9	64
rectified random keys	28.5	28.0	56
β=1.0 (immediate transfer)	28.2	27.7	64
similarity gate $h = \max(0, \langle \mathbf{k}_{t-1}, \mathbf{k}_t \rangle)^2$	28.6	28.1	45
no carry-over ($h = 0$)	32.4	30.1	18
oracle residual reset at switches†	28.3	27.7	47
budget $r_{\max}$=16, append-only	**30.8**	**29.8**	16
budget $r_{\max}$=16, least-energy replacement	33.4	32.1	16
budget $r_{\max}$=32, append-only	**29.1**	**28.2**	32
budget $r_{\max}$=32, least-energy replacement	30.5	29.4	32
budget $r_{\max}$=64, append-only (default)	**27.9**	**27.5**	64
budget $r_{\max}$=64, least-energy replacement	28.1	27.7	64

Figure 9 reports sensitivity to the transfer rate, growth threshold, averaging factor, and rank cap. Table 13 compares the pool and shared basis under increasing nominal capacities.

Table 13: **Comparison across capacity limits.** CIFAR-100-C, ResNeXt-29, 15 corruptions and 20 recurrences; visit-average error (%) at recurrence 20. The pool stores at most K adapted states, and ENGRAM learns at most $r_{\max}$ directions. Storage accounting is given in Appendix C.

Nominal capacity K or $r_{\max}$	4	8	12	16
ReservoirTTA (K states)	**29.8**	**29.1**	28.6	**28.4**
ENGRAM ($r_{\max}$ directions)	31.5	29.8	**28.2**	29.8

D.5 PAIRED STATISTICS AND ADDITIONAL BENCHMARKS

Table 14 reports paired differences across five runs, with negative values favouring ENGRAM. The confidence intervals use the Student-t distribution with four degrees of freedom. Table 15 extends the re-entry and visit-average comparison to CIFAR-10-C and ImageNet-C.

Table 14: **Paired differences at recurrence 20.** Five seeds; mean $\pm$ sample standard deviation of the error difference (percentage points), pointwise 95% Student-t confidence interval, and t statistic. Negative favours ENGRAM.

Setting	Comparison	Δ (points)	95% CI	t
CIFAR-100-C, re-entry	ENGRAM − ETA (single model)	-9.49 ± 1.12	$[-10.88, -8.10]$	-19.0
CIFAR-100-C, re-entry	ENGRAM − ReservoirTTA	-3.02 ± 0.57	$[-3.73, -2.32]$	-11.9
CIFAR-100-C, visit average	ENGRAM − ETA (single model)	-9.67 ± 0.28	$[-10.02, -9.33]$	-77.8
CIFAR-100-C, visit average	ENGRAM − ReservoirTTA	-0.92 ± 0.33	$[-1.34, -0.51]$	-6.2
weight-ensembled host, re-entry	ENGRAM − EATA (single model)	-8.05 ± 1.65	$[-10.10, -6.00]$	-10.9
weight-ensembled host, visit average	ENGRAM − EATA (single model)	-2.99 ± 0.30	$[-3.37, -2.62]$	-22.0

Table 15: **Recall on two further benchmarks.** Five seeds; classification error (%) at recurrence 20. The single model uses EATA. Re-entry is the first-batch error; average is the visit-average error. Source errors are 43.5% on CIFAR-10-C and 82.0% on ImageNet-C.

	CIFAR-10-C / WRN-28		ImageNet-C / ResNet-50	
Method	Re-entry	Average	Re-entry	Average
EATA	20.1	17.8	61.2	55.9
+ReservoirTTA	17.6	16.4	54.1	51.0
+ENGRAM	**16.2**	**15.9**	**50.3**	**49.5**

D.6 RESIDUAL CARRY AND LEARNED DIRECTIONS

Figure 11 compares three carry policies: full carry-over, similarity gating, and an oracle that resets the residual at every domain switch. In the first comparison, each policy grows its own basis. In the second, all policies share the terminal basis from the full-carry run with seed 0. This is an offline control because the basis has seen the complete stream.

With the shared basis, full carry and oracle reset reach similar errors. This suggests that differences in learned directions contribute to their gap when bases grow independently. Similarity gating shows the largest increase in error with descriptor noise, consistent with its attenuation of residual carry whenever consecutive keys differ, even within one domain.

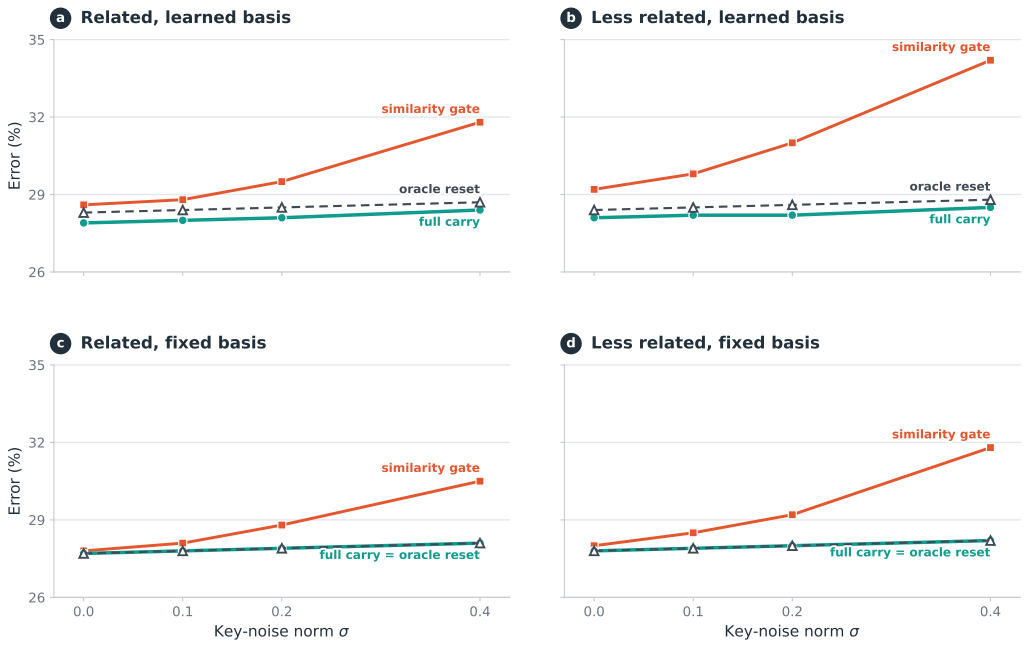


Figure 11: **Carry policies under key noise.** CIFAR-100-C, two paired streams and identical rank caps; σ is the norm of additive noise injected before key normalisation. (a,b) Each policy learns its own basis. (c,d) Policies use the terminal full-carry basis from seed 0, held fixed as an offline representation control. Curves show mean error over recurrences 16–20.



Figure 12: **Basis capacity and later-arriving domains.** CIFAR-100-C, 15 corruptions. (a) Singular-energy spectrum of terminal oracle-pool displacements. (b,e) Learned rank and mean reference energy captured. (d) Reference energy captured by leading singular vectors. (c,f) Re-entry error when seven corruptions join eight initial corruptions after five recurrences, with $r_{\max} = 8$ or 64. The larger cap allows additional directions to be retained.

Figure 12 compares the learned basis with the terminal states of an oracle pool. The singular spectrum of those states describes only where adaptation ends, whereas the growing basis can also contain directions that were used transiently along the way. In the late-arrival experiment, seven corruptions join the stream after five recurrences of the first eight. Once the rank cap is reached, a new displacement is recalled only to the extent that existing directions span it; its out-of-basis part stays in the working residual, available for the current visit but not retrievable from memory.

Contraction relative to empirical references. Figure 13 compares recalled adjustments with projected reference displacements, obtained by running the weight-ensembled host for 3,000 steps on each corruption. It also reports the retrieval gap, the distance between what memory recalls for a domain and what the host had reached at the end of its previous visit. The dashed curve illustrates the independent-uniform rate in Proposition 4. It is not a bound for these runs: keys vary, the basis grows, and visits follow a cyclic order.

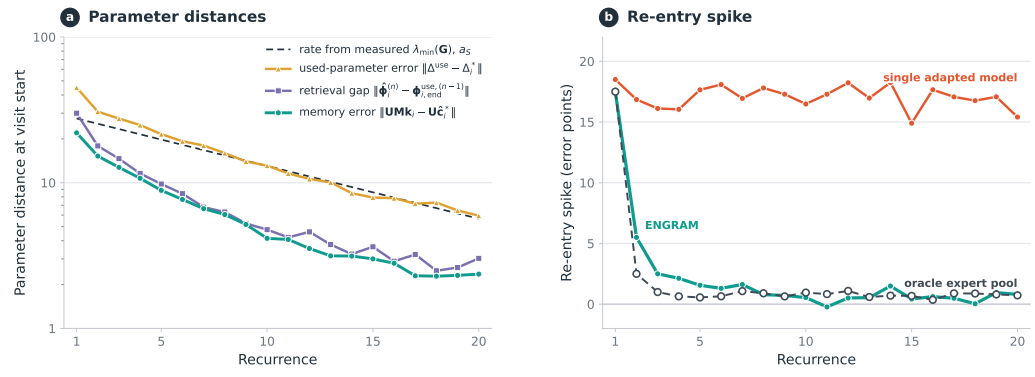


Figure 13: **Memory distances across recurrences on CIFAR-100-C.** Reference displacements $\mathbf{y}_i^*$ are obtained from a long-running weight-ensembled host. (a) Stored-read distance to $\mathbf{P}\mathbf{y}_i^*$, retrieval gap relative to the preceding visit, and active-model distance to $\mathbf{y}_i^*$. (b) Re-entry error minus visit-average error.

D.7 WEIGHT-ENSEMBLED HOST

To test consolidation with source anchoring, we apply $\phi \leftarrow 0.99\phi + 0.01\phi_{\text{src}}$ after each host gradient step. Figure 14 reports the CIFAR-100-C trajectories. At recurrence 20, the single model has visit-average error 31.0%, the oracle pool 27.9%, and ENGRAM 28.0%. The pool and ENGRAM both have re-entry error 28.5%. These results show that useful recall remains possible with this anchoring update. The memory consolidates the host's anchored state; it does not remove the anchor or recover an unregularised optimum.

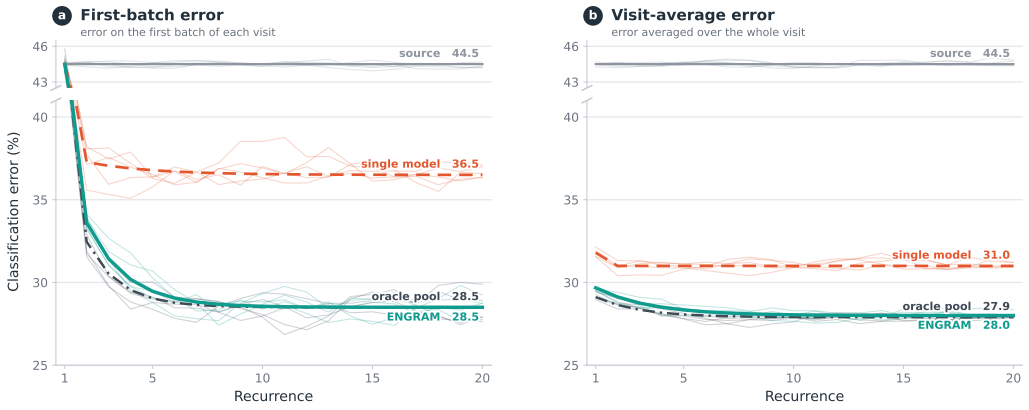


Figure 14: **Recall with a weight-ensembled host.** CIFAR-100-C, with a 0.01 interpolation towards source parameters after each gradient step. Thin lines show five paired seeds; bold lines show their means. Consolidation retains the completed anchored update, as characterised for a fixed key by Proposition 2.

D.8 ADDITIONAL MECHANISM DIAGNOSTICS

Conservation. We verified Proposition 1 on ResNeXt-29. After every write, the reconstruction $\phi_{\text{src}} + \mathbf{A}_t\mathbf{k}_t + \boldsymbol{\delta}_t$ matches the completed host parameters to numerical precision, and with a constant key and matched host state the next initialisation reproduces the host's own trajectory. With batch-dependent VGG keys the next initialisation can differ, because the key has moved, even though the reconstruction remains exact. Figure 10 measures how much the keys move and Figure 13 how far the recalled parameters are from their references; neither by itself establishes long-horizon prediction stability.

Absence and return. The withheld-domain experiment in Figure 4 extends to eight cycles. The default returns at 27.9% error at both zero and eight cycles of absence. With global decay, return error at eight cycles rises to 31.2% for $\gamma = 0.9995$ and 33.2% for $\gamma = 0.999$. Avoiding global decay removes this source of forgetting; it does not make reads invariant to writes with correlated keys.

Address quality and representation. Table 11 and Figure 12 examine two limits: overlapping keys can interfere, and a finite basis can omit directions needed by a returning condition. A larger basis can represent more directions but does not guarantee accurate recall from overlapping keys. Appendix G.1 varies key geometry, and Appendix G.2 tests carry under the main CIFAR-100-C configuration.

E ADDITIONAL BENCHMARK RESULTS

Tables 16 and 17 extend the evaluation to a vision transformer and continuously changing corruptions. Tables 18 and 23 report object-level style shifts, and Table 19 gives the segmentation results. The numerical baseline rows are retained at their published precision; new ENGRAM rows report experimental measurements. Allocation formulas in Appendix C distinguish memory factors, working state, and the full system footprint.

Table 16: **ImageNet-C with ViT-B/16.** Error (%) at recurrences 1 and 20 for CSC/CDC and at steps 6.7k, 40.2k, and 80k for CCC. Time is relative to source inference. Adaptive-state accounting is given in Appendix C.

	Recurring CSC			Recurring CDC			CCC			Time
Method	1	20	Δ	1	20	Δ	6.7k	40.2k	80k	
Source	48.8	48.8	0.0	48.8	48.8	0.0	51.9	49.3	50.7	1.0
ETA	38.9	48.4	+9.5	42.9	35.9	−7.0	42.7	40.6	40.8	2.0
+CoLA (Chen et al., 2024)	41.0	33.8	−7.2	40.9	34.9	−6.0	44.8	40.5	40.2	2.0
+ReservoirTTA	39.4	31.9	−7.5	41.6	32.9	−8.7	44.1	40.2	38.5	3.0
+ENGRAM	39.4	**30.4**	−9.0	41.6	**31.4**	−10.2	43.1	**39.0**	**37.0**	3.0
DPCore	40.2	46.0	+5.8	42.8	47.6	+4.8	42.2	42.3	43.1	3.8
VPT+ReservoirTTA	**38.0**	33.7	−4.3	**38.0**	34.3	−3.7	42.9	40.2	39.2	3.0
VPT+ENGRAM	**38.0**	32.2	−5.8	**38.0**	32.8	−5.2	**42.0**	39.1	37.7	3.0

Table 17: **Continuously changing ImageNet-C with ResNet-50.** Error (%) at the indicated steps and averaged over the stream.

Method	6.7k	40.2k	80k	Avg
Source	83.1	83.6	83.3	83.2
ETA	64.7	73.0	78.7	72.7
+ReservoirTTA	63.1	60.5	61.9	61.6
+ENGRAM	61.6	59.0	60.3	60.1
RDumb	62.3	58.4	59.8	59.8
PeTTA	68.9	62.6	63.8	63.6
EATA	60.9	58.1	60.2	59.5
+ReservoirTTA	61.7	57.0	57.9	58.4
+ENGRAM	60.2	55.4	**56.4**	**56.8**
ROID*	61.9	57.0	58.8	58.8
+ReservoirTTA	61.2	56.7	58.5	58.5
+ENGRAM	**59.7**	**55.2**	56.9	56.9

Table 18: **Object-level style shifts.** Mean error (%) over 20 recurring cycles and all source-domain splits; per-source results in Table 23, gradual and class-correlated streams in Table 24.

Method	DomainNet-126	PACS
Source	45.40	46.24
EATA	35.92	29.38
+ReservoirTTA	35.13	28.68
+ENGRAM	**33.65**	**27.20**

Table 19: **Complete Cityscapes→ACDC results.** SegFormer-B5 mIoU (%, ↑) at recurrences 1 and 10 of Fog→Night→Rain→Snow, with batch one. Bold marks column maxima.

	Fog		Night		Rain		Snow	
Method	1	10	1	10	1	10	1	10
Source	69.1	69.1	40.3	40.3	59.7	59.7	57.8	57.8
TENT	69.0	65.3	40.2	34.1	60.0	56.7	57.5	52.0
+ReservoirTTA	69.1	67.6	40.2	38.5	59.9	58.0	57.6	54.7
+ENGRAM	69.4	68.8	40.5	39.8	60.3	59.5	58.0	56.4
CoTTA*	71.5	71.2	41.4	39.9	62.7	63.3	59.9	59.8
+ReservoirTTA	72.9	72.7	41.1	40.2	64.4	64.7	60.2	60.0
+ENGRAM	**73.1**	**74.9**	41.5	42.0	**65.0**	66.6	60.8	61.8
BECoTTA	72.0	72.5	41.2	40.8	63.4	64.8	60.3	60.3
LoRA+ReservoirTTA	72.7	72.4	41.5	41.4	64.3	64.9	61.0	61.2
LoRA+ENGRAM	73.0	**74.9**	**41.8**	**43.7**	64.9	**67.9**	**61.3**	**64.2**

Segmentation component ablations. We evaluate the memory mechanisms using the LoRA adaptation configuration of Table 19, with ten recurrences of Fog→Night→Rain→Snow. Visit-10 mIoU averages the four condition-level mIoUs at the final recurrence; stream-average mIoU averages all 40 condition-visits. For early-return mIoU, we accumulate a confusion matrix over the first ten images of each condition during recurrences 2–10, compute mIoU separately for each of the 36 windows, and average across windows. This metric measures performance during early adaptation after a condition returns.

We change one memory mechanism at a time. Without compensation, the pre-write mismatch is retained as the residual after the memory write. Without residual carry, the residual is omitted from the next initialization but is still computed for consolidation and basis growth. Global decay attenuates the coefficient map by $\gamma = 0.999$ per batch while retaining compensation. Fixing memory after recurrence 1 stops both coefficient writes and basis growth after the first complete cycle; host adaptation and residual recomputation continue.

Table 20: **Full segmentation ablations.** LoRA/SegFormer-B5 on ACDC, 960×540 inputs, batch one, ten recurrences, five paired seeds. Values are mean mIoU (%, ↑); final rank is mean ± SD.

	mIoU (%, ↑)			
Variant	Visit 10	Stream avg.	Early return	Final rank r
LoRA host alone	58.6	57.8	56.5	–
LoRA + ReservoirTTA	60.0	59.1	59.4	–
Without residual compensation	57.2	56.8	56.2	58.4 ± 3.2
Without residual carry	60.8	59.4	59.6	28.6 ± 5.1
Global memory decay ($\gamma = 0.999$)	61.1	60.1	59.2	45.8 ± 3.9
Memory fixed after recurrence 1	60.4	59.5	60.1	16.0 ± 0.0
LoRA + ENGRAM	**62.7**	**61.2**	**61.8**	45.2 ± 4.1

Table 20 shows that the complete method achieves the highest mIoU on all three metrics. Relative to LoRA + ReservoirTTA, it improves final, stream-average, and early-return mIoU by 2.7, 2.1, and 2.4 points, respectively. Removing compensation causes the largest final-performance loss (5.5 points), despite producing a larger basis. Removing residual carry reduces final mIoU by 1.9 points, supporting the role of adjustments not yet captured by the memory read.

Global decay reduces early-return mIoU from 61.8% to 59.2%, although its final rank is similar to that of the complete method. This comparison supports retaining stored adjustments between visits. Fixing memory after the first recurrence reduces final mIoU by 2.3 points, demonstrating the benefit of continued memory learning. Because this intervention stops both coefficient writes and basis growth, it assesses their combined contribution.

Table 21: Baseline classification error (%) at the first and final recurrence, averaged over five seeds. Corresponding ENGRAM endpoints are reported in Table 1.

	Recurring CSC			Recurring CDC		
Method	first	final	mean	first	final	mean
CIFAR-10-C / WideResNet-28						
Source	43.52	43.52	43.52	43.52	43.52	43.52
TENT	19.28	87.84	67.37	20.49	86.95	69.30
TENT + ReservoirTTA	18.28	17.56	17.55	18.15	17.37	17.36
CoTTA*	18.75	22.40	19.86	18.75	22.29	19.80
RoTTA	19.35	18.35	17.79	21.86	20.38	19.92
ETA	17.78	30.89	25.15	17.90	33.49	25.80
ETA + ReservoirTTA	17.49	**16.41**	**16.47**	**17.40**	**16.27**	**16.30**
SAR	20.39	20.39	20.38	20.41	20.39	20.41
SAR + ReservoirTTA	20.35	20.36	20.35	20.38	20.35	20.35
RDumb	17.78	18.41	17.97	17.90	18.12	18.03
PeTTA	22.98	17.20	17.82	27.16	20.81	21.70
EATA	**17.46**	17.76	17.68	17.70	17.88	17.87
EATA + ReservoirTTA	17.53	16.44	16.54	17.46	16.39	16.46
ROID*	17.75	17.65	17.58	18.04	18.07	17.98
ROID* + ReservoirTTA	17.75	16.79	16.88	17.86	16.78	16.87
CIFAR-100-C / ResNeXt-29						
Source	46.45	46.45	46.45	46.45	46.45	46.45
TENT	61.41	99.00	96.90	60.21	98.87	96.61
TENT + ReservoirTTA	38.06	43.97	42.67	33.90	39.73	38.02
CoTTA*	35.10	65.46	49.71	35.09	65.13	49.46
RoTTA	34.80	59.10	45.95	36.82	73.82	54.36
ETA	31.95	37.64	35.77	32.36	37.62	35.81
ETA + ReservoirTTA	31.59	30.02	29.94	30.87	29.57	29.52
SAR	31.92	60.36	44.16	31.64	57.81	43.36
SAR + ReservoirTTA	31.91	30.48	30.37	31.72	29.79	29.85
RDumb	31.95	32.86	32.17	32.36	32.63	32.41
PeTTA	39.44	32.91	33.21	42.13	35.34	35.49
EATA	30.51	30.47	30.46	31.01	31.08	30.98
EATA + ReservoirTTA	30.56	28.38	28.57	30.44	28.40	28.56
ROID*	**29.45**	29.25	29.25	30.17	30.12	30.00
ROID* + ReservoirTTA	29.60	**27.80**	**27.92**	**29.62**	**27.78**	**27.95**
ImageNet-C / ResNet-50-BN						
Source	82.03	82.03	82.03	82.03	82.03	82.03
TENT	62.60	99.48	90.09	61.98	99.54	89.66
TENT + ReservoirTTA	62.60	58.22	57.76	62.43	57.52	57.20
CoTTA*	67.58	61.68	62.64	67.73	61.50	62.50
RoTTA	67.27	99.37	85.74	71.61	99.51	88.90
ETA	60.00	59.40	58.86	59.33	60.07	59.22
ETA + ReservoirTTA	59.77	53.10	53.90	58.63	52.18	52.91
SAR	61.87	67.06	61.73	61.48	66.17	61.48
SAR + ReservoirTTA	62.21	53.13	55.15	62.63	53.56	55.50
RDumb	59.82	56.78	58.14	59.57	59.47	59.33
PeTTA	67.50	60.11	62.18	71.60	69.52	68.17
EATA	57.52	55.94	55.77	58.46	57.02	56.95
EATA + ReservoirTTA	58.03	**51.00**	**51.99**	58.53	**51.81**	**52.63**
ROID*	**56.07**	55.48	55.45	58.70	58.25	58.33
ROID* + ReservoirTTA	56.42	52.07	52.54	**56.95**	53.01	53.28

Table 22: **Additional ImageNet-C backbones.** Error (%) at the first and final recurrence and mean over 20 recurrences for ResNet-50-GN and ViT-B/16.

	Recurring CSC			Recurring CDC		
Method	first	final	mean	first	final	mean
ImageNet-C / ResNet-50-GN						
Source	72.55	72.55	72.55	72.55	72.55	72.55
TENT	71.51	99.11	91.17	75.13	99.58	96.16
TENT + ReservoirTTA	70.87	78.97	77.07	71.79	84.61	81.64
CoTTA*	72.81	74.26	73.31	72.88	75.68	73.96
ETA	65.58	61.44	62.14	66.95	61.90	61.89
ETA + ReservoirTTA	65.61	57.29	58.82	66.00	57.11	58.68
ETA + ENGRAM	65.61	55.79	57.32	66.00	55.61	57.18
SAR	67.92	78.91	67.58	68.41	63.52	65.66
SAR + ReservoirTTA	68.13	63.40	62.64	70.75	65.79	65.96
SAR + ENGRAM	68.13	61.90	61.14	70.75	64.29	64.46
RDumb	65.38	66.08	66.10	67.36	66.41	66.91
EATA	62.90	59.85	60.55	64.57	59.98	60.78
EATA + ReservoirTTA	63.41	55.22	56.10	66.05	54.78	56.30
EATA + ENGRAM	63.41	**53.72**	**54.60**	66.05	**53.28**	**54.80**
ROID*	62.39	62.84	62.59	66.63	65.91	66.18
ROID* + ReservoirTTA	**62.37**	57.77	58.08	**63.41**	58.36	58.74
ROID* + ENGRAM	**62.37**	56.27	56.58	**63.41**	56.86	57.24
ImageNet-C / ViT-B/16						
Source	48.75	48.75	48.75	48.75	48.75	48.75
TENT	42.35	37.64	37.96	44.21	38.68	39.05
TENT + ReservoirTTA	42.87	35.04	37.08	46.64	40.19	41.82
CoTTA*	48.55	46.46	46.64	48.37	45.94	46.15
ETA	38.91	48.43	38.71	42.89	35.87	36.66
ETA + CoLA	40.98	33.82	35.04	40.90	34.87	35.73
ETA + ReservoirTTA	39.40	31.86	33.10	41.55	32.92	34.45
ETA + ENGRAM	39.40	**30.36**	**31.60**	41.55	**31.42**	**32.95**
SAR	44.52	39.50	40.15	45.86	39.57	40.32
SAR + ReservoirTTA	43.44	36.33	38.26	45.47	37.91	39.75
SAR + ENGRAM	43.44	34.83	36.76	45.47	36.41	38.25
RDumb	39.32	38.68	39.28	43.91	39.20	39.81
EATA	39.11	34.60	35.19	39.89	34.53	35.33
EATA + ReservoirTTA	39.76	32.24	33.81	42.08	34.19	35.84
EATA + ENGRAM	39.76	30.74	32.31	42.08	32.69	34.34
ROID*	40.14	40.03	40.08	42.09	41.12	42.46
ROID* + ReservoirTTA	40.09	37.77	37.98	40.35	37.82	38.03
ROID* + ENGRAM	40.09	36.27	36.48	40.35	36.32	36.53
DPCore	40.21	45.95	44.73	42.79	47.58	45.22
VPT + ReservoirTTA	**37.98**	33.72	34.61	**38.01**	34.32	35.11
VPT + ENGRAM	**37.98**	32.22	33.11	**38.01**	32.82	33.61

Table 23: **Object-level shifts by source domain.** Error (%) with a ResNet-50 trained on the indicated source and adapted as the other three domains recur for 20 cycles. Means average the four source splits.

DomainNet-126 Method	Real	Painting	Clipart	Sketch	Mean
Source	45.16	41.57	49.52	45.33	45.40
EATA	38.49	33.49	38.34	33.34	35.92
EATA + ReservoirTTA	36.89	32.25	38.65	32.74	35.13
EATA + ENGRAM	**35.41**	**30.77**	**37.17**	**31.26**	**33.65**
PACS Method	Photo	Art painting	Cartoon	Sketch	Mean
Source	56.97	32.21	23.71	72.06	46.24
EATA	38.22	22.80	18.82	37.68	29.38
EATA + ReservoirTTA	37.79	22.55	18.63	35.75	28.68
EATA + ENGRAM	**36.31**	**21.07**	**17.15**	**34.27**	**27.20**

Table 25: **Transfer to unseen ImageNet-C corruptions.** Severity-5 error (%, ↓), ViT-base, after adaptation to five corruptions and read-only evaluation on ten others. ENGRAM is the reported five-seed result. Gain is the reduction from Source in points. Best: bold; second: underlined.

	Unseen corruptions											
Method	Motion	Zoom	Snow	Frost	Fog	Brightness	Contrast	Elastic	Pixelate	JPEG	Mean↓	Gain↑
Source	58.5	63.3	49.9	54.2	57.7	26.4	91.4	57.5	38.0	36.2	53.3	0.0
TENT (Wang et al., 2021)	56.0	61.3	45.7	49.6	56.6	24.8	94.0	55.6	37.1	35.1	51.6	+1.7
CoTTA (Wang et al., 2022)	57.3	62.1	49.1	52.0	57.1	26.4	91.9	57.1	37.6	35.3	52.6	+0.7
ViDA (Liu et al., 2024)	46.4	52.7	39.8	43.7	42.2	23.5	71.5	49.6	33.9	33.3	43.7	+9.6
MoASE (Zhang et al., 2026a)	43.9	53.7	38.0	37.6	46.0	24.0	66.0	42.8	28.9	29.6	41.0	+12.3
MoASE++ (Zhang et al., 2026b)	42.4	52.1	37.5	39.1	42.6	22.6	63.3	42.9	28.2	27.3	39.8	+13.5
EATA + ENGRAM	**41.0**	**50.6**	**36.3**	**37.2**	**41.2**	**21.9**	**61.9**	**41.4**	**27.2**	**26.3**	**38.5**	**+14.8**

Table 24: **Gradual severity and temporal correlation.** Classification error (%) at the first and final recurrence and averaged over the stream. Severity cycles $1 \to 5 \to 1$ on CIFAR-100-C with ResNeXt-29. Correlated CSC uses CIFAR-10-C, WRN-28, a class-Dirichlet coefficient of 0.1, and the RoTTA host.

Stream	Method	first	final	mean
Gradual CIFAR-100-C	Source	33.58	33.58	33.58
	EATA	25.35	25.40	25.41
	EATA + ReservoirTTA ($K^{\max}$=16)	**25.17**	25.07	25.06
	EATA + ReservoirTTA ($K^{\max}$=64)	**25.17**	24.96	24.95
	EATA + ENGRAM	**25.17**	**23.48**	**23.47**
Correlated CIFAR-10-C	Source	43.52	43.52	43.52
	RoTTA	26.62	38.87	30.37
	RoTTA + ReservoirTTA	**26.19**	25.67	25.89
	RoTTA + ENGRAM	**26.19**	**24.17**	**24.39**

Additional backbones and stress settings. Table 21 reports the reference baseline results at their original precision; Table 22 extends the comparison to ResNet-50-GN and ViT-B/16. ENGRAM lowers final error relative to the matched expert pools in these additional comparisons. Under CDC with ResNet-50-GN, SAR achieves 63.52% error, compared with 64.29% for SAR + ENGRAM.

Under gradual severity, EATA + ENGRAM reaches 23.48% final error, compared with 24.96% for the pool with cap 64. On the temporally correlated stream, RoTTA + ENGRAM reaches 24.17%, compared with 25.67% for RoTTA + ReservoirTTA (Table 24). The host's own sample buffer and teacher remain part of its transition.

For segmentation, the adapted vector contains backbone normalisation affine parameters for TENT and CoTTA*, and flattened LoRA factors for the LoRA configuration (Table 19; Hu et al., 2022). The memory equations are unchanged. The extended experiments assess prediction performance; they do not by themselves isolate retrieval from the remaining host updates. The component and address interventions in Appendices D and G provide that complementary evidence.

Transfer to unseen corruptions. Following Zhang et al. (2026b), we sequentially adapt an ImageNet-pretrained ViT-base (`vit_base_patch16_224`) on Gaussian noise, shot noise, impulse noise, defocus blur, and glass blur, then evaluate on the remaining ten corruptions. Here, ENGRAM wraps EATA, which updates the LayerNorm affine parameters using SGD with learning rate 2.5×10^{-4} and batch size 64. Adaptation makes one online pass over each of the five corruptions; results are averaged over five seeds.

During held-out evaluation, each batch queries the learned memory without host updates, memory writes, or basis growth. The working residual is reset to zero before evaluation and kept zero throughout, so predictions use the source parameters plus the retrieved adjustment. ENGRAM achieves 38.5% mean error, compared with 39.8% for MoASE++, and obtains the lowest error on all ten unseen corruptions among the compared methods (Table 25).

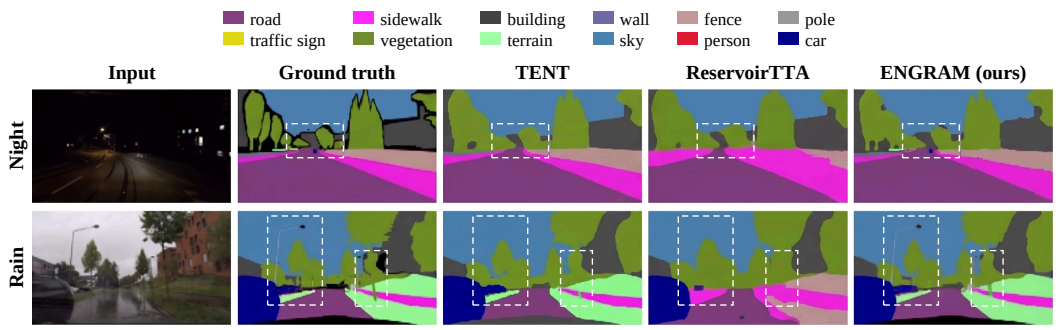


Figure 15: **Additional qualitative Cityscapes→ACDC results.** Night and rain examples complement the fog and snow scenes in Figure 5. Dashed boxes highlight boundaries and small structures. Colours denote semantic classes.

Qualitative segmentation. In addition to qualitative results in Figure 5, Figure 15 adds night and rain domains, together covering all four ACDC conditions. The highlighted regions compare road–sidewalk boundaries and small structures across methods. These examples complement the per-condition mIoU results in Table 19.

F Related Work

Continual and persistent TTA. TENT (Wang et al., 2021) adapts normalisation parameters by minimising prediction entropy on unlabelled test batches. Subsequent methods stabilise adaptation through teacher predictions and parameter restoration (Wang et al., 2022), sample memories (Yuan et al., 2023), reliable-sample filtering and Fisher regularisation (Niu et al., 2022), sharpness-aware optimisation (Niu et al., 2023), and weight ensembling (Marsden et al., 2024). For prolonged deployment, RDumb (Press et al., 2023) periodically resets the model, while PeTTA (Hoang et al., 2024) detects signs of collapse and adjusts adaptation. LCoTTA (Duan et al., 2025) tracks the principal subspace of adaptation gradients and projects updates onto it to suppress directions associated with unreliable samples. Its subspace constrains the current update; our basis represents completed parameter adjustments that can subsequently be retrieved. These approaches address the stability of ongoing adaptation, but do not explicitly associate stored parameter states with returning conditions. Stabilising the current model therefore does not by itself provide a mechanism for recalling a previous adaptation. ENGRAM adds this mechanism while retaining the host's update rule; it does not remove the host's dependence on a useful adaptation objective.

Experts and reusable adaptation states. ReservoirTTA (Vray et al., 2025) discovers domains by clustering style descriptors and stores specialised trainable-parameter states. It selects one state for adaptation but predicts using a similarity-weighted parameter ensemble of the pool. Its discrete storage and adaptation routing should therefore not be confused with hard expert selection for prediction. DPCore (Zhang et al., 2025) retains visual prompts in a dynamic coreset, updating existing prompts or adding new ones according to the incoming domain. BECoTTA (Lee et al., 2024) uses input-dependent routing among low-rank experts. MoASE++ (Zhang et al., 2026b) combines activation-sparsity experts and domain-aware routing with on-policy distillation guided by an exponential-moving-average teacher. IMSE (Baek et al., 2026) adapts singular values in fixed pretrained spectral bases; its retrieval variant stores these spectral codes with domain descriptors and retrieves a matching code after detecting a domain shift. These methods show how specialisation and reuse can support adaptation, using either routed expert modules or explicit collections of reusable states. The corresponding design choices include expert routing, domain detection, and the allocation and selection of stored entries. ENGRAM instead queries a learned map from each batch descriptor to a parameter adjustment, with directions learned from the host's adaptation trajectory. This avoids discrete entry selection, but shared storage introduces interference when keys overlap and cannot represent arbitrary adjustments under a fixed rank budget. Thus, the distinction concerns how adaptations are represented and reused, rather than an unconditional advantage of shared memory over separate states.

Memory mechanisms for adaptation. MINT (Yi et al., 2025) retrieves learned prompt values using hierarchical visual features, while ComMem (Sun et al., 2026) couples a visual cache with progressively refined textual prototypes. These methods accumulate information for vision–language adaptation through their prompt or feature representations. More broadly, delta-rule learning (Widrow & Hoff, 1960) provides an error-correcting update for associative memories and underlies fast-weight formulations (Schmidhuber, 1992; Ba et al., 2016; Schlag et al., 2021). Test-time-training sequence models also update internal memory states online (Sun et al., 2025; Wang et al., 2025), and Miras (Behrouz et al., 2025) organises such designs through their objectives, architectures, retention mechanisms, and update rules. Our write rule uses this established machinery; the contribution is its coupling to an independently specified TTA update through a compensating residual and a growing shared basis. The residual makes consolidation preserve the completed model parameters, while basis expansion preserves all existing reads. These properties specify how memory interacts with active adaptation, a requirement that a generic associative update alone does not resolve. They also leave explicit limits: conservation does not guarantee accurate predictions, and avoiding global decay does not eliminate interference between correlated keys (Section 3).

G Additional Benchmark Controls

The following paired CIFAR-100-C experiments examine address geometry and residual carry under the specified host configurations.

G.1 Address geometry on CIFAR-100-C

We compare three key designs using the same CIFAR-100-C stream, ResNeXt-29 backbone, and EATA host. Each run repeats the 15 corruptions 20 times, with five paired seeds and $r_{\max} = 64$. **Orthogonal keys** encode each corruption as a one-hot vector and require privileged domain labels. **VGG keys** use the standard visual descriptors. **Correlated keys** mix these descriptors so that different corruptions produce more similar keys. Domain labels are used only to construct diagnostic keys; the adaptation objective remains unchanged and uses no class labels.

Table 26 reports error on the first batch of each returning corruption at recurrence 20. These results reflect the full adaptation run: changing keys affects retrieval and can also change later host updates and the directions added to the basis.

Table 26: **Recall under different address geometries.** CIFAR-100-C, ResNeXt-29 and EATA; first-batch error (%) at recurrence 20, mean ± sample standard deviation across five paired seeds. All ENGRAM variants use full carry-over and $r_{\max} = 64$. The pool reference receives oracle domain identities.

Method	Key Geometry	Re-entry Error (%)
ReservoirTTA (Oracle Routing)	N/A	$\mathbf{26.54} \pm 0.41$
ENGRAM	Oracle (Orthogonal)	27.02 ± 0.35
ENGRAM	VGG (Natural Overlap)	27.88 ± 0.38
ENGRAM	Correlated (Blended)	36.81 ± 0.65

Orthogonal keys achieve 27.02% error, close to the oracle-routed pool's 26.54%. Standard VGG keys give 27.88%, while deliberately correlated keys raise error to 36.81%. These results show that recall is sensitive to key similarity, consistent with the interference analysis in Proposition 3. Key overlap does not determine recall alone: memory consolidation, residual carry, and the learned directions also influence performance.

G.2 Residual carry under the main evaluation protocol

This ablation uses CIFAR-100-C under recurring CSC with the ResNeXt-29 backbone and EATA host. It follows the main host configuration, including learning rate 10^{-3}, batch size 200, and Fisher regularisation. Five paired seeds give identical batches, keys, and corruption order to each method within a run. The endpoint is visit-average error at recurrence 20. This batch-size-200 experiment is separate from the batch-size-64 diagnostics in Appendix D.

Both policies use $r_{\max} = 64$, $\beta = 0.1$, $\epsilon = 0.25$, $\lambda = 0.9$, and a minimum growth spacing of ten batches. Full carry passes the entire residual to the next batch; similarity gating attenuates it according to consecutive keys (Equation (9)). The write, compensation, and growth rules are shared, but each policy learns its own basis. The comparison therefore measures the effect of carry policy throughout adaptation, including its influence on subsequent updates and basis growth. Table 9 reports the endpoint errors.

Full carry reduces error relative to similarity gating by 0.75 percentage points (95% CI: $[0.44, 1.07]$). The reductions relative to EATA and EATA + ReservoirTTA are 3.31 $[2.94, 3.68]$ and 0.82 $[0.60, 1.04]$ points, respectively. Confidence intervals are computed from the five seed-level paired differences using the Student-t distribution with four degrees of freedom. Figure 16 shows the individual differences and their means.

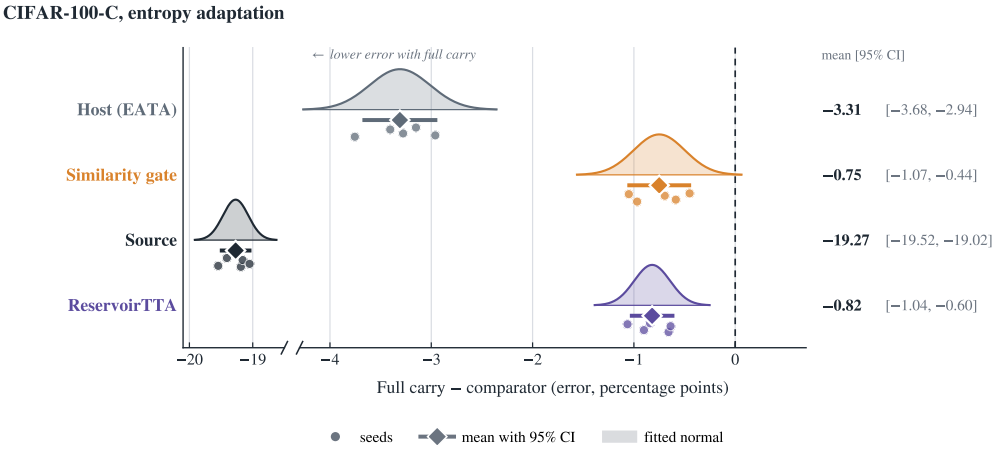


Figure 16: **Paired effects of residual carry.** CIFAR-100-C, five paired stream seeds. Dots are per-seed visit-average error differences at recurrence 20; diamonds and bars show their means with pointwise 95% confidence intervals, and each shaded curve is the normal distribution fitted to that row. Negative values favour full carry.

At the same precision, both carry policies have the same maximum memory allocation (basis, coefficient map, residual, growth accumulator, and previous key); their realised ranks differ because the policies induce different adaptation trajectories. This accounting covers the memory mechanism only and is distinct from peak device memory and from the storage of the pool reference (Appendix C).