CONTINUAL LEARNING FOR VIRTUAL CELL MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent virtual cell or single-cell foundation models are typically pretrained on fixed datasets (Adduri et al., 2026; Cui et al., 2024). However, with the growth of biological data, there is a pressing need to continually train such models on new data distributions without sacrificing performance on previously seen domains. We investigate this continual-learning problem in three existing models, scGPT, scDFM, and X-Cell, using a four-stage Perturb-seq benchmark. Across these backbones, naive fine-tuning loses 69–84% of old-domain performance. Continual training with 5% replay reduces but does not remove this forgetting. On X-Cell, larger replay budgets can reduce forgetting but improve new-domain learning only marginally, so retention trades off against acquisition. Motivated by interference in shared memory, we introduce Profile-Routed Residual Memory (PRM), residual product-key memory layers with a boundary-aware mechanism that separates read composition from write ownership, building on prior work on sparse product-key memory layers. This boundary-aware continual learning method requires observable context and awareness of training boundaries, but not task identifiers, replay of earlier perturbed examples, or distillation. Integrated into X-Cell, a recent single-cell perturbation model, PRM achieves a mean acquisition Pearson-Δ of 0.575 with a final old-domain Pearson-Δ of 0.583, corresponding to 0.6% relative forgetting and both retains and acquires more learning throughout the training campaign compared to other continual learning methods. On other model backbones, PRM shows minimal forgetting. These results provide initial evidence that asymmetric soft reads and private writes are a useful design principle for stable, plastic adaptation.

1 INTRODUCTION

Machine learning models are increasingly expected to adapt after pretraining as new data distributions become available. Continual learning (Parisi et al., 2019) requires balancing two objectives that can be in tension: acquiring useful information from the current distribution while preserving capabilities learned from earlier ones. Naive fine-tuning often causes catastrophic forgetting, where parameter updates that improve the current task degrade performance on previously learned data. Training data replay and distillation can reduce this interference, but introduce their own trade-offs. This problem is particularly relevant for virtual-cell models, which are increasingly pretrained on large fixed collections of single-cell data while new perturbation experiments, cell types, and experimental contexts continue to accumulate.

Large sparse memory layers provide an appealing substrate for continual learning because they offer high parameter capacity while activating and updating only a small subset of values for each input (Berges et al., 2025; Lin et al., 2025). However, sparse access alone does not prevent interference: examples from different data distributions may route to overlapping memory rows, so later updates can overwrite values used by earlier data. As we show in our experiments (Section 5.3; Appendix E), straightforward sparse-memory-based adaptation is not sufficient in sequential single-cell perturbation prediction, where changing cellular contexts induce substantial distribution shifts: unprotected sparse writes forget, whereas protecting earlier writes without adjusting routing leaves new contexts little capacity to learn.

We introduce Profile-Routed Residual Memory (PRM), a residual memory addressed by gene identity and the observable control cell *profile* that separates memory access from write ownership, primarily for continual adaptation of virtual cell models (Figure 1, Figure 2). Product-key addresses

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.

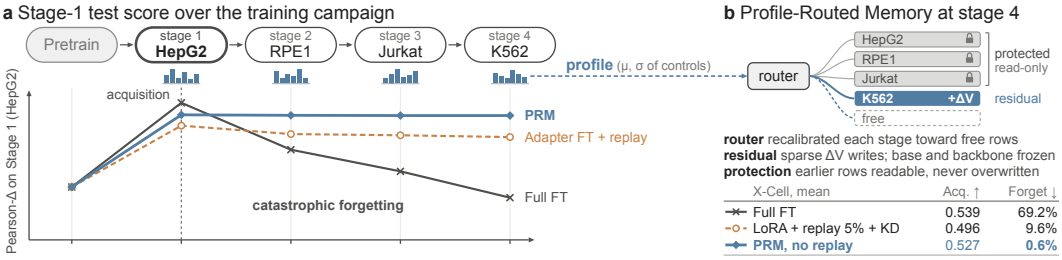

Figure 1: (**a**) Schematic of the stage-1 (HepG2) score during sequential training. (**b**) PRM reads all memory rows but writes only newly allocated residual rows. Inset: artificial example (Table 1).

are factorized into content and observable context components, while continual updates are written to sparse residual values on top of a frozen base memory and backbone. Previously learned residuals remain available for retrieval but are protected from later gradients. When an incoming context routes primarily to protected memory, boundary-time routing calibration and allocation expose writable capacity while anchoring the routing of earlier contexts. The resulting method is boundary-aware during training but requires no task identifier at inference. Importantly, PRM never replays earlier perturbed examples or distills earlier predictions and revisits earlier unperturbed controls only to calibrate routing at boundaries. We evaluate this design on three structurally distinct perturbation backbone models, scGPT (Cui et al., 2024), X-Cell (Wang et al., 2026) and scDFM (Yu et al., 2026), and show that it can substantially reduce forgetting while retaining acquisition of new perturbation domains. Our contributions are:

- We introduce a residual product-key memory framework for continual learning that separates *read access* from *write ownership*: previously learned memory remains readable, while later updates are restricted to writable support and earlier writes can be gradient-protected.
- We formalize continual single-cell perturbation prediction as a sequential learning protocol that evaluates both acquisition of new cellular contexts and retention of previously learned perturbation responses. The protocol specifies the information available during adaptation and inference, and provides common acquisition, retention, and forgetting metrics for comparing continual-learning methods across model backbones.
- We evaluate the proposed memory method on X-Cell, scDFM, and scGPT against full fine-tuning, parameter-efficient adaptation, replay, distillation, regularization, cumulative retraining per stage, and alternative memory-based adaptation strategies. On both memory-augmented backbones, the results characterize the acquisition–retention trade-off and show that the proposed method can achieve strong retention without replaying earlier perturbed examples or distilling earlier predictions as well as avoid catastrophic forgetting through a training campaign.

2 Related Work

Continual Learning. Our setting is boundary-aware during training and domain-incremental in the sense that the output space is fixed while the input context changes (van de Ven et al., 2022). Elastic weight consolidation penalizes movement of parameters important to earlier tasks (Kirkpatrick et al., 2017); Learning without Forgetting preserves earlier outputs through distillation (Li & Hoiem, 2016); and exemplar replay retains a bounded sample of earlier data (Rebuffi et al., 2017). Parameter-isolation methods such as PackNet reserve disjoint parameter subsets for successive tasks (Mallya & Lazebnik, 2018). We evaluate replay with full fine-tuning and LoRA (Hu et al., 2022), and regularization (stagewise L_2, online EWC (Schwarz et al., 2018)) and distillation (latent and response) with LoRA. Cumulative retraining on all data seen so far is a data-retaining reference rather than an online method.

Memory layers. The idea of memory in neural networks was popularized by Memory Networks (Weston et al., 2015) and Neural Turing Machines (Graves et al., 2014), where differentiable mem-

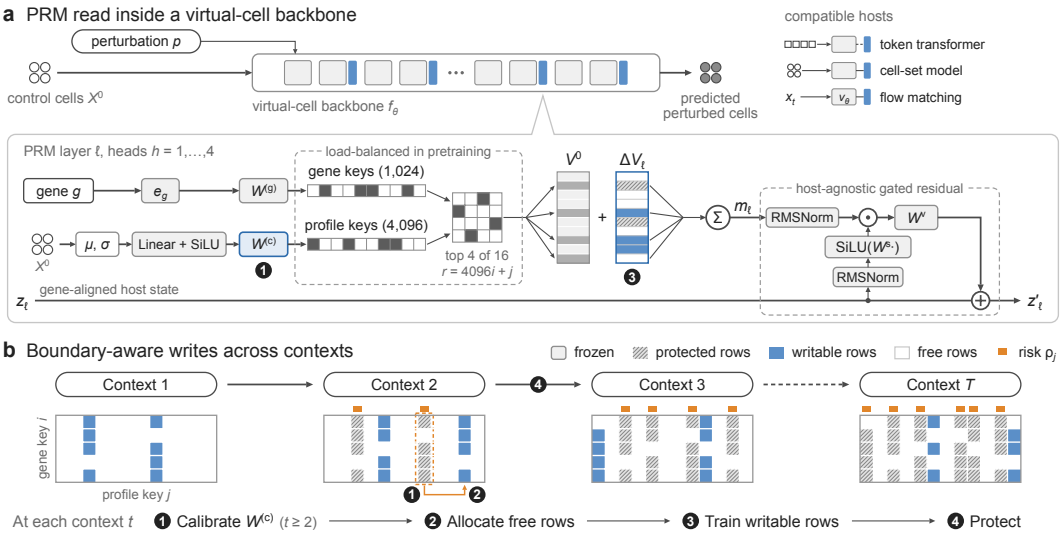

Figure 2: Profile-Routed Residual Memory for continual virtual-cell adaptation. Gene identity and observable control-cell statistics (the profile) address a shared product-key memory. Retrieved values modulate entity-aligned host state. At every training stage boundary, calibration exposes writable capacity while a router claims unprotected residual rows. Write protection of prior rows prevents overwrite while leaving every row readable. Inference uses no task identifier.

ory systems separate computation from addressable state. Product-key memory (PKM) factorizes a large address space into two subkey tables, permitting sparse retrieval from a large value table (Lample et al., 2019). Recent PKM-based layers scale this idea inside transformer language models: memory layers that replace selected feed-forward blocks improve pretraining and factual recall (Berges et al., 2025), product-key retrieval routes among over a million tiny experts (He, 2024), and fine-tuning only the memory slots most specific to new data reduces forgetting when language models acquire new knowledge after pretraining (Lin et al., 2025). We use this factorization to expose product-row routing and to mask residual-value updates across stages.

Single-cell representation and perturbation models. scGen, CPA, and GEARS predict perturbation responses through latent-state arithmetic, compositional covariates, or graph-structured perturbation relations (Lotfollahi et al., 2019; 2023; Roohani et al., 2024). Foundation models such as scGPT and Geneformer instead pretrain gene or cell representations on broad single-cell corpora before downstream adaptation (Cui et al., 2024; Theodoris et al., 2023). Perturbation-specific studies emphasize generalization to held-out interventions or contexts, whereas our protocol fixes a sequence of contexts and measures acquisition and retention after every update (Section 3, Appendix A.

3 Problem Setting: Continual Perturbation Prediction

We consider a sequence of perturbation datasets $\mathcal{D}_1, \ldots, \mathcal{D}_T$, each collected under a distinct biological context, such as a cell line. Each example pairs a perturbation input with its observed response and matched unperturbed control cells. The controls describe the cellular state in which the perturbation is applied; they do not reveal its response. At stage t, the model learns from $\mathcal{D}_t$ and is then evaluated on held-out perturbations from every context in the sequence; the summaries below use only contexts already encountered. The prediction space remains fixed across stages.

We assume known *training boundaries* such that the transition between two successive dataset updates is explicitly identified. This does not imply access to a stage or task identifier at inference. Instead, predictions use the perturbation input and observable context supplied by matched controls. Our method may additionally resample unperturbed controls from earlier contexts for boundary-time calibration, but it never revisits early perturbed examples or targets. Replay baselines retain a

specified subset of earlier perturbed examples, whereas cumulative retraining uses all training data observed so far.

Let $P_{t,s}$ denote a held-out evaluation score on context s immediately after training stage t, with larger scores indicating better predictions. Context s is first introduced at stage s. For a complete sequence with $T \geq 2$, we report mean acquisition,

$$\mathrm{Acq} = \frac{1}{T}\sum_{s=1}^{T} P_{s,s}, \qquad \uparrow \text{ is better} \tag{1}$$

and mean relative forgetting,

$$\mathrm{Forget}_{\mathrm{rel}} = \frac{1}{T-1}\sum_{s=1}^{T-1} \frac{P_{s,s} - P_{T,s}}{P_{s,s}}, \qquad \downarrow \text{ is better.} \tag{2}$$

Acquisition measures performance immediately after learning each context, and retention $\mathrm{Ret} = \frac{1}{T-1}\sum_{s=1}^{T-1} P_{T,s}$ is the final score on earlier contexts. Relative forgetting measures the subsequent score change relative to that context's own acquisition score. Positive values indicate forgetting and negative values indicate backward transfer. The final context is excluded because it undergoes no subsequent update.

Summaries are computed per training sequence and averaged over independent runs. Our primary score is perturbation-delta Pearson correlation (Pearson-Δ).

4 Profile-Routed Residual Memory for Continual Virtual Cells

Profile-Routed Residual Memory (PRM) separates how memory is *read* from where a continual-learning stage is allowed to *write*. All memory remains readable, while each stage updates only newly allocated residual rows that its context can reach.

4.1 Information contract and virtual-cell interface

We write the memory-augmented predictor as $\widehat{y} = f_{\phi,\Theta_t}(u, C)$, where u contains the perturbation and other backbone-specific inputs, C is a matched set of unperturbed control cells, ϕ denotes backbone parameters, and Θ_t denotes the memory state after stage t. Standard PRM keeps ϕ fixed during continual learning. The context router may be adjusted at a training boundary, whereas selected residual memory values are updated during subsequent task training.

At memory insertion layer ℓ, the backbone exposes a hidden state $z_{\ell bng}$ for gene g in cell n of biological set b, together with a fixed gene descriptor e_g. In both implementations, e_g is the backbone's detached gene embedding. The router additionally receives a matched control set $C_b = \{(x_{bn}, m_{bn})\}_{n=1}^{S_b}$, summarized by the masked per-gene mean and standard deviation,

$$c(C_b) = [\mu(C_b); \sigma(C_b)]. \tag{3}$$

Thus e_g specifies *which gene* is queried, while $c(C_b)$ describes the observable *cellular context*. Neither contains a stage identifier. The perturbation-conditioned hidden state modulates the retrieved memory readout, but does not determine its address.

4.2 Context-routed residual memory

Each memory layer uses a product-key router with an entity-key bank $\{K^{\mathrm{ent}}_{\ell hi}\}_{i=1}^{N_1}$ and a context-key bank $\{K^{\mathrm{ctx}}_{\ell hj}\}_{j=1}^{N_2}$ for routing head h. The corresponding queries are

$$\begin{aligned} q^{\mathrm{ent}}_{\ell h}(g) &= W^{\mathrm{ent}}_{\ell h}\,\mathrm{LN}(e_g) + b^{\mathrm{ent}}_{\ell h}, \\ q^{\mathrm{ctx}}_{\ell h}(C) &= W^{\mathrm{ctx}}_{\ell h}\,\mathrm{LN}(\mathrm{SiLU}(W^{s}_{\ell}\, c(C) + b^{s}_{\ell})) + b^{\mathrm{ctx}}_{\ell h}. \end{aligned} \tag{4}$$

Their Cartesian product defines $N_1 N_2$ memory rows. Row $r = (i, j)$ receives score

$$a_{\ell h}(i, j \mid g, C) = \langle q^{\mathrm{ent}}_{\ell h}(g), K^{\mathrm{ent}}_{\ell hi}\rangle + \langle q^{\mathrm{ctx}}_{\ell h}(C), K^{\mathrm{ctx}}_{\ell hj}\rangle. \tag{5}$$

As in product-key memory (Lample et al., 2019), top candidates from the two subkey banks are combined to recover the highest-scoring product rows without searching the full table. Let $\mathcal{R}_{\ell h}(g, C)$ denote the retrieved rows and $\alpha_{\ell h r}(g, C)$ their normalized weights.

The two factors have distinct semantics in PRM. The first depends only on gene identity, while the second depends on the control profile and is the only factor modified during boundary calibration. Context keys are shared learned addresses rather than task-specific buckets.

Each product row stores a pretrained base value and a continual residual. The base table V^0 is shared across insertion layers, while each layer has its own zero-initialized residual table ΔV_ℓ. The memory read is

$$m_{\ell g}(C) = \frac{1}{H} \sum_{h=1}^{H} \sum_{r \in \mathcal{R}_{\ell h}(g,C)} \alpha_{\ell h r}(g, C) \left(V_r^0 + \Delta V_{\ell r} \right). \tag{6}$$

Ownership affects updates but never Equation 6: rows written for earlier contexts remain available to later contexts.

The retrieved memory is injected through a state-dependent residual,

$$z'_{\ell b n g} = z_{\ell b n g} + W_\ell^v [\mathrm{RMSNorm}(m_{\ell g}(C_b)) \odot \mathrm{SiLU}(W_\ell^z \, \mathrm{RMSNorm}(z_{\ell b n g}))]. \tag{7}$$

Hence gene identity and controls determine *where* memory is read, while the perturbation-conditioned backbone state determines how that read affects the prediction. Router preparation and its auxiliary consistency, separation, and load-balancing objectives are described in Appendix B.

4.3 Boundary calibration and constrained adaptation

Let $O_{\ell,<t}$ denote memory rows allocated to earlier continual-learning stages. Standard PRM protects all of them from later updates. Protection prevents direct overwriting, but creates a second problem: an incoming context may route predominantly to protected rows and therefore have little access to writable parameters. PRM addresses this before training each noninitial stage.

Protected-risk calibration. Let $\mathbf{P}_{\ell,<t}(i, j)$ indicate whether product row (i, j) is protected. From the current data, PRM first estimates the normalized traffic $\eta_{\ell h i}^{(t)}$ through each entity key. The protected risk of context key j is

$$\rho_{\ell h j}^{(t)} = \sum_i \eta_{\ell h i}^{(t)} \mathbf{P}_{\ell,<t}(i, j). \tag{8}$$

Intuitively, $\rho_{\ell h j}^{(t)}$ estimates how often choosing context key j would pair current entity traffic with a protected product row. It is a factor-level surrogate; exact product-row routes are measured after calibration.

For current control set C, let $p_{\ell h}^-(j \mid C)$ denote the pre-boundary context-key distribution. PRM constructs a lower-risk target

$$q_{\ell h}^*(j \mid C) \propto p_{\ell h}^-(j \mid C) \left(1 - \widetilde{\rho}_{\ell h j}^{(t)}\right)^\gamma, \tag{9}$$

where $\widetilde{\rho}$ is clipped below one and γ controls the strength of avoidance. This is the closed-form solution to a KL-regularized risk minimization where $\gamma = 0$ preserves the original route, while larger values increasingly suppress high-risk context keys.

Only the context-query projection is optimized to fit these targets. At the same time, routing on earlier controls is anchored to its pre-boundary distribution:

$$\min_{\{\theta_\ell\}} \frac{1}{LH} \sum_{\ell,h} \Big[\mathbb{E}_{C \sim \mathcal{C}_t} \mathrm{KL}(q_{\ell h}^*(\cdot \mid C) \,\|\, p_{\theta_\ell, h}(\cdot \mid C)) + \lambda \, \mathbb{E}_{s<t, C \sim \mathcal{C}_s} \mathrm{KL}\big(p_{\ell h}^-(\cdot \mid C) \,\|\, p_{\theta_\ell, h}(\cdot \mid C)\big) \Big]. \tag{10}$$

The second term preserves memory-routing distributions on earlier unperturbed controls; it does not use earlier perturbation responses. All other memory parameters and the backbone remain fixed. Calibration is skipped for the first stage and applied progressively at every subsequent boundary.

Route-mass allocation and residual updates. After calibration, PRM rescans current controls through the full product router and estimates the normalized route mass $\pi_{\ell,t}(r)$ of every product row. Protected and unobserved rows are excluded. Among the remaining rows, PRM orders candidates by $\pi_{\ell,t}(r)$ and selects the smallest prefix that reaches a target route mass τ, subject to minimum and maximum support sizes $K_{\min}$ and $K_{\max}$. We denote this writable support by $A_{\ell,t}$, with

$$A_{\ell,t} \cap O_{\ell,<t} = \varnothing. \tag{11}$$

If the requested mass or support cannot be reached, allocation is best-effort.

During stage-t task training, the router, backbone, base values, and readout projections are frozen. Only residual rows in $A_{\ell,t}$ may change:

$$\begin{aligned} \min_{U_t} \quad & \mathbb{E}_{(u,y,C)\sim\mathcal{D}_t} \mathcal{L}_{\text{task}} \left(f_{\phi,\Theta_t(U_t)}(u, C), y \right) \\ \text{s.t.} \quad & \text{supp}_{\text{row}}(U_{\ell,t}) \subseteq A_{\ell,t}, \qquad \|\Delta V_{\ell r}^{<t} + U_{\ell,t,r}\|_2 \leq \beta \|V_r^0\|_2. \end{aligned} \tag{12}$$

The support constraint is implemented by a row-exact gradient mask, and only allocated rows actually retrieved by a minibatch receive gradients. At the end of the stage, $A_{\ell,t}$ is added to the ownership set for future boundaries.

This gives PRM its read–write asymmetry: earlier memory remains globally readable, but its owned rows are protected from later writes. Write protection removes direct overwriting but does not eliminate all interference, because calibration can alter earlier routes and earlier contexts may read newly updated rows. We analyze these remaining channels and corresponding new→old, new→new, and old→new route-mass diagnostics in Appendix C.

4.4 Virtual-cell instantiations

The same PRM procedure is inserted after cross-attention blocks of X-Cell (Wang et al., 2026), after differential-perceiver blocks of scDFM (Yu et al., 2026), and after Transformer Encoder blocks in scGPT (Cui et al., 2024), using each backbone's gene embeddings as e_g. Backbone-specific settings are given in Appendix B.

5 Experiments

5.1 Data and evaluation

We used the processed Replogle–Nadig Perturb-seq collection (Replogle et al., 2022; Nadig et al., 2025), restricted to a fixed 1,898-gene panel. The collection has data from four cell lines: K562 and RPE1 derive from Replogle et al. (2022); HepG2 and Jurkat derive from Nadig et al. (2025). Each stage trains on one cell-line split and evaluates on fixed held-out HepG2, RPE1, Jurkat, and K562 splits (Appendix A, Table 2). For each held-out perturbation identity p, evaluation samples $S = 64$ target cells and 64 matched non-targeting controls from the same cellular context. Let $\widehat{\mu}_{p,g}$, $\mu_{p,g}$, and $\mu_{p,g}^{\text{ctrl}}$ denote the predicted, observed, and control means for gene g across the sampled cells, and define

$$\widehat{\Delta}_{p,g} = \widehat{\mu}_{p,g} - \mu_{p,g}^{\text{ctrl}}, \qquad \Delta_{p,g} = \mu_{p,g} - \mu_{p,g}^{\text{ctrl}}. \tag{13}$$

The primary per-perturbation score is Pearson correlation across the fixed $G = 1{,}898$ genes, $\rho_p = \text{corr}_g(\widehat{\Delta}_{p,g}, \Delta_{p,g})$. A context score is the unweighted mean of ρ_p over its held-out perturbation identities; it is therefore a perturbation-macro, not cell-level, statistic.

Let $P_{t,s}$ be this score on context s after training stage t, for $T = 4$ stages. We report acquisition, retention, and relative forgetting as defined in Section 3. All secondary metrics are defined in Appendix D.

5.2 Comparisons

The cross-backbone comparison assigns up to six adaptation roles to each virtual-cell host: full fine-tuning, full fine-tuning with 5% replay, an architecture-native parameter-efficient update, that update with 5% replay, PRM, and the parameter-efficient update with both 5% replay and PRM.

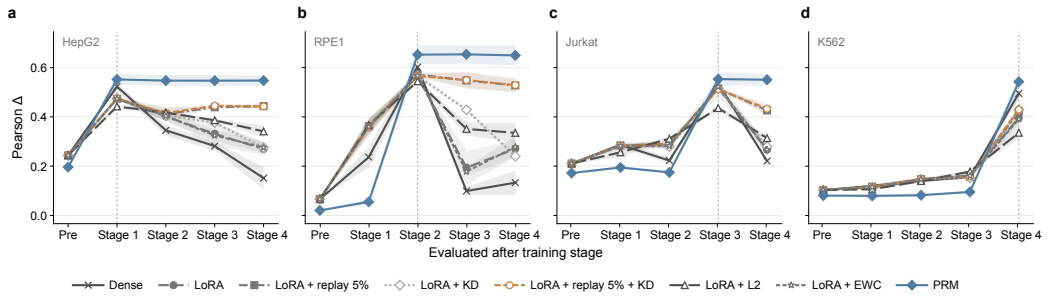

Figure 3: Continual perturbation prediction on X-Cell. PRM learns each new context and keeps it. Perturbation Pearson-Δ on each held-out context (panels) before sequential training (Pre) and after every training stage; the vertical dotted line marks that context's own stage. Dense denotes full fine-tuning and KD response distillation. PRM starts from its own memory-pretrained checkpoint; baseline Pre points were not evaluated on the fixed 1,898-gene panel. Lines and bands, mean and 95% t-interval over three runs.

The parameter-efficient path is rank-64 LoRA in X-Cell, rank-64 residual adapters in scGPT, and the perturbation-response adapters in scDFM. Each replay arm retains 5% of the perturbation groups of each earlier context independently.

The combined arm updates the host-specific adapter parameters together with the selected continual residual values. Boundary calibration and route-mass allocation are unchanged, and the calibrated router is frozen during response optimization. On X-Cell we additionally compare latent LwF, response distillation from a 5% buffer with and without replay, stagewise L_2 regularization, online EWC, and two single-run references: cumulative retraining, which retains all earlier data and is therefore a data-retaining reference, and progressive slots, a stage-private memory with a frozen router and its own pretrained checkpoint (Appendix B). The remaining X-Cell baselines adapt the dense X-Cell checkpoint, whereas PRM adapts a separately pretrained, memory-augmented X-Cell checkpoint.

Table 1 and Figure 3 summarize the comparison. On every backbone, updates without replay lose 48–84% of earlier-context performance, and 5% replay reduces but does not remove this loss. PRM consistently improve the acquisition and retention (less forgetting) on all backbone models tested: particularly on scGPT and X-cell, the PRM variants achieved > 0.57 acquisition with near zero

Table 1: Continual adaptation across virtual-cell backbones under Pearson Δ.

Backbone	Adaptation	Acquisition ↑	Rel. forgetting ↓	Final average ↑
scGPT	Full FT	0.5241 ± 0.0163	$(83.73 \pm 0.88)\%$	0.1857 ± 0.0040
	Full FT + replay 5%	0.5264 ± 0.0112	$(49.97 \pm 6.72)\%$	0.3244 ± 0.0239
	Rank-64 adapters	0.3119 ± 0.0035	$(66.58 \pm 0.15)\%$	0.1223 ± 0.0004
	Adapters + replay 5%	0.3180 ± 0.0025	$(58.51 \pm 2.41)\%$	0.1512 ± 0.0061
	PRM	$\mathbf{0.5735 \pm 0.0020}$	$\mathbf{(0.01 \pm 0.01)\%}$	$\mathbf{0.5735 \pm 0.0021}$
	Adapters + replay 5% + PRM	0.5705 ± 0.0005	$(0.08 \pm 0.26)\%$	0.5702 ± 0.0014
scDFM	Full FT	0.4033 ± 0.0066	$(76.34 \pm 0.43)\%$	0.1533 ± 0.0020
	Full FT + replay 5%	$\mathbf{0.4057 \pm 0.0037}$	$(12.80 \pm 4.05)\%$	0.3664 ± 0.0097
	Response adapters	0.3860 ± 0.0057	$(74.97 \pm 1.31)\%$	0.1544 ± 0.0017
	Adapters + replay 5%	0.3809 ± 0.0070	$(16.28 \pm 3.30)\%$	0.3339 ± 0.0078
	PRM	0.3739 ± 0.0041	$(-0.03 \pm 0.03)\%$	0.3740 ± 0.0042
	Adapters + replay 5% + PRM	0.3625 ± 0.0100	$\mathbf{(-5.77 \pm 2.41)\%}$	$\mathbf{0.3788 \pm 0.0021}$
X-Cell	Full FT	0.5386 ± 0.0039	$(69.20 \pm 2.51)\%$	0.2501 ± 0.0077
	Full FT + replay 5%	0.5444 ± 0.0032	$(10.84 \pm 1.01)\%$	0.4999 ± 0.0012
	Rank-64 LoRA	0.4883 ± 0.0061	$(47.90 \pm 2.58)\%$	0.3006 ± 0.0057
	LoRA + replay 5%	0.4955 ± 0.0025	$(10.40 \pm 0.62)\%$	0.4549 ± 0.0029
	PRM	0.5752 ± 0.0032	$(0.59 \pm 0.13)\%$	0.5726 ± 0.0038
	LoRA + replay 5% + PRM	$\mathbf{0.5901 \pm 0.0029}$	$\mathbf{(-2.02 \pm 1.20)\%}$	$\mathbf{0.5993 \pm 0.0029}$

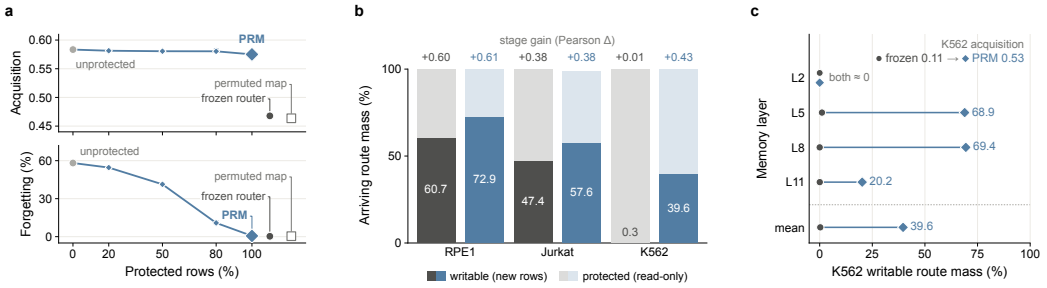

Figure 4: Protection prevents overwriting while calibration restores reachability. (**a**) X-axis sweeps the share of earlier contexts' owned route mass that is write-protected. (**b**) Share of each arriving context's route mass on writable (dark) and protected (light) rows under full protection for a frozen router (gray) and calibrated routing (blue, PRM), with the Pearson-Δ gained at that stage. (**c**) Writable final-stage (K562) route mass per memory layer.

forgetting among stages. X-cell per-context scores stay flat once acquired (Figure 3); the single-run stage-private progressive-slot reference reaches 0.5921. On scDFM, whose memory-pretrained checkpoint starts below the host checkpoint (Appendix B), PRM shows no measurable forgetting (-0.03%) at lower acquisition than full fine-tuning (0.3739 vs. 0.4033); adding adapters and 5% replay yields backward transfer (-5.77%) at lower acquisition (0.3625). X-Cell PRM does, however, have the lowest DE100 recovery at acquisition (0.294 vs. 0.313–0.359) and the highest pseudobulk MAE (Table 5), although both are retained.

5.3 Mechanism and scaling studies

The protection interventions reveal a clear stability–plasticity trade-off (Figure 4). As progressively more earlier-owned memory is write-protected, forgetting falls sharply while acquisition remains nearly unchanged under calibrated routing. Full protection alone, however, is not sufficient. With the router frozen, the final K562 context routes almost entirely to rows that are readable but no longer writable, producing strong retention but little new acquisition. Progressive protected-risk calibration instead redirects a substantial fraction of K562 route mass onto writable support (about 40% on average), and recovers K562 acquisition (Figure 4b–c). Together, these interventions separate two requirements for continual memory: *protection* prevents destructive writes, whereas *calibration* preserves access to trainable capacity.

Replay provides a complementary view of the same stability–plasticity problem (Figure 5). Increasing the replay budget for LoRA with response distillation moves the model primarily toward better retention: forgetting decreases substantially, whereas acquisition improves only modestly and begins to saturate. Replay therefore buys stability mostly by re-exposing the model to earlier perturbation data rather than by improving plasticity on the incoming context. This benefit also carries increasing data retention and training cost (Figure 5a–b). PRM occupies a different region of the acquisition–retention plane, combining strong acquisition with low forgetting without storing earlier perturbed examples. Finally, PRM is insensitive to moderate violations of the clean-boundary assumption in the regime tested here. When 5–20% of each context's perturbations are deferred into the following stage, neither acquisition nor forgetting changes systematically relative to clean-boundary PRM (Figure 5d). Thus the method requires identifiable training boundaries for calibration and allocation, but its performance does not appear to rely on perfectly pure stage composition at these spillover levels.

6 Discussion

PRM enforces three properties by construction. First, the same shared residual tables are readable by every context. Stage identity does not choose a private table at inference. Second, boundary calibration changes only the control-side context projection and anchors its route distributions on

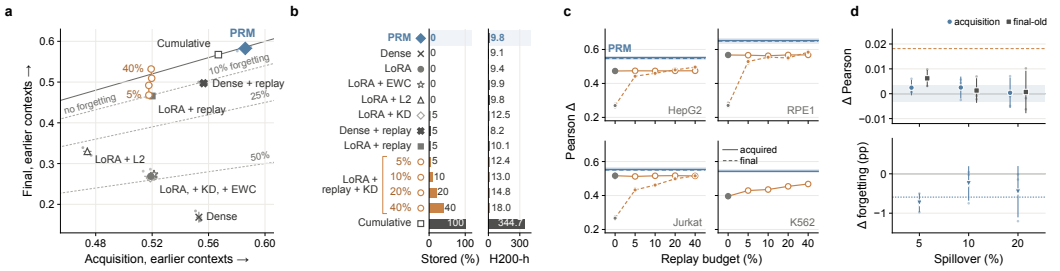

Figure 5: Replay buys retention, not plasticity, while PRM gets both without storing perturbed examples. (**a**) Acquisition vs. final score, both averaged over the three earlier contexts. Solid line marks no forgetting and dashed lines 10, 25, and 50% relative forgetting. (**b**) Stored share of each earlier context's perturbed examples and approximate active-training H200-hours, excluding boundary calibration and evaluation (Appendix B). (**c**) Per-context acquisition (solid) and final score (dashed) vs. replay budget for LoRA with replay and response distillation (orange); blue, PRM. (**d**) Paired change under adjacent-boundary spillover relative to clean PRM; band, ± s.d. of clean PRM acquisition; orange dashed, acquisition gain of 40% replay over LoRA; blue dotted, the change that would remove PRM's forgetting. Mean ± s.d. over at least 3 runs.

earlier controls. Third, write protection is a gradient mask so it does not prevent an old or new context from reading any row.

The current evidence is specific to a boundary-aware four-stage protocol. The protection and routing interventions test write protection and calibrated routing directly while the replay-budget sweep tests whether retaining more data closes the acquisition–retention gap. The spillover study tests a controlled violation of clean stage boundaries Appendix E supporting boundary aware continual learning for modern virtual cell models.

7 Conclusion and Limitations

Profile-Routed Residual Memory combines context-dependent product-key routing, protected residual writes, and boundary-time calibration for continual adaptation of virtual-cell models. The method keeps previously written memory globally readable while restricting later updates to newly allocated support, and uses matched unperturbed controls rather than task labels to determine memory access. Across the virtual-cell backbones studied here, this provides a replay-free mechanism for balancing acquisition of new perturbation contexts with retention of earlier ones.

Several limitations define the scope of the current study. First, training is boundary-aware. Stage transitions must be known, and boundary calibration assumes access to unperturbed controls from both the incoming and earlier contexts, although this can be relaxed. Inference is stage-label-free but likewise requires a control set per context of interest. Second, the experiments cover four cell-line contexts from one processed Perturb-seq collection and a fixed gene panel; broader biological shifts and longer continual sequences remain to be tested. Third, PRM protects knowledge acquired during the continual sequence, not the full pretraining distribution. Pretraining contexts are neither assigned continual ownership nor anchored during boundary calibration, and we therefore observe degradation on the held-out pretraining distribution (Table 4). Finally, the current implementation is sparse in computation but not in storage. The residual tables are densely materialized even though only a small fraction of their rows are eligible for update at each stage. In X-Cell, the complete model contains approximately 10.9B stored values, while at most ∼15M residual values are update-eligible in a stage. In addition, the main X-Cell PRM runs begin from a separately pretrained memory-augmented checkpoint, whereas dense and parameter-efficient baselines begin from the corresponding dense checkpoint. Differences in acquisition therefore reflect both the continual-adaptation rule and, for this comparison, the starting representation. Matching the initialization and storage cost more closely across methods is an important direction for future evaluation.

REPRODUCIBILITY STATEMENT

Appendix A specifies the perturbation-disjoint splits, sampling, normalization, and evaluation protocol. Appendix B gives model and optimization settings. Appendix D defines secondary metrics. All tables and figures are generated by a single script from logged training runs stored in `wandb`. The supplementary material will include code, configuration files, and the source tables for every reported result upon publication.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experiments, implement methods, and interpret results. We have not used generative AI tools to generate synthetic data, develop theoretical models or conceptual frameworks, formulate mathematical claims or assist with proofs, propose or refine hypotheses, clean or reformat datasets, or support qualitative or thematic analysis. Translation is not applicable to this work. Additionally, we have used generative AI tools to assist with experimental-method feedback, software implementation and review, result analysis, figure preparation, bound derivation, and drafting and editing the manuscript. The authors executed the experiments, checked the generated code and analyses against the experiment logs, reviewed all AI-assisted text and claims, and take responsibility for the the complete submission and final content of this work.

ETHICS STATEMENT

This work analyzes previously published single-cell perturbation data. All work and evaluations reported here are retrospective. The original dataset publications provide the provenance of the underlying human-derived data.

REFERENCES

Abhinav K. Adduri, Dhruv Gautam, Beatrice Bevilacqua, Mohsen Naghipourfar, Alishba Imran, Rohan Shah, Noam Teyssier, Rishi Verma, Christopher Carpenter, Basak Eraslan, Francis Chalissery, Rajesh Ilango, Vishvak Subramanyam, Chiara Ricci-Tam, Sanjay Nagaraj, Aidan Winters, Mingze Dong, Stefanie Fellinger, Adam Krejci, Tilmann Burckstummer, Sravya Tirukkovular, Jeremy Sullivan, Brian S. Plosky, Nicholas D. Youngblut, Jure Leskovec, Luke A. Gilbert, Silvana Konermann, Patrick D. Hsu, Alexander Dobin, Dave P. Burke, Hani Goodarzi, and Yusuf H. Roohani. Predicting cellular responses to perturbation across diverse contexts with State. *Cell*, 189(19):5914–5931.e20, 2026. doi: 10.1016/j.cell.2026.07.052. URL `https://pubmed.ncbi.nlm.nih.gov/42673963/`. Published online 31 August 2026.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen-tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3831–3842. PMLR, 2025.

Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scGPT: toward building a foundation model for single-cell multi-omics using generative AI. *Nature Methods*, 21(8):1470–1480, 2024. doi: 10.1038/s41592-024-02201-0.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. *arXiv preprint arXiv:1410.5401*, 2014.

Xu Owen He. Mixture of a million experts. *arXiv preprint arXiv:2407.04153*, 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.

Ann C. Huang, Tsung-Han S. Hsieh, Jiang Zhu, Jackson Michuda, Ashton Teng, Soohong Kim, Elizabeth M. Rumsey, Sharon K. Lam, Ikenna Anigbogu, Philip Wright, Mohamed Ameen,

Kwontae You, Christopher J. Graves, Hyunsung John Kim, Adam J. Litterman, Rene V. Sit, Alex Blocker, and Ci Chu. X-Atlas/Orion: Genome-wide Perturb-seq datasets via a scalable fix-cryopreserve platform for training dose-dependent biological foundation models. *bioRxiv preprint*, 2025. doi: 10.1101/2025.06.11.659105.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, volume 9908 of *Lecture Notes in Computer Science*, pp. 614–629, 2016. doi: 10.1007/978-3-319-46493-0_37.

Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. Continual learning via sparse memory finetuning. *arXiv preprint arXiv:2510.15103*, 2025.

Mohammad Lotfollahi, F. Alexander Wolf, and Fabian J. Theis. scGen predicts single-cell perturbation responses. *Nature Methods*, 16:715–721, 2019. doi: 10.1038/s41592-019-0494-8. URL https://www.nature.com/articles/s41592-019-0494-8.

Mohammad Lotfollahi, Anna Klimovskaia Susmelj, Carlo De Donno, Leon Hetzel, Yuge Ji, Ignacio L. Ibarra, Sanjay R. Srivatsan, Mohsen Naghipourfar, Riza M. Daza, Beth Martin, Jay Shendure, Jose L. McFaline-Figueroa, Pierre Boyeau, F. Alexander Wolf, Nafissa Yakubova, Stephan Günnemann, Cole Trapnell, David Lopez-Paz, and Fabian J. Theis. Predicting cellular responses to complex perturbations in high-throughput screens. *Molecular Systems Biology*, 19(6):e11517, 2023. doi: 10.15252/msb.202211517. URL https://link.springer.com/article/10.15252/msb.202211517.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

Ajay Nadig, Joseph M. Replogle, Angela N. Pogson, Mukundh Murthy, Steven A. McCarroll, Jonathan S. Weissman, Elise B. Robinson, and Luke J. O'Connor. Transcriptome-wide analysis of differential expression in perturbation atlases. *Nature Genetics*, 57(5):1228–1237, 2025. doi: 10.1038/s41588-025-02169-3.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Joseph M. Replogle, Reuben A. Saunders, Angela N. Pogson, Jeffrey A. Hussmann, Alexander Lenail, Alina Guna, Lauren Mascibroda, Eric J. Wagner, Karen Adelman, Gila Lithwick-Yanai, Nika Iremadze, Florian Oberstrass, Doron Lipson, Jessica L. Bonnar, Marco Jost, Thomas M. Norman, and Jonathan S. Weissman. Mapping information-rich genotype–phenotype landscapes with genome-scale Perturb-seq. *Cell*, 185(14):2559–2575.e28, 2022. doi: 10.1016/j.cell.2022.05.013.

Yusuf Roohani, Kexin Huang, and Jure Leskovec. Predicting transcriptional outcomes of novel multigene perturbations with GEARS. *Nature Biotechnology*, 42:927–935, 2024. doi: 10.1038/s41587-023-01905-6.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pp. 4528–4537, 2018.

Christina V. Theodoris, Ling Xiao, Anant Chopra, Mark D. Chaffin, Zeina R. Al Sayed, Matthew C. Hill, Helene Mantineo, Elizabeth M. Brydon, Zexian Zeng, X. Shirley Liu, and Patrick T. Ellinor. Transfer learning enables predictions in network biology. *Nature*, 618:616–624, 2023. doi: 10.1038/s41586-023-06139-9.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1185–1197, 2022. doi: 10.1038/s42256-022-00568-3.

Chloe Wang, Mehran Karimzadeh, Neal G. Ravindra, Lexi R. Bounds, Nader Alerasool, Ann C. Huang, Shihao Ma, Daniel R. Gulbranson, Haotian Cui, Yongju Lee, Anusuya Arjavalingam, Elliot J. MacKrell, Matthew S. Wilken, Jieming Chen, Benjamin W. Herken, Jesse A. Weber, Massimo M. Onesto, Barbara Gonzalez-Teran, Nicole F. Leung, Sally Yu Shi, Byron J. Smith, Sharon K. Lam, Adam Barner, Philip Wright, Elizabeth M. Rumsey, Soohong Kim, Rene V. Sit, Adam J. Litterman, Ci Chu, and Bo Wang. X-Cell: Scaling causal perturbation prediction across diverse cellular contexts via diffusion language models. *bioRxiv preprint*, 2026. doi: 10.64898/2026.03.18.712807.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015.

Chenglei Yu, Chuanrui Wang, Bangyan Liao, and Tailin Wu. scDFM: Distributional flow matching model for robust single-cell perturbation prediction. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=QSGanMEcUV.

A Data and Evaluation Protocol

Sequential splits are constructed by perturbation identity within each cell line. All cells carrying a held-out perturbation form that context's test split and the remaining cells form its fine-tuning split, so train–test perturbation overlap is zero within every context. Upstream test identities are held out; contexts with none upstream (RPE1, Jurkat, and K562) deterministically hold out $\lfloor 0.05N \rfloor$ of their N perturbation identities. Non-targeting cells are pooled by context as controls and are not counted as replayed perturbed examples (Table 2). The X-Cell checkpoints are pretrained on the public genome-wide Perturb-seq atlas (Huang et al., 2025) with 5% of perturbations per screen held out as the pretraining test set (Table 4). No Replogle-Nadig cells enter pretraining. The sequential stages use the Replogle-Nadig collection (Adduri et al., 2026) derived from the K562 and RPE1 screens of Replogle et al. (2022) (`https://doi.org/10.25452/figshare.plus.20029387`) and the HepG2 and Jurkat screens of Nadig et al. (2025).

Table 2: Sequential splits before tokenizer and minimum-cell filtering. Controls are non-targeting cells available for matched control sampling; the last column counts the perturbation groups that pass filtering and enter training.

	Perturbations		Target cells		Available for training	
Context	Train	Test	Train	Test	Controls	Perturbation groups
HepG2	960	380	69,738	26,878	4,976	943
RPE1	1,425	74	166,433	7,304	11,485	1,397
Jurkat	1,461	76	175,618	8,852	12,013	1,424
K562	1,314	69	178,388	10,202	10,691	1,285

Only tokenizer-resolvable perturbations whose groups meet the minimum-cell requirements enter a loader; every evaluation group has at least 32 target cells. Evaluation uses one fixed set per held-out perturbation: 64 target cells sampled with a fixed seed (with replacement only for groups of 32–63 cells) and the first 64 controls of that context in source order, reused for every perturbation in the context. Raw counts are normalized to 10,000 counts per cell using each cell's full library size and $\log(1 + x)$-transformed; already processed inputs are kept. There is no sequential validation; the four-context test suite is evaluated after every stage.

Prediction protocol. X-Cell predicts in four calls over a fixed gene ordering. After each of the first three calls, predictions for the next 25% of the 1,898 genes are clamped at zero and inserted into the next call's input; the fourth call's complete output is scored. scDFM and scGPT predict all genes in a single call. Prediction and target are centered by the same sampled control pseudobulk.

B Implementation Details

Checkpoints. Full fine-tuning, LoRA, regularization, distillation, and replay baselines on X-Cell start from one dense X-Cell checkpoint (108.5M parameters). PRM starts from a separately pretrained memory-augmented X-Cell checkpoint (2.28B parameters, mostly the base value table) whose host, router, and base values are trained jointly for 10 epochs; progressive slots use their own memory-pretrained checkpoint, and cumulative retraining starts from scratch. Differences between checkpoint families, for example in pseudobulk MAE acquisition, therefore include a pretraining effect. Adaptation seeds vary sequential optimization and sampling only.

Memory, calibration, and allocation. Memory pretraining adds support-consistency, cross-context-separation, and entity-/context-subkey load-balancing losses with weights 0.05, 10^{-4}, 10^{-2}, and 10^{-5}; all are disabled during sequential training. Calibration estimates the entity traffic $\eta^{(t)}$ from the first minibatch of eight current control sets and fits all four context projections jointly. The $4{,}194{,}304$-row address space holds one frozen shared base table (2.15B values) and four densely materialized residual tables (8.59B values), so PRM provides sparse optimization and ownership, not sparse storage. When fewer routed unprotected rows exist than $K_{\min}$, or they cannot reach τ, allocation takes all of them. In the reported X-Cell runs the realized support is 24–8,293 rows per layer, so the $K_{\max}$ cap never binds, and the selected share of the current route, averaged over layers,

Table 3: Canonical PRM settings on X-Cell.

Component	Setting
Backbone	12 layers; width 512; 8 attention heads; frozen during sequential adaptation
Memory placement	After cross-attention blocks 2, 5, 8, and 11 (0-indexed)
Product keys	4 heads per block; $1{,}024 \times 4{,}096$ rows; 128 dimensions per subkey
Retrieval	top 4 entity and top 4 context subkeys; top 4 of the 16 product rows
Values	512-wide base table shared across blocks; block-specific zero-initialized residual tables
Control summary	Masked per-gene mean and standard deviation over 64 control cells
Boundary calibration	Context-query projections only; 200 AdamW steps; learning rate 10^{-4}; batch 64; $\gamma = 4$, $\lambda = 1$, $\epsilon = 10^{-6}$; at most 1,024 current and 256 per-earlier-context control sets
Allocation per layer	$K_{\min} = 8{,}192$ rows; $\tau = 0.5$; $K_{\max} = 32{,}768$ rows; best effort
Residual adaptation	10 epochs per stage; fixed learning rate 10^{-2}; row norm at most $\beta = 2$ times the base row
Optimization	Effective batch 64 sets; gradient-norm clip 1; bfloat16
Evaluation	64 targets and 64 controls per perturbation; 1,898 genes

is 0.97, 0.73, 0.58, and 0.40 at HepG2, RPE1, Jurkat, and K562, below τ at the final boundary. A row-exact gradient hook masks all residual rows outside the allocation, and the allocated rows join the protected set after the stage. Protection changes write eligibility only: base and residual rows remain readable in every forward pass. Calibration resamples earlier unperturbed controls, but neither allocation nor optimization loads earlier perturbed inputs or targets.

Mechanism interventions. The mechanism arms (Table 7, Figure 4) replace adaptive allocation with a fixed nominal support of 8,192 rows per layer, which may include rows that receive no current route mass; all other settings are canonical. Fractional protection protects, for each earlier context, the smallest set of its owned rows that covers 20, 50, or 80% of its route mass on owned rows; 0% and 100% protect no or all owned rows. Frozen arms skip calibration. The permuted-risk arm rolls the protected-risk map by half the context-subkey bank (2,048 indices) during calibration, while the real protection mask stays aligned.

Baselines. X-Cell full fine-tuning uses learning rate 10^{-4}, and rank-64 LoRA ($\alpha = 128$) on the response path uses 2×10^{-4}; both use cosine decay with 10% warmup and 10 epochs per stage. Stagewise L_2 penalizes LoRA parameters' distance from their stage-start values (coefficient 0.01); online EWC uses coefficient 10; latent LwF distills a frozen previous-stage teacher's latents on current inputs. A replay fraction is applied independently to each earlier stage: that fraction of the stage's (context, perturbation) groups is sampled without replacement and combined with all current groups, so 5% means 5% of each earlier stage. Buffer response knowledge distillation (KD) uses the same retained old inputs with a frozen previous-stage teacher but no old targets. Cumulative retraining trains X-Cell from scratch at every stage on the pretraining split plus all sequential data seen so far. Progressive slots is a earlier stage-private memory design: a frozen prototype router maps the control summary to context-specific slots of the context-subkey bank, and each stage writes only rows in its slot that no earlier stage has read. Because its writes are confined to control-selected, stage-private slots, it is reported as a reference rather than as a shared-memory learner.

scDFM and scGPT. The adapted scDFM host (Yu et al., 2026) is a four-block differential-perceiver velocity network (width 512, eight heads); a PRM block with the X-Cell memory dimensions follows every perceiver block, and detached gene-encoder outputs supply entity descriptors. PRM warm-starts the clean scDFM checkpoint and trains only the router and base memory, with the host frozen, for six epochs (at most 3,125 updates). This step lowers held-out pretraining Pearson from 0.510 to 0.458, so scDFM PRM starts from a weaker checkpoint than its baselines. Clean scDFM pretraining and its baselines sample 1,000 genes per training set, whereas PRM uses the fixed 1,898-gene panel; splits and evaluation are shared. scDFM full fine-tuning uses learning rate 5×10^{-5} and response adapters 2×10^{-4}. scGPT baselines start from a pretrained scGPT checkpoint (Cui et al., 2024) and are trained for either five epochs per stage or three epochs per stage (fewer epochs for PRM related variants). Full fine-tuning at 5×10^{-5} and rank-64 adapters in all 12 layers

Table 4: Perturbation-Δ Pearson on the held-out pretraining test set before and after the four sequential stages. The dense X-Cell initial score (*) was measured on the pretraining evaluation panel of up to 4,000 genes rather than the fixed 1,898-gene panel used for every other entry, so no drop is reported for dense-checkpoint methods. Final values are single runs except PRM (mean $\pm$ one sample standard deviation, three runs); cumulative retraining starts from scratch.

Method	Checkpoint	Initial	After K562	Drop $\downarrow$
Naive full fine-tuning	Dense X-Cell	0.6565^*	0.0996	—
Rank-64 LoRA	Dense X-Cell	0.6565^*	0.0493	—
Rank-64 LoRA + 5% replay	Dense X-Cell	0.6565^*	0.0972	—
PRM	PRM	0.7335	0.1358 ± 0.0042	0.5978 ± 0.0042
Cumulative retraining	From scratch	—	0.6979	—

at 2×10^{-4}. Hybrid arms (adapters + 5% replay + PRM) train the host's parameter-efficient path at 2×10^{-4} and the selected residual values at 10^{-2}; calibration, allocation, and write protection are unchanged, and all other host, key, base-value, and router parameters are frozen during response optimization.

Compute. Adaptation compute (Figure 5) is an active-training estimate: the median sampled microbatch time multiplied by completed optimizer updates, gradient-accumulation steps, and the 64 allocated H200 GPUs. It excludes data loading, evaluation, boundary calibration, queueing, and initialization, so it compares training work rather than wall-clock or billed cost.

Retention of the pretraining function. Retention in the main protocol refers to earlier sequential contexts, not to the pretraining distribution (Table 4). On the held-out pretraining test set, the PRM checkpoint scores 0.7335 before sequential adaptation and 0.1358 ± 0.0042 after K562. PRM protects rows owned by earlier sequential stages, but pretraining contexts receive neither row ownership nor route anchoring, so their routes can reach newly written, globally readable residuals or move under calibration. Separating these two channels requires a targeted pretraining-route intervention and remains a limitation.

C Interference Decomposition

For values-only PRM, the host and router remain fixed during response optimization. For an earlier context $s < t$, average over the same references, entities, and heads used by the routing scan:

$$\bar{\pi}_{\ell,s}(r) = \mathbb{E}_{C \sim \mathcal{C}_s, g, h}[\alpha_{\ell h r}(g, C)], \qquad \bar{m}_{\ell,s} = \sum_r \bar{\pi}_{\ell,s}(r) v_{\ell r}, \tag{14}$$

where $\alpha_{\ell h r} = 0$ outside the retrieved support. Let $\bar{\pi}^-_{\ell,s}$ and $\bar{\pi}^+_{\ell,s}$ be these distributions immediately before and after boundary calibration, let $v^{<t}_{\ell r} = V^0_{\ell r} + \Delta V^{<t}_{\ell r}$, and let $u^{(t)}_{\ell r} = U_{\ell,t,r}$ be the stage increment. The read-level change decomposes exactly as

$$\begin{aligned} \bar{m}^t_{\ell,s} - \bar{m}^{t-1}_{\ell,s} = & \underbrace{\sum_r \big(\bar{\pi}^+_{\ell,s}(r) - \bar{\pi}^-_{\ell,s}(r)\big) v^{<t}_{\ell r}}_{\text{boundary route drift}} \\ & + \underbrace{\sum_{r \in A_{\ell,t} \cap O_{\ell,<t}} \bar{\pi}^+_{\ell,s}(r) u^{(t)}_{\ell r}}_{\text{reuse of earlier-owned rows}} + \underbrace{\sum_{r \in A_{\ell,t} \setminus O_{\ell,<t}} \bar{\pi}^+_{\ell,s}(r) u^{(t)}_{\ell r}}_{\text{prior exposure to newly claimed rows}} . \end{aligned} \tag{15}$$

Under standard full protection, $\mathcal{P}_{\ell,<t} = O_{\ell,<t}$ and $A_{\ell,t} \cap O_{\ell,<t} = \varnothing$, so the middle overwrite term vanishes. If $B_\ell = \max_r \|v^{<t}_{\ell r}\|$ and $R_{\ell,t} = \max_{r \in A_{\ell,t}} \|u^{(t)}_{\ell r}\|$, the remaining two channels satisfy

$$\left\| \bar{m}^t_{\ell,s} - \bar{m}^{t-1}_{\ell,s} \right\| \le B_\ell \|\bar{\pi}^+_{\ell,s} - \bar{\pi}^-_{\ell,s}\|_1 + R_{\ell,t}\, \bar{\pi}^+_{\ell,s}(A_{\ell,t}). \tag{16}$$

The prior-router KL in Equation 10 is a factor-level surrogate intended to reduce route drift; full product routes directly measure the route-mass factor $\bar{\pi}^+_{\ell,s}(A_{\ell,t})$ in the second term. Equations 15–16 describe the memory read before residual modulation and downstream nonlinearities; they are not

a bound on prediction drift or end-to-end forgetting. Any update that also modifies host parameters introduces additional interference channels outside this values-only decomposition.

This decomposition motivates three boundary diagnostics, all measured before response optimization. With "new" denoting the current context, "old" the set of earlier contexts, and π^+ the post-calibration routes ($\pi^+_{\ell,t} = \pi_{\ell,t}$, $\pi^+_{\ell,s} = \bar{\pi}^+_{\ell,s}$), define

$$\begin{aligned} M^{\text{new}\to\text{old}}_{\ell,t} &= \pi^+_{\ell,t}(O_{\ell,<t}), \\ M^{\text{new}\to\text{new}}_{\ell,t} &= \pi^+_{\ell,t}(A_{\ell,t} \setminus O_{\ell,<t}), \\ M^{\text{old}\to\text{new}}_{\ell,t} &= \frac{1}{t-1}\sum_{s<t} \pi^+_{\ell,s}(A_{\ell,t} \setminus O_{\ell,<t}). \end{aligned} \tag{17}$$

The first quantity is current mass on earlier-owned rows; under full protection it is readable but unavailable for new writes. The second is current mass on newly claimed writable rows. The third is the mean prior-context exposure to those changes and appears directly in Equation 16. The current route partitions as

$$1 = M^{\text{new}\to\text{old}}_{\ell,t} + M^{\text{new}\to\text{new}}_{\ell,t} + \pi^+_{\ell,t}\big([N_1]\times[N_2] \setminus \big(O_{\ell,<t} \cup (A_{\ell,t} \setminus O_{\ell,<t})\big)\big). \tag{18}$$

Thus full write protection may yield perfect non-overwrite while still destroying plasticity: if new→old mass approaches one, almost none of the current route can reach newly writable values. Boundary calibration attacks precisely this failure mode; route-mass allocation converts exposed support into trainable capacity; and prior-route anchoring plus bounded residuals target the two read-level channels that remain under full protection.

D Secondary Metrics

All scores are macro-averaged over held-out perturbations. Pearson correlation centers each vector across genes and clamps the product of norms at 10^{-8}; relative forgetting requires $P_{s,s} > 10^{-8}$, which holds for every reported run. Let $M_{t,s}$ be the pseudobulk MAE on context s after stage t. Because lower is better, its forgetting analogue is the relative increase

$$\text{Increase}^{\text{MAE}}_{\text{rel}} = \frac{1}{T-1}\sum_{s=1}^{T-1} \frac{M_{T,s} - M_{s,s}}{M_{s,s}}, \tag{19}$$

normalized per context before averaging; positive values indicate degradation. The DE100 metrics use, for each perturbation, the 100 genes with the largest observed $|\Delta_{p,g}|$. DE100 Pearson correlates $\widehat{\Delta}_{p,g}$ with $\Delta_{p,g}$ on those genes, and DE100 recovery is the fraction of them that are also among the 100 largest predicted $|\widehat{\Delta}_{p,g}|$. Both use the acquisition and relative-forgetting definitions of Section 3.

E Additional Results

Host drift despite protected memory. Jointly updating the host and PRM (Table 6) raises X-Cell acquisition from 0.5752 to 0.6025 but also raises relative forgetting from 0.59% to 24.38%; on scDFM it yields 31.19% forgetting over two runs. Row ownership prevents direct value overwrite but cannot constrain drift in the host features and queries that address those values, which motivates the frozen host.

Adapters, replay, and memory. On scDFM, response adapters with 5% replay and PRM reach $(-5.77 \pm 2.41)\%$ relative forgetting, but acquisition falls to 0.3625 ± 0.0100, below replay-free PRM (0.3739 ± 0.0041). Write isolation and replay can therefore produce backward transfer without improving plasticity.

Protocol robustness and earlier designs. Figure 6a repeats the adjacent-boundary spillover study for PRM. Stage-order robustness (Figure 6b) was measured only for progressive slots. Across 22 of 24 stage orders, acquisition ranges from 0.5920 to 0.5927 and relative forgetting stays at or below 0.1%. Earlier memory designs, run under development protocols and not directly comparable, failed

Table 5: Additional continual-learning controls on X-Cell. Values are mean ± one sample standard deviation across adaptation seeds from a fixed method-specific pretrained checkpoint; single-seed references are reported without uncertainty. All scores use Pearson Δ. Relative forgetting is computed per context and chain before aggregation.

Method	n	Acquisition ↑	Relative forgetting ↓	Final average ↑
Naive full fine-tuning (10 epochs)	3	0.5386 ± 0.0039	$(69.20 \pm 2.51)\%$	0.2501 ± 0.0077
Rank-64 LoRA (10 epochs)	3	0.4883 ± 0.0061	$(47.90 \pm 2.58)\%$	0.3006 ± 0.0057
Rank-64 LoRA + 5% replay (10 epochs)	3	0.4955 ± 0.0025	$(10.40 \pm 0.62)\%$	0.4549 ± 0.0029
Rank-64 LoRA + 5% buffer response KD	3	0.4963 ± 0.0016	$(48.47 \pm 0.40)\%$	0.3059 ± 0.0014
Rank-64 LoRA + latent LwF	3	0.4863 ± 0.0043	$(46.10 \pm 0.51)\%$	0.3066 ± 0.0023
Rank-64 LoRA + 5% replay + response KD	3	0.4956 ± 0.0019	$(9.62 \pm 0.74)\%$	0.4583 ± 0.0047
Rank-64 LoRA + stagewise L2	3	0.4395 ± 0.0047	$(29.75 \pm 2.40)\%$	0.3314 ± 0.0047
Rank-64 LoRA + online EWC	3	0.4887 ± 0.0040	$(47.18 \pm 2.28)\%$	0.3032 ± 0.0058
Progressive slots (10 epochs)	1	0.5921	0.01%	0.5921
Profile-Routed Residual Memory	3	0.5752 ± 0.0032	$(0.59 \pm 0.13)\%$	0.5726 ± 0.0038
Cumulative retraining	1	0.5678	0.21%	0.5681

Table 6: Full-backbone adaptation ablation. Mean ± one sample standard deviation across completed adaptation seeds; replay retains 5% per earlier context. PRM rows start from the PRM checkpoint and jointly update the host and continual values; n is the number of completed runs.

Backbone	Adaptation	n	Acquisition ↑	Relative forgetting ↓
scDFM	Full fine-tuning	3	0.4033 ± 0.0066	$(76.34 \pm 0.43)\%$
	Full fine-tuning + 5% replay	3	$\mathbf{0.4045 \pm 0.0038}$	$(\mathbf{12.06 \pm 0.73})\%$
	Full fine-tuning + PRM	2	0.3746 ± 0.0044	$(31.19 \pm 2.36)\%$
X-Cell	Full fine-tuning	3	0.5386 ± 0.0039	$(69.20 \pm 2.51)\%$
	Full fine-tuning + 5% replay	3	0.5444 ± 0.0032	$(\mathbf{10.84 \pm 1.01})\%$
	Full fine-tuning + PRM	3	$\mathbf{0.6025 \pm 0.0020}$	$(24.38 \pm 1.90)\%$

in one of two ways. Updating the router, selected feed-forward layers, or the full host acquired more (0.39–0.55) but forgot 23–73%, whereas values-only writes, including TF-IDF- and access-frequency-ranked row selection as in sparse memory finetuning (Lin et al., 2025), forgot at most 11.7% but acquired only 0.15–0.36. Changing the row selector or capping route collisions changed acquisition by at most 0.001.

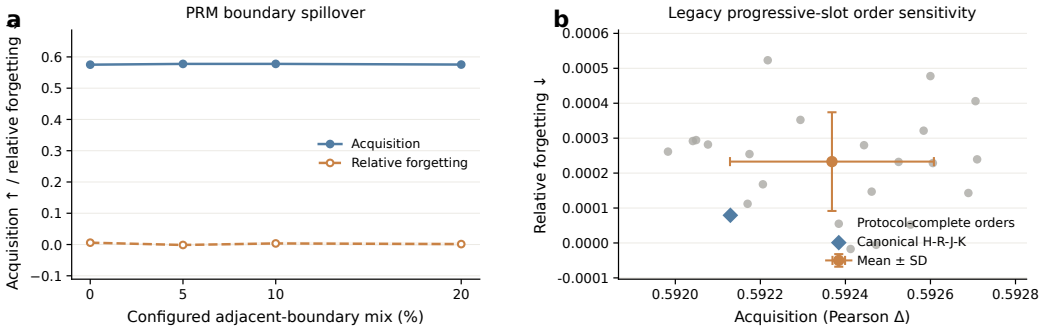

Figure 6: Protocol robustness. (**a**) PRM acquisition and relative forgetting under adjacent-boundary spillover. (**b**) The progressive-slot reference across 22 protocol-complete stage orders; the canonical HepG2→RPE1→Jurkat→K562 order is highlighted.

Table 7: Profile-Routed Memory mechanism ablations. Target and achieved protection are prior-context route-mass fractions measured before adaptation. Acquisition averages all four contexts; K562 isolates the final-boundary plasticity failure; relative forgetting normalizes each earlier context's signed acquisition-to-final drop by its acquisition reference. Every row uses the same pretrained checkpoint, fixed 8,192-row support per layer, canonical order, and 10 epochs per stage. Values are mean ± one sample standard deviation across the completed adaptation seeds shown by n.

Router	Target protection	Achieved protection	n	Acquisition ↑	K562 acquisition ↑	Relative forgetting ↓
Frozen	0.0%	0.0%	2	0.5834 ± 0.0001	0.5711 ± 0.0130	$(58.16 \pm 0.39)\%$
Frozen	100.0%	100.0%	1	0.4678	0.1143	0.30%
Progressive	20.0%	21.1%	1	0.5814	0.5583	54.56%
Progressive	50.0%	51.5%	1	0.5805	0.5601	41.33%
Progressive	80.0%	80.6%	3	0.5804 ± 0.0036	0.5605 ± 0.0102	$(10.85 \pm 1.42)\%$
Progressive	100.0%	100.0%	1	0.5750	0.5310	0.59%
Permuted risk map	100.0%	100.0%	3	0.4636 ± 0.0046	0.1138 ± 0.0003	$(0.38 \pm 0.06)\%$

Table 8: Matched hyperparameter studies. Replay fractions retain that percentage independently from each prior stage; positive replay fractions also use response distillation, whereas the 0% row is the LoRA-only baseline. Spillover delays the indicated fraction of each context into the next stage without materially changing the example budget. Values are mean ± one sample standard deviation over adaptation seeds 42–44 from a fixed method-specific pretrained checkpoint.

Study	Fraction	n	Acquisition ↑	Relative forgetting ↓
LoRA-only baseline	0%	3	0.4883 ± 0.0061	$(47.90 \pm 2.58)\%$
Replay + response KD	5%	3	0.4956 ± 0.0019	$(9.62 \pm 0.74)\%$
Replay + response KD	10%	3	0.4970 ± 0.0015	$(5.01 \pm 0.19)\%$
Replay + response KD	20%	3	0.5032 ± 0.0015	$(1.85 \pm 0.91)\%$
Replay + response KD	40%	3	0.5064 ± 0.0019	$(-2.49 \pm 0.09)\%$
Adjacent-boundary spillover	0%	3	0.5752 ± 0.0032	$(0.59 \pm 0.13)\%$
Adjacent-boundary spillover	5%	3	0.5776 ± 0.0017	$(-0.14 \pm 0.32)\%$
Adjacent-boundary spillover	10%	3	0.5777 ± 0.0028	$(0.35 \pm 0.56)\%$
Adjacent-boundary spillover	20%	3	0.5756 ± 0.0030	$(0.14 \pm 0.68)\%$