LEARNING (AND FORGETTING) AS STORAGE AND COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks exhibit strong generalization, yet the mechanisms underlying feature learning and its failure to learn under distributional shift (catastrophic forgetting) remain poorly understood. We introduce the Storage, Compression, and Targets (SCT) framework, which unifies neural learning as the interplay between storing associations, compressing representations, and aligning neural activity with local targets. We show that classical credit assignment algorithms—including backpropagation, predictive coding, and target propagation—can be formulated as specific instances of SCT rules. Through this lens, we identify compression as a primary driver of feature learning: once associations are stored, compression alone can reorganize their retrieval geometry, producing feature learning and generalization without further storage However, the same mechanism can also drive forgetting. By studying the representation drift due to storage and compression rules, we prove that fully preventing task degradation requires corrections coupled to the current sample and, once representations reorganize, access to past inputs through replay or a generative proxy. Finally, we empirically validate our findings by deriving a replay-based correction mechanism that improves retention and sample-efficiency of replay-based methods for class-incremental learning.

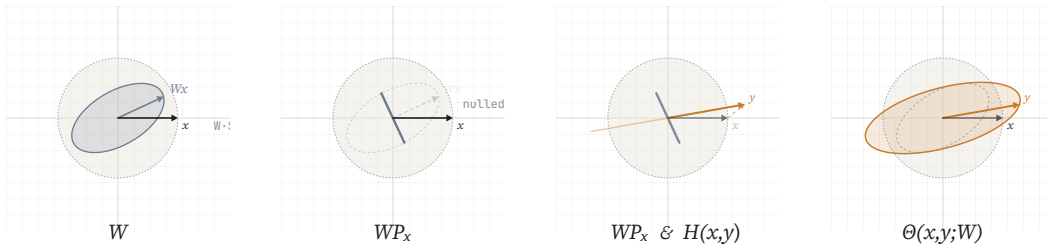
1 INTRODUCTION

Deep neural networks exhibit remarkable generalization, yet the exact mechanistic processes driving feature learning remain poorly understood. While the Neural Tangent Kernel (NTK) (Jacot et al., 2018; Arora et al., 2019) elegantly characterizes gradient descent in the infinite-width limit, it often obscures the layer-wise representational changes that define the *rich* regime (Dominé et al., 2024). Conversely, representation-focused theories, such as the Information Bottleneck (Tishby & Zaslavsky, 2015; Achille & Soatto, 2018), emphasize global compression but abstract away the mechanistic weight updates. Bridging this gap requires a localized, mathematical understanding of how individual layers store and compress information.

A particularly striking consequence of this theoretical gap is our understanding of *catastrophic forgetting* (McCloskey & Cohen, 1989; Parisi et al., 2019). When trained sequentially on changing distributions, networks rapidly overwrite prior knowledge. Standard mitigation strategies, including weight regularization (Kirkpatrick et al., 2017; Zenke et al., 2017), architectural isolation (Rusu et al., 2016), and experience replay (Rebuffi et al., 2017; Buzzega et al., 2020), largely treat forgetting as a bug to be suppressed. However, early connectionist theories hypothesized that forgetting is intimately tied to the very mechanisms that enable generalization (French, 1999). A key open question remains: how can we mathematically unify generalization and forgetting to navigate the "stability-plasticity dilemma" (Mermillod et al., 2013)?

To address this, we reframe learning as the layer-wise interplay of **Storage, Compression, and Targets (SCT)**. We posit that any learning update decomposes into: **1. (Storage)** a Hebbian-like process (Oja, 1982) encoding desired input-output associations; **2. (Compression)** a mechanism that "nulls" responses to current inputs to make space for new associations; and **3. (Targets)** the local or global objectives guiding the representational change. Crucially, this framework can describe standard backpropagation as well as biologically plausible alternative learning rules, such as target propagation (Bengio, 2014; Meulemans et al., 2020) and predictive coding (Rao & Ballard, 1999; Whittington & Bogacz, 2017). We show that these algorithms are specific instantiations of the same

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.



(a) **Original** Image of the unit circle under W (ellipse)

(b) **Compression** Rank drops by 1: the ellipse collapses onto the line $(Wx)^{\perp}$.

(c) **Storage insertion** Overlay WP_x with a rank-1 segment from 0 to y along the x-direction.

(d) **Updated** New ellipse (warm) with $\Theta x = y$. vs. original W (cool dashed).

Figure 1: Visualization of storage and compression for a matrix W.

underlying SCT operators, differing primarily in their cross-layer target assignment. Previous works connected compression and the learning capabilities of statistical models (Arora et al., 2018; Campi & Garatti, 2023; Pandey, 2024), but they do not address how compression is enacted mechanistically.

By isolating the compressive operator common to these rules, we leverage Random Matrix Theory to demonstrate that compression is the primary driving force behind feature learning, generalization, and overfitting. In particular, we show how compression in multilayer networks can act as a spectral sieve, where feature learning happens as a compression driven Baik-Ben Arous-Péché (BBP) transition (Baik et al., 2005) in the eigenvalues of each layer input's covariance. By geometric extension, we prove that it is also the root cause of catastrophic forgetting.

Characterizing this layer-wise representational drift, we prove that, once representations reorganize, preventing task degradation requires access to past data, via replay or generative proxies (Ha & Schmidhuber, 2018), supporting recent theoretical findings (Khademi Nori & Kim, 2024). Guided by these insights, we propose a correction term of the forgetting drift due to backpropagation that shifts how replays are used. We empirically show on a class-incremental benchmark that this geometry-aware correction significantly improves both retention and sample efficiency.

Our contributions are summarized as follows:

- We introduce the SCT framework, providing a layer-wise mechanistic interpretation of neural learning, and prove its equivalence to popular deep learning algorithms.
- Under this lens, we show that the compressive term in these algorithms drives generalization and overfitting by shaping the metric through which associations are retrieved, and that the same compression kernel drives forgetting.
- We characterize the layer-wise distortion that drives forgetting in backpropagation, proving a set of necessary conditions to overcome it. We empirically demonstrate how the correction of this drift improves retention and sample efficiency of replay-based methods for class-incremental learning.

2 SCT Framework: Storage, Compression, and Targets

We now formalize a unifying framework that describes neural network learning through three fundamental principles: **Storage**, **Compression**, and **Targets** (SCT). At its core, the SCT framework posits that any learning algorithm can be decomposed into layer-wise operations that **store** desired input-output relationships, **compress** existing representations to make space for new knowledge and promote generalization, and are guided by **targets** that define the overall learning objective.

2.1 The Building Blocks of Learning

Consider the simplest form of learning: a single linear layer with weight matrix $W \in \mathbb{R}^{m \times n}$ mapping an input vector $x \in \mathcal{X} \subseteq \mathbb{R}^n$ to a desired output $y \in \mathcal{Y} \subseteq \mathbb{R}^m$. We ask: what is the family of update operators Θ such that the new weights satisfy $\Theta x = y$? We define the following matrix

parameterization:

$$\Theta(x, y) = \underbrace{H(x, y)}_{x \mapsto y \text{ map}} + \underbrace{K(x)}_{\text{kernel}}$$

where $H(x, y)$ satisfies $H(x, y)x = y$, and $K(x)$ is a matrix containing x in its kernel ($K(x)x = 0$). More generally, let $\mathcal{M}_x^y$ be the space of $x \mapsto y$ mapping matrices, such that $Mx = y$ if $M \in \mathcal{M}_x^y$, and let the kernel space be $\mathcal{K}_x \simeq \mathcal{M}_x^0$. Thus, Θ is the sum of a *mapping* function $H(x, y) \in \mathcal{M}_x^y$ and an orthogonal *kernel* component $K(x) \in \mathcal{K}_x$.

We add the constraint that, given a prior weight matrix W and a subspace $\mathcal{V} \subset \mathcal{X} \setminus \{x\}$, our kernel function must satisfy:

$$K(x)v \implies \pi(W|x)v = \begin{cases} 0 & \text{if } v = x \\ Wv & \text{if } v \in \mathcal{V} \end{cases} \tag{1}$$

Consequently, we parameterize the update operator Θ as the combination of these two fundamental components:

$$\Theta(x, y; W) = \underbrace{H(x, y)}_{\text{Storage Function}} + \underbrace{\pi(W|x)}_{\text{Compression Function}} \tag{2}$$

1. **Storage Function,** $H(x, y) \in \mathcal{M}_x^y$**:** This component directly encodes the desired (x, y) association. An exemplary realization is the normalized rank-1 *Hebbian term* $\bar{H}(x, y) = \frac{yx^T}{\|x\|^2}$ (Oja, 1982).
2. **Compression Function,** $\pi(W|x) \in \mathcal{K}_x$**:** This component manages existing mappings to enable new learning. It takes the prior weight matrix W (Fig 1a) and modifies it such that its response to the current input x is "nulled" or compressed (Fig 1b). This makes space for the new association to be stored via $H(x, y)$ (Fig 1c-d). Since $\pi(W|x)x = 0$, the output for x is solely determined by the storage term. For $\mathcal{V} = \mathbb{R}^n \setminus \{x\}$, a natural realization is the orthogonal projector:

$$\bar{\pi}(W|x) = W\left(\mathbb{I} - \frac{xx^T}{\|x\|^2}\right) = WP_x$$

where P_x projects onto the subspace perpendicular to x.

Combining these specific realizations yields the exact *Hebbian-Orthogonal Projector (HOP)* operator:

$$\bar{\Theta}(x, y; W) = \frac{yx^T}{\|x\|^2} + W\left(\mathbb{I} - \frac{xx^T}{\|x\|^2}\right)$$

We can modulate this update by introducing a learning rate β^{-1}, leading to a "relaxed" (as opposed to exact) mapping where $\Theta(x, y; W)x = \lambda y + (1 - \lambda)Wx$ for some $\lambda \in \mathbb{R}$. The relaxed HOP operator becomes:

$$\bar{\Theta}(x, y; W) = \frac{yx^T}{\beta} + W\left(\mathbb{I} - \frac{xx^T}{\beta}\right)$$

which is exact when $\beta = ||x||^2$. The following properties immediately follow.

Lemma 2.1 (Equivalence to Gradient Descent)**.** *The HOP operator $\bar{\Theta} : \mathcal{X} \times \mathcal{Y} \times \mathcal{M} \to \mathcal{M}$, with the normalized Hebb rule (Oja's rule) as the storage function and the orthogonal projector WP_x as the compression function, is equivalent to a gradient step in weight space with respect to the MSE loss $\mathcal{L} = \frac{1}{2}||Wx - y||^2$ with learning rate β^{-1}:*

$$\bar{\Theta}(x, y; W) = W - \beta^{-1}\frac{\partial \mathcal{L}}{\partial W}$$

Theorem 2.2 (Contraction Property)**.** *Any operator Θ defined as in Eq. 2, with $H(x, y) \in \mathcal{H}_{x,y}$ and $\pi(W|x) \in \mathcal{K}_x$ satisfying Eq. 1, is a transverse contraction in matrix space toward the solution subspace $\mathcal{M}_x^y \simeq \mathbb{R}^{m \times (n-1)}$. Moreover, any such operator can be rewritten as:*

$$\begin{aligned} \Theta(x, y; W) &= \beta^{-1}yx^T + WP_x + \Delta\pi \\ &= (1 - \lambda)W + \lambda W_\perp^* + \Delta\pi \end{aligned}$$

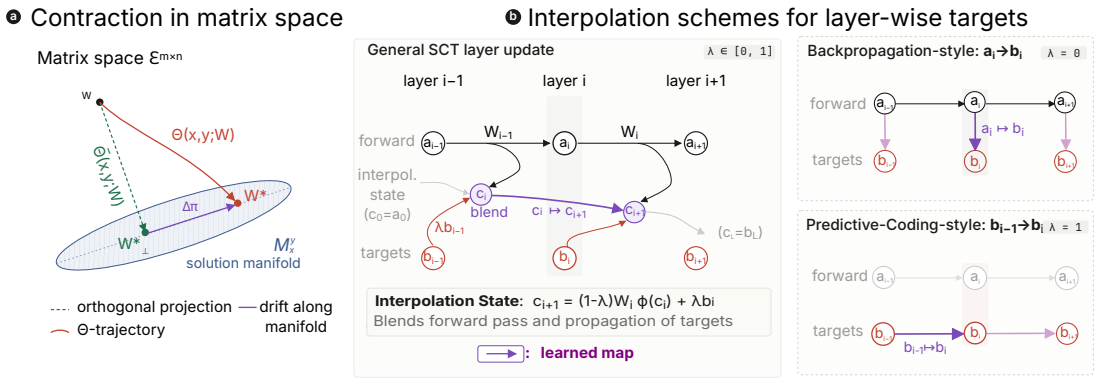


Figure 2: **(a)** Contraction property of Θ in matrix space. $\Theta(x, y; W)$ pushes W toward the solution space $\mathcal{M}_x^y$. **(b)** Interpolation Schemes: the top diagram illustrates the general case where a layer learns to map a blend of its input a_i and the previous layer's target b_{i-1} to its target b_i (see App. C.4). The bottom-left shows $\lambda = 0$ (Backpropagation-style: $a_i \mapsto b_i$) and the bottom-right shows $\lambda = 1$ (Predictive Coding-style: $b_{i-1} \mapsto b_i$).

where $\Delta\pi \in \mathcal{K}_x$ is a "drift" in the space of solutions, $\lambda = \|x\|^2/\beta$, and $W_\perp^* = \lim_{t\to\infty} \bar{\Theta}(x, y; W_t)$ is the gradient descent solution.

We refer to Appendix A.1 for proofs and discussion. Because any deviation from $\bar{\Theta}$ (e.g., $\pi(W|x) \neq WP_x$) results in a drift $\Delta\pi$ within $\mathcal{M}_x^y$, we can design Θ to satisfy secondary objectives while still solving the primary optimization problem (Fig. 2a). As we will show in Section 3, this drift term is critical: it allows us to enact an "internal reorganization" of memories, for instance to counteract forgetting. Appendix C discusses how to extend Θ to recover standard optimization tweaks (e.g., momentum, weight decay, natural gradients).

2.2 Extension to Deep Learning

Consider an L-layer neural network with weights $\{W_i\}_{i=1,\dots,L}$ and activation functions ϕ_i. For notational simplicity, we omit biases (extensions are provided in Appendix C). Let the forward inference be $\hat{y} = \text{NNet}(x) = \phi_L \circ W_{L-1}\phi_{L-1} \circ \cdots \circ \phi_1 \circ W_0 x$. We formalize the layer-wise states at training step t as:

Forward Inference: $a_{i+1}^t = W_i\phi_i(a_i^t)$, where $\phi_0 = id(\cdot)$, $a_0^t = x^t$, and $a_L^t = \hat{y}^t$.

Here, a_i^t and $\phi_i(a_i^t)$ are the pre- and post-activation states. To learn, the network is guided by a set of layer-specific targets, commonly generated either from a top-down label signal (supervised) or based on label-free loss gradients (unsupervised). We define these targets as:

Targets: $\{b_i^t\}_{i=0,\dots,L-1}$, where $b_{L-1}^t = y^t$.

Following Appendix C.4, the i-th layer update at time t using our framework is simply defined as $W_i^{t+1} = \Theta(a_i^t, b_i^t; W_i^t)$. For a Hebb-like storage rule and exact inputs/targets ($a_i^t = a_i$, $b_i^t = b_i$), this converges to a solution $W_i^* \in \mathcal{M}_{a_i}^{b_i}$ provided $0 < \beta^{-1} < \min_t \frac{2}{||a_i^t||^2}$. For batched training (A_i^t, B_i^t), this extends linearly via expectation:

$$W_i^{t+1} = \mathbb{E}_{A_i^t, B_i^t}\left[\Theta(a_i^t, b_i^t; W_i^t)\right]$$

A subtle but crucial distinction between existing learning algorithms is whether they locally update weights based on the "pre-synaptic" forward activity a_i^t or the "post-synaptic" target of the previous layer b_{i-1}^t. We introduce a parameter $\lambda \in [0, 1]$ to smoothly interpolate between these modalities (see Fig. 2b and App. C.4).

Under specific storage, compression, and target rules, the SCT framework recovers established credit assignment algorithms. Specifically, Backpropagation (BP) (Linnainmaa, 1976; Rumelhart et al., 1986), Predictive Coding (PC) (Rao & Ballard, 1999; Whittington & Bogacz, 2017), and Target Learning (TL) (Bengio, 2014; Meulemans et al., 2020) all rely on the *HOP* operator (orthogonal projection $W_i P_{x_i}$ for compression; normalized Hebb rule $\beta^{-1} y_i x_i^T$ for storage), while they structurally differ for the interpolation factor λ and the targets definition b_i^t. In App. C.5 we report the conditions for this equivalence. We emphasize that this equivalence is not an approximation. SCT is a layer-wise update decomposition; however, the forward pass structure and the target definition provide arbitrary non-linear complexity.

It is worth observing that, in principle, compression and storage can be computed with respect to different inputs (breaking the gradient descent equivalence) and still obtain stable training. This is the case for Contrastive Hebbian Learning (Detorakis et al., 2019) where the network stores *clamped* dynamics while compressing on the *free* ones. In the following sections, we focus exclusively on the HOP operator, as it forms the backbone of standard backpropagation.

2.3 The Role of Compression in Generalization and Overfitting

Hebbian learning alone can hardly drive generalization without some form of compression (Agliari et al., 2024): it stores orthogonal patterns faithfully, but correlated data interfere in $W = \sum_t \beta^{-1} y^t x^{t\top}$. We therefore isolate the compression term and ask what it does to associations that are already stored. For a fixed batch, with $C_x = \mathbb{E}_j[x^j x^{j\top}]$ and $C_{yx} = \mathbb{E}_j[y^j x^{j\top}]$, the SCT dynamics has the exact solution

$$W^t = W^0 \Omega^t + C_{yx} C_x^+ \kappa_t, \qquad \Omega = \mathbb{I} - \beta^{-1} C_x, \qquad \kappa_t = \mathbb{I} - \Omega^t, \tag{3}$$

and at every horizon t the same weights are reached by compression alone, $W^t = \widetilde{W}^0(t)\,\Omega^t$, from a modified initialization (App. B.1). The second term reads the Hebbian associations through the metric $M_t = C_x^+ \kappa_t$,

$$W^t v = \mathbb{E}_j \left[y^j \, x^{j\top} M_t \, v \right],$$

so the layer answers a query with the stored targets, weighted by their alignment with the query measured in M_t. Compression therefore does not store associations: it builds the *geometry* in which they are read. M_t moves from the isotropic Hebbian metric ($\propto \mathbb{I}$) to the *sample geometry* C_x^+, the unique metric in which the stored patterns are orthonormal and each of them is recalled exactly.

Theorem 2.3. *Compression-only dynamics induces both generalization and overfitting. Driven by compression, the effective weights undergo a Baik-Ben Arous-Péché (BBP) phase transition: Initially, the network isolates the latent features of the data (generalization); eventually, it collapses into the sample geometry, where it fits the input's noisy components (overfitting).*

Formal statements and proofs are given in App. B.2. In Figure 3, the generalization, turnaround and overfitting phases are the passage of the gains of κ_t from the feature spikes to the edge of the noise bulk and through it; the separation of the signal eigenvalues from the bulk in Fig. 3(c) is akin to the classic BBP phenomena. A subcritical feature is never acquired ahead of the noise, so it can only be learned together with it. The peak $\lambda^*(t) = \beta/(1+t)$ of the band-pass G_t marks the cutoff of κ_t: modes above it have been whitened by the retrieval metric and modes below have not (Prop. B.3). Tracking the intersection of the filter peak $G^*_{peak}(t) = G_{\lambda^*}(t)$ with the empirical eigenvalues reveals the three distinct phases:

1. **Generalization:** As $G^*_{peak}(t)$ intersects the eigenvalues of the true hidden features (λ_h), the model isolates signal from noise ($\mathcal{L}_{noiseless} < \mathcal{L}_{noisy}$). Figure 3(b) visualizes these emerging eigen-features for MNIST.
2. **Turnaround:** When the filter reaches the boundary between the smallest feature eigenvalue (λ_m) and the largest noise eigenvalue ($\tilde{\lambda}_M$), performance begins to degrade.
3. **Overfitting:** As $G^*_{peak}(t)$ sweeps into the noise bulk, the network enters a pure overfitting regime ($\mathcal{L}_{noisy} < \mathcal{L}_{noiseless}$).

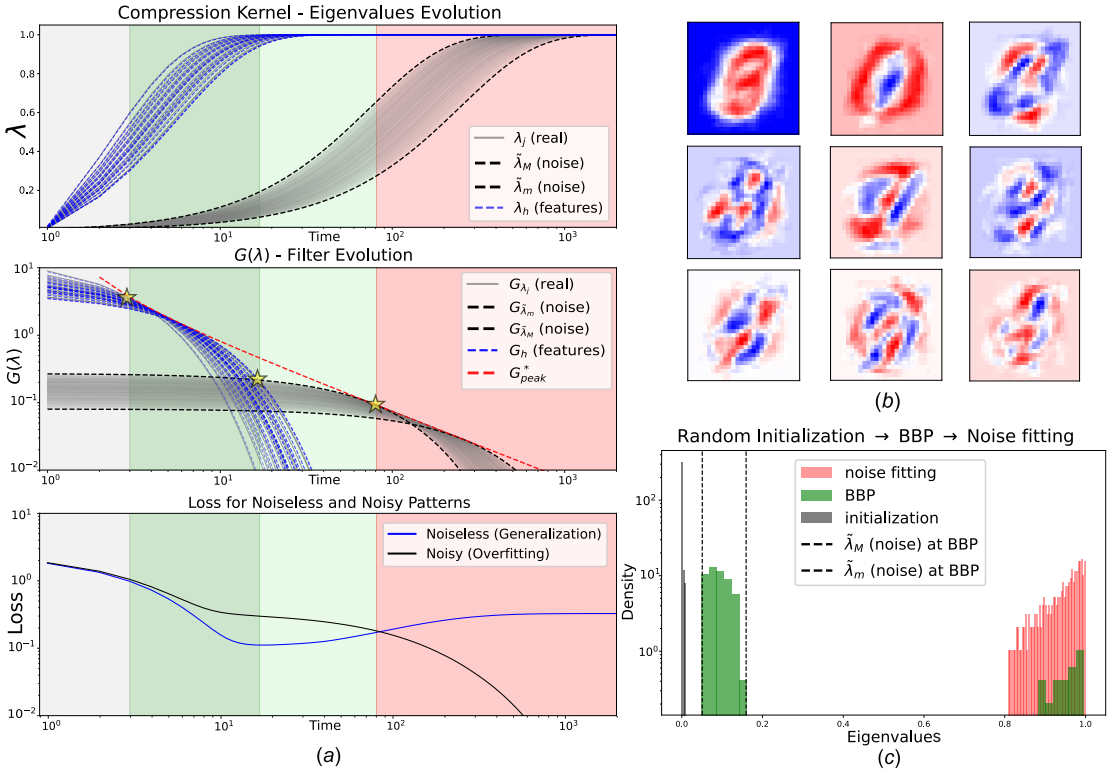


Figure 3: Compression drives generalization. Dashed lines for theoretical predictions (App. B.2) and solid lines for empirical values **(a)** Top: Eigenvalues of $\kappa_x(t)$. Middle: Spectral filters $G_j(t)$ tracking the moving peak $G^*(t)$ (red dashed line). Bottom: loss between the compressed input and the full noisy patterns $\mathcal{L}_{noisy}(t) = \mathbb{E}_j \|\kappa_x(t)x^j - x^j\|_2^2$ or their denoised version $\mathcal{L}_{noiseless}(t) = \mathbb{E}_j \|\kappa_x(t)x^j - \text{denoised}(x^j)\|_2^2$. The three phases of generalization and overfitting delineated by the filter's intersection with signal vs. noise eigenvalues. **(b)** Eigenvectors of the compression kernel for MNIST during the generalization phase. **(c)** BBP transition in the eigenvalues of $\kappa_x(t)$ during compression; signal eigenvalues (green) separate from the noise bulk (red).

Forgetting is the same geometry applied to old memories. When a layer holding W_{A} is trained on a new task B, $W^t x^\mu - W_{\mathrm{A}} x^\mu = (\widehat{W}_{\mathrm{B}} - W_{\mathrm{A}})\kappa_{\mathrm{B},t} x^\mu$ exactly, with $\widehat{W}_{\mathrm{B}}$ the least-squares solution of task B: old memories are re-read through B's compression kernel (App. B.3). Knowledge shared with B's features is overwritten while B generalizes; knowledge orthogonal to them decays only at the rate at which B acquires its noise bulk, and the interference on unrelated memories equals B's own variance. The onset of overfitting on B and the catastrophic forgetting of A's unrelated knowledge are one spectral event.

In a deep linear network trained by compression only, every layer's covariance is a spectral function of the input covariance (App. B.4): layers select bands of the input spectrum whose scale decreases harmonically with depth, $\lambda_\ell^* \approx \beta/(t\ell)$, deep stacks turn a single Hebbian write into ridge-regularized retrieval, and no depth can separate directions of equal variance. A nonlinear layer instead lifts the geometry compressed by the next layer to tensor powers and higher cumulants of its input (App. B.4): a direction that is invisible to every linear compression network, because the input covariance is isotropic, becomes an isolated eigenvalue of the covariance of a quadratic layer (App. B.4)..

3 Forgetting in Backpropagation

A central challenge when training neural networks on time-varying distributions is the tendency to overwrite previously learned patterns. Catastrophic forgetting occurs when newly learned associations destructively interfere with existing ones. In this section, we investigate the exact layer-wise

distortions driving forgetting in backpropagation and establish the theoretical conditions necessary to mitigate it using the SCT framework.

Proposition 3.1. *Consider a neural network layer update based on a new input-output pair (x, y). We wish to retain a previously stored memory (x^μ, y^μ) satisfying $Wx^\mu = y^\mu$. The impact of an update step $W^+ = \bar{\Theta}(x, y; W) + R$ (where R captures deviations from the exact HOP operator, such as regularizers or $\Delta\pi$) on the old memory is:*

$$W^+ x^\mu = y^\mu - \beta^{-1}(Wx - y)\langle x, x^\mu\rangle + Rx^\mu \tag{4}$$

This equation quantifies forgetting. It reveals that the memory output y^μ is perturbed proportionally to the error on the new sample $(Wx - y)$ and the geometric alignment $\langle x, x^\mu\rangle$ between the new input and the old memory. Forgetting occurs when $\langle x, x^\mu\rangle \neq 0$ and the new task is not perfectly learned. Therefore, to perfectly preserve the memory pattern while training on the new sample, the correction Rx^μ must compensate for the drift in (4). From this exact condition, we can mathematically deduce a profound limitation of standard continual learning approaches:

Theorem 3.2 (Requirements for Zero Forgetting)**.** *Under nonlinear representational reorganization, any compact static description of old geometry can become stale; exact layer-wise protection without sacrificing additional plasticity ultimately requires access to sufficient information about old inputs, either explicitly (replays) or through a generator (generative proxies).*

The formal statements are Thms. D.4 and D.6 in App. D. In particular, Thm. D.4 adds that corrections blind to the current sample are not forgetting-free in general. As a consequence, weight regularization alone cannot prevent forgetting, and once representations reorganize, a strict solution must rely on memory replay or an equivalent generative proxy such as a world model (Ha & Schmidhuber, 2018), in line with Khademi Nori & Kim (2024). Protecting a memory requires its input x^μ but not its target y^μ; only repairing drift that has already occurred needs the targets.

When scaling to multilayer networks, forgetting compounds with depth. Because the intermediate hidden representations $x_i^\mu(t)$ drift during training, their alignment with new data $\langle x_i^\mu(t), x_i(t)\rangle$ varies dynamically. The cumulative layer-wise forgetting drift can be defined as follows:

Proposition 3.3 (Forgetting Dynamics)**.** *Given a memory $(x^\mu, y^\mu) \in \mathcal{T}_A$ from task A, and a dataset $\mathcal{T}_B = \{(x_t, y_t)\}_{t=1,\dots,P}$ for task B. For an L-layer network $f_L(\cdot, \theta)$ trained with backpropagation on $\mathcal{T}_B$, the forgetting drift at the i-th layer $\delta_i = f_i(x^\mu; \theta^t) - x_i^\mu$ evolves over P steps as:*

$$\delta_i^P = -\beta^{-1} \sum_{t=1}^{P} \Big(\langle x_i^\mu, x_i^t\rangle + \langle \delta_{i-1}^{t-1}, x_i^t\rangle \Big) \left(\prod_{k=0}^{i+1} J_{\phi_{L-k}} W_{L-k}^T \right) (x_L^t - y_t)$$

See Appendix D.5 for details. Figure 7 depicts "memory recovery", demonstrating how numerically tracking and compensating for this integral during training successfully recovers past performance.

Beyond geometric alignment, we also identify a secondary driver of forgetting: noise.

Theorem 3.4 (Forgetting Under Random Gaussian Inputs)**.** *Noisy data also plays a diffusive role in forgetting. Under assumption D.9, random Gaussian inputs $x(t) \sim \mathcal{N}(\bar{x}, \epsilon \mathbb{I}_d)$ and a fixed target y^μ, a memory (x^μ, y^μ) with alignment $\omega = \langle x^\mu, \bar{x}\rangle$ experiences a diffusive drift, converging to the stationary distribution:*

$$y_{eq}^\mu \sim \mathcal{N}\left(\frac{\omega}{M_K} y^\mu, \frac{\rho^2 S_K}{2\beta M_{K+1}} \right)$$

where M_K, S_K, K, and ρ are architecture-dependent constants.

Proof in Appendix D.6. This analytical solution reveals how unstructured noise in a novel task's distribution can slowly erode stable memories, motivating the need for an elastic correction mechanism that actively pulls drifting memories back to their original state.

3.1 Correction of the HOP Operator

To counteract the layer-wise drift equation 4 that drives forgetting (Prop. 3.3), we modify the compression operator so that each layer's map on a set of memories is left unchanged and the update

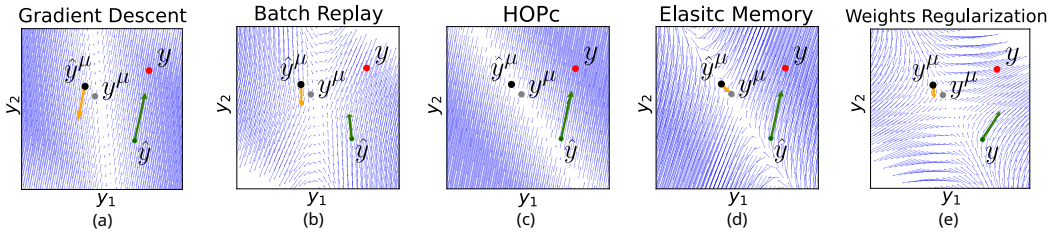


Figure 4: Update field for a 2-dim layer's output $\delta y_v = (\Theta(x, y; W) - W)v$. Green dot: current inference $\hat{y}$; Red dot: target y; Black dot: current memory inference $\hat{y}^\mu$; Gray dot: original, uncorrupted, memory y^μ. (a) Standard gradient descent update $\bar{\Theta}$. (b) Gradient descent with batch replay $\frac{1}{2}\bar{\Theta}(x, y; W) + \frac{1}{2}\bar{\Theta}(x^\mu, y^\mu; W)$. (c) HOPc tilts the null space of HOP so that memories are not affected, but it cannot recover from memory corruption. (d) HOPc with elastic memories, additional vector field in the null space. (e) Regularization based (elastic force in the weights), trade-off between learning y and recovering the original weights.

acts only along their components orthogonal to the new samples. For a memory x^μ with alignment $\alpha = \langle x^\mu, x\rangle$ to a new sample x, let $z_\perp^\mu = x^\mu - \frac{\alpha}{\|x\|^2}x$ be its private component. The correction

$$\Delta\pi_\epsilon(x^\mu) = \left(\frac{\alpha}{\beta}(\hat{y} - y) - \epsilon\,(Wx^\mu - y^\mu)\right)\frac{z_\perp^{\mu\top}}{\|z_\perp^\mu\|^2} \tag{5}$$

leaves the step on the current sample unchanged ($\Delta\pi_\epsilon\, x = 0$) and sets the memory's drift to $W^+x^\mu - y^\mu = (1-\epsilon)(Wx^\mu - y^\mu)$. For $\epsilon = 0$ it is the minimal forgetting-free drift of Thm. 2.2 (Prop. D.2): it protects the memory *without* its target y^μ, but only cancels the drift of the current step. The elastic term ($\epsilon > 0$) also repairs drift that has already accumulated, such as the diffusive drift of Thm. 3.4, and needs y^μ. In Fig. 4, the first term tilts the null space of the update away from the memories and the second adds a restoring field inside it. Ablating the elastic term removes most of the gains (App. E.3.1), so the drift accumulated over many updates matters.

What HOPc stores. A sample buffer spans only as many directions as it holds. We instead summarize a finished task at each layer l by the top-k eigenvectors $u_{l,j}$ (eigenvalues $s_{l,j}$) of the uncentred second moment of its inputs h_l, with their images $y_{l,j} = W_l u_{l,j}$ under the weights just learned. Since a weight change moves the old pre-activations by $\mathbb{E}\|\Delta W_l h_l\|^2 = \sum_j s_{l,j}\|\Delta W_l u_{l,j}\|^2$, keeping $W_l u_{l,j} = y_{l,j}$ preserves the layer's input–output map on past representations up to the discarded variance. The components of successive tasks are merged into one orthonormal basis per layer, and no sample or label is kept. The number k of protected directions sets the stability–plasticity boundary: new inputs largely share the old principal subspace, so protecting all of it freezes the network, whereas $k = 10$ per layer leaves the remaining directions free. The cost, $k(d_l + d_{l+1} + 1)$ floats per layer, does not grow with the number of tasks, samples or classes (15 to 53 sample-equivalents on our benchmarks, App. E.4), so sample efficiency scales with the intrinsic dimension of the latent space (Prop. D.7) rather than with the complexity of the data. Estimated from a small buffer, the same components do not help. The pseudocode is Alg. 2 (App. E.3.1).

HOPc and replay. HOPc preserves each layer's map on the dominant directions of past data, but it carries no information on where old and new classes, whose inputs share these directions, should be separated at the output; in class-incremental learning this boundary must be learned from old-class inputs in the loss, i.e. from replay or a generative proxy (Thm. 3.2). Because HOPc corrects the update rather than the loss, it composes with any replay: HOPc + ER and HOPc + DER++ apply it to the replayed batch of a sample buffer, and HOPc + Gaussian replay removes the samples altogether, replaying fresh draws from class Gaussians (mean and top covariance eigenvectors, storage-matched to the buffer; App. E.3), in the spirit of Pfülb & Gepperth (2021). Replay rehearses the old decisions; HOPc protects the representations they are read from.

Class-incremental learning. We evaluate on Split-CIFAR-10 (5 tasks of 2 classes) and Split-Tiny-ImageNet (10 tasks of 20 classes) in the setting of (de Haan et al., 2026), and on Split-MNIST in App. E.7. All methods (ER, DER++ (Buzzega et al., 2020), OGD (Farajtabar et al., 2020), Gaussian replay and the HOPc variants) share the learner and code path; hyperparameters are selected on a held-out validation split, and we report test accuracy over four fresh seeds under a stopping rule that

reads neither validation nor test data (App. E.5). HOPc's components are stored in addition to the buffer. Tab. 1 reports the final accuracy against the buffer size B: HOPc helps most when memory is scarce (HOPc + DER++ gains 40 points over DER++ on Split-CIFAR-10 at $B = 10$), and the gain shrinks as the buffer grows. A constraint alone, like OGD, stays far below replay.

Method	**CIFAR10**			**TinyImageNet**		
Joint (upper bound)	87.21 ± 0.13			66.14 ± 0.24		
SGD (lower bound)	19.76 ± 0.18			16.87 ± 0.2		
OGD (@ best buffer)	59.35 ± 1.74			10.66 ± 0.69		
buffer size B	10	500	1000	10	500	1000
Naive Replay (ER)	32.03 ± 5.06	77.48 ± 1.12	80.09 ± 0.27	12.41 ± 0.23	35.06 ± 0.29	42.49 ± 0.50
DER++	36.95 ± 4.18	80.91 ± 0.62	$\underline{82.73 \pm 0.23}$	15.55 ± 0.75	47.21 ± 0.24	50.80 ± 0.76
HOPc (no replay)	69.84 ± 1.05	80.61 ± 0.15	80.31 ± 0.03	$\underline{27.04 \pm 0.77}$	37.94 ± 0.79	38.19 ± 0.51
HOPc w/ DER++	$\underline{76.91 \pm 1.45}$	$\underline{82.10 \pm 0.33}$	82.70 ± 0.15	25.91 ± 3.42	$\underline{48.56 \pm 0.64}$	$\underline{52.26 \pm 0.28}$
HOPc w/ GR	$\mathbf{78.20 \pm 0.30}$	$\mathbf{84.95 \pm 0.15}$	$\mathbf{84.78 \pm 0.33}$	–	$\mathbf{57.26 \pm 0.48}$	$\mathbf{58.20 \pm 0.09}$

Table 1: Accuracy (%) on Split-CIFAR-10 and Split-TinyImageNet for buffer sizes 10 , 500 and 1000. Values are mean $\pm$ standard deviation. HOPc+GR is not defined for TinyImagenet and buffer 10 because that size is not sufficient to model all classes with at least one mode.

4 Discussion

We introduced the Storage–Compression–Targets (SCT) framework, which decomposes every layer-wise update into the storage of new associations and the compression of existing ones, and proved that backpropagation, predictive coding and target learning are instances of it (Sec. 2). In this lens compression builds the geometry in which stored associations are read: it drives generalization and then overfitting (Thm. 2.3), and the same kernel re-reads, and erases, old memories when a new task is learned (Thm. B.8). As deep linear networks (**?**Dominé et al., 2024) and random-feature models (Rahimi & Recht, 2007; Gerace et al., 2020) did for learning dynamics, compression dynamics offer a tractable handle on the rich regime, and SCT a common language in which credit-assignment algorithms can be compared and interpolated.

For backpropagation we characterized the exact layer-wise drift that erases past knowledge and what zero forgetting requires: once representations reorganize, exact protection at no extra cost in plasticity needs the past inputs, replayed or generated (Thm. 3.2). HOPc spends the free drift of the update on protecting past representations; unlike gradient projection (Farajtabar et al., 2020; Saha & Roy, 2023), it preserves the step in function space rather than the gradient in weight space (Prop. D.2). Its limits are the scale of our experiments and the lack of a dynamical memory selector based on an adaptive forgetting "risk"; tuning this risk dynamically would let the network move its own stability–plasticity boundary.

AI use statement

In this work, we used LLMs (ChatGPT and Claude) to refine the writing, draft layouts for schematic illustrations (which were later edited by the authors), and support implementation and downstream data analysis. The authors wrote this statement and minimally polished it with LLM assistance. They thoroughly reviewed all AI-assisted and AI-generated content and take full responsibility for the accuracy and integrity of the final content presented in the submitted paper.

Reproducibility statement

In the supplementary materials, we provide anonymized source code and instructions for reproducing the main results. For each experiment, we specify the hyperparameter settings used. To facilitate reproducibility, we document the implementation and the experimental setup in Appendix E.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, 19(50):1–34, 2018. URL http://jmlr.org/papers/v19/17-646.html.

Elena Agliari, Francesco Alemanno, Miriam Aquaro, Adriano Barra, Fabrizio Durante, and Ido Kanter. Hebbian dreaming for small datasets. *Neural Networks*, 173:106174, 2024. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2024.106174. URL https://www.sciencedirect.com/science/article/pii/S0893608024000984.

Alnur Ali, J. Zico Kolter, and Ryan J. Tibshirani. A continuous-time view of early stopping for least squares regression. In *International Conference on Artificial Intelligence and Statistics*, pp. 1370–1378, 2019.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pp. 254–263. PMLR, 2018.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International conference on machine learning*, pp. 322–332. PMLR, 2019.

Zhidong Bai, Baiqi Miao, and Guangming Pan. On asymptotics of eigenvectors of large sample covariance matrix. *The Annals of Probability*, 35(4):1532–1572, 2007.

Jinho Baik and Jack W. Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of Multivariate Analysis*, 97(6):1382–1408, 2006.

Jinho Baik, Gérard Ben Arous, and Sandrine Péché. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. 2005.

Florent Benaych-Georges and Raj Rao Nadakuditi. The singular values and vectors of low rank perturbations of large rectangular random matrices. *Journal of Multivariate Analysis*, 111:120–135, 2012.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *arXiv preprint arXiv:2006.11942*, 2020.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Marco C Campi and Simone Garatti. Compression, generalization and learning. *Journal of Machine Learning Research*, 24(339):1–74, 2023.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, volume 22, 2009.

Francesco D'Amico, Dario Bocchi, Luca Maria Del Bono, Saverio Rossi, and Matteo Negri. Pseudo-likelihood produces associative memories able to generalize, even for asymmetric couplings. *arXiv preprint arXiv:2507.05147*, 2025.

Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Sander de Haan, Yassine Taoudi-Benchekroun, Pau Vilimelis Aceituno, and Benjamin F Grewe. Continual learning through control minimization. *arXiv preprint arXiv:2602.04542*, 2026.

Georgios Detorakis, Travis Bartley, and Emre Neftci. Contrastive hebbian learning with random feedback weights. *Neural Networks*, 114:1–14, 2019.

Thang Doan, Mehdi Abbana Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In *International Conference on Artificial Intelligence and Statistics*, pp. 1072–1080. PMLR, 2021.

Clémentine CJ Dominé, Nicolas Anguita, Alexandra M Proca, Lukas Braun, Daniel Kunin, Pedro AM Mediano, and Andrew M Saxe. From lazy to rich: Exact learning dynamics in deep linear networks. *arXiv preprint arXiv:2409.14623*, 2024.

Noureddine El Karoui. The spectrum of kernel random matrices. *The Annals of Statistics*, 38(1): 1–50, 2010.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics*, pp. 3762–3773. PMLR, 2020.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Federica Gerace, Bruno Loureiro, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Generalisation error in learning with random features and the hidden manifold model. In *International Conference on Machine Learning*, pp. 3452–3462. PMLR, 2020.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hong Hu and Yue M. Lu. Universality laws for high-dimensional learning with random features. *IEEE Transactions on Information Theory*, 69(3):1932–1964, 2023.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Ta-Chu Kao, Kristopher T. Jensen, Gido M. van de Ven, Alberto Bernacchia, and Guillaume Hennequin. Natural continual learning: Success is a journey, not (just) a destination. In *Advances in Neural Information Processing Systems*, volume 34, pp. 28067–28079, 2021.

Milad Khademi Nori and Il-Min Kim. Task confusion and catastrophic forgetting in class-incremental learning: A mathematical framework for discriminative and generative modelings. *Advances in Neural Information Processing Systems*, 37:47678–47707, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is NP-hard. In *International Conference on Machine Learning*, pp. 5327–5337. PMLR, 2020.

Antti Knowles and Jun Yin. Anisotropic local laws for random matrices. *Probability Theory and Related Fields*, 169:257–352, 2017.

Vladimir Koltchinskii and Karim Lounici. Concentration inequalities and moment bounds for sample covariance operators. *Bernoulli*, 23(1):110–133, 2017.

Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160, 1976.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75 (4):667–766, 2022.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Alexander Meulemans, Francesco Carzaniga, Johan Suykens, João Sacramento, and Benjamin F Grewe. A theoretical framework for target propagation. *Advances in Neural Information Processing Systems*, 33:20024–20036, 2020.

Beren Millidge, Tommaso Salvatori, Yuhang Song, Rafal Bogacz, and Thomas Lukasiewicz. Predictive coding: Towards a future of deep learning beyond backpropagation? *arXiv preprint arXiv:2202.09467*, 2022.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Rohan Pandey. Gzip predicts data-dependent scaling laws. *arXiv preprint arXiv:2405.16684*, 2024.

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.

Debashis Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, 17(4):1617–1642, 2007.

Liangzu Peng, Paris V. Giampouras, and René Vidal. The ideal continual learner: An agent that never forgets. In *International Conference on Machine Learning*, pp. 27585–27610. PMLR, 2023.

Benedikt Pfülb and Alexander Gepperth. Overcoming catastrophic forgetting with gaussian mixture replay. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2021.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

R P Rao and D H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.*, 2(1):79–87, January 1999.

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.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Gobinda Saha and Kaushik Roy. Continual learning with scaled gradient projection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 9677–9685, 2023.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. Ieee, 2015.

Suzanne van der Veldt, Gido M. van de Ven, Sanne Moorman, and Guillaume Etter. Learning continually with representational drift. *arXiv preprint arXiv:2512.22045*, 2025.

James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5): 1229–1262, 2017.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Yi Yu, Tengyao Wang, and Richard J. Samworth. A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

A PROOFS ON SCT

A.1 CONTRACTION PROPERTY AND EQUIVALENCE TO GRADIENT DESCENT

We will now verify the fundamental property of this formulation, proving lemma 2.1: and theorem 2.2. We start with the operator

$$\Theta(x, y; W) = H(x, y) + \pi(W|x)$$

Where π is some function that kernelizes W with respect to $x \in \mathbb{R}^{\kappa}$. We define the matrix subspace $\mathcal{M}_x^y \subset \mathbb{R}^{m \times n}$ as the space of matrices such that $W \in \mathcal{M}_x^y \implies Wx = y$. Let us start by assuming $H(x, y) = \frac{yx^T}{\beta}$ and the following compression function to prove Lemma 2.1:

$$\pi(W|x) = \bar{\pi}(W/x) = W(\mathbb{I} - \frac{xx^T}{\beta}) = W(\mathbb{I} - C_x) = WP_x$$

Every W^* s.t. $W^*x = y$ is then an equilibrium for the storage-compression parametrization

$$\Theta(x, y; W^*) = \frac{yx^T}{\beta} + W^*P_x = W^*\frac{xx^T}{\beta} + W^*(\mathbb{I} - \frac{xx^T}{\beta}) = W^*$$

In general, if W is s.t. $Wx = \hat{y}$, this is equivalent to a contraction towards the $\mathcal{M}_x^y$ matrices:

$$\Theta(x, y; W) = \frac{yx^T}{\beta} + WP_x = \frac{yx^T}{\beta} + W(\mathbb{I} - \frac{xx^T}{\beta}) = W - \frac{1}{\beta}(Wx - y)x^T = W - \frac{1}{\beta}\frac{\partial \mathcal{L}}{\partial W}$$

where

$$\mathcal{L} = \frac{1}{2}||\hat{y} - y||^2$$

Now, to prove theorem 2.2, we need to do the same for a general compression function π and a storage function $H(x, y)$:

$$\Theta(x, y; W) = H(x, y) + \pi(W|x)$$

To be precise, what we are looking for is a *transverse contraction* toward $\mathcal{M}_x^y$. As $H(x, y)x = y$ we can always rewrite it like:

$$H(x, y) = \frac{yx^T}{\|x\|^2} + \sum_j^{\rho-1} o_j r_j^T$$

where ρ is the rank of $H(x, y)$, $o_j \in \mathbb{R}^m$ and $r_j \in \mathbb{R}^{\kappa}$ defines a $\rho - 1$ dimensional basis of $R^{\perp} \simeq \mathbb{R}^{\rho-1} \subseteq \text{span}\{x\}^{\perp} \simeq \mathbb{R}^{n-1}$. We can include these extra-terms in $\Delta\pi$ as $\sum_j^{\rho-1} o_j r_j^T x = 0$ by construction.

Given $\bar{\pi}(W/x) = WP_x$ and $\hat{y} = Wx$, we can always define

$$W = WC_x + W - WC_x = \frac{1}{\|x\|^2}\hat{y}x^T + \bar{\pi}(W/x)$$

For $e = \hat{y} - y$ we get

$$\Theta(x, y; W) = \frac{yx^T}{\|x\|^2} + \pi(W|x) = \frac{(\hat{y} - e)x^T}{\|x\|^2} + \pi(W|x)$$

$$\frac{\hat{y}x^T}{\|x\|^2} - \frac{ex^T}{\|x\|^2} + \pi(W|x) = (W - \bar{\pi}(W/x)) - \frac{ex^T}{\|x\|^2} + \pi(W|x)$$

$$\Theta(x, y; W) - W = -\frac{ex^T}{\|x\|^2} + (\pi(W|x) - \bar{\pi}(W|x))$$

Finally, if we define $\mathcal{L} = \frac{1}{2}\|Wx - y\|^2$ and relax the operator ($\|x\|^2 \to \beta$) get

$$\Delta W = \Theta(x, y; W) - W = -\frac{1}{\beta}\frac{\partial \mathcal{L}}{\partial W} + \Delta\pi$$

Notice that $\Delta\pi x = 0$ therefore if $W^* \in \mathcal{M}_x^y \implies W^* + \Delta\pi \in \mathcal{M}_x^y$. So we get a contraction toward $\mathcal{M}_x^y$ plus a tangent term. Being $\mathcal{M}_x^y$ a linear subspace of $\mathbb{R}^{\dim(y)\times\dim(x)}$, we can define π so that $\Theta(x, y; W)$ contracts towards all possible elements of $\mathcal{M}_x^y$. If $\Theta(x, y; W)$ is exact, there exists π such that we can map W to all the matrix $W^* \in \mathcal{M}_x^y$.

The distance to the solution manifold is $\operatorname{dist}(W, \mathcal{M}_x^y) = \|Wx - y\|/\|x\|$, and for every admissible $H \in \mathcal{M}_x^y$, $\pi \in \mathcal{K}_x$,

$$W^+x = \lambda y + (1 - \lambda)Wx \implies \operatorname{dist}(W^+, \mathcal{M}_x^y) = |1 - \lambda| \operatorname{dist}(W, \mathcal{M}_x^y), \qquad \lambda = \|x\|^2/\beta.$$

So the update contracts transversally to $\mathcal{M}_x^y$ at exact rate $|1 - \lambda|$, and the drift $\Delta\pi$ leaves the transverse coordinate invariant since $\Delta\pi\, x = 0$: the decomposition is a contraction across the manifold composed with free motion along it. The admissible step size is $0 < \beta^{-1} < 2/\|x\|^2$, with exact projection at $\lambda = 1$ ($\beta = \|x\|^2$), recovering the exactness condition stated for HOP; in the batched case the per-mode rate is $(1 - \eta\lambda_j)$ with $\eta < 2/\lambda_{\max}(C_x)$, so the retention filter P^t of App.**??** is precisely the accumulated contraction factor.

B Compression as Retrieval Geometry

One idea runs through this appendix: storage writes an association once, and compression builds the metric in which the stored associations are read. For a fixed batch, the SCT trajectory is a kernel readout over the stored targets, in a metric that is a spectral function of the input covariance (App. B.1). The metric moves from the isotropic Hebbian geometry, through a *feature geometry* in which only the latent features are whitened, to the *sample geometry* in which every stored pattern is an orthonormal memory (App. B.2). The same kernel re-reads old memories when a new task is learned (App. B.3). Linear depth only reweights one covariance, whereas nonlinear layers lift it to tensor powers and higher cumulants (App. B.4). The single-layer filter is the classical Landweber (early-stopping) filter (Yao et al., 2007; Ali et al., 2019); what is new is its reading as a retrieval geometry and its consequences. Table 2 lists where each claim of Sec. 2.3 is proved.

Claim in Sec. 2.3	**Result**
Exact solution, compression-only gauge, retrieval metric	Prop. B.1, Def. B.2
The sample geometry recalls every stored pattern exactly	Prop. B.3
Feature phase iff BBP; bounded risk; divergent sample phase (Thm. 2.3)	Thm. B.6, Fig. 5
Forgetting re-reads old memories through the new kernel	Thm. B.8
Linear depth: harmonic bands, ridge retrieval, no new directions	Thm. B.9
Nonlinear lift of the next layer's geometry; ReLU gating	Thm. B.10, Prop. B.11

Table 2: Where the claims of Sec. 2.3 are proved.

Notation. Stored inputs and targets are the columns of $X \in \mathbb{R}^{n\times P}$ and $Y \in \mathbb{R}^{m\times P}$; $C = XX^\top/P$, $C_{yx} = YX^\top/P$, Π projects onto $\operatorname{range}(C)$ and $\widehat{W} = C_{yx}C^+$ is the least-squares (minimum-norm) solution. With step $\eta = \beta^{-1}$ and $\Omega = \mathbb{I} - \eta C$,

$$\kappa_t = \mathbb{I} - \Omega^t = k_t(C), \qquad k_t(\lambda) = 1 - \omega_t(\lambda), \qquad \omega_t(\lambda) = (1 - \eta\lambda)^t, \tag{6}$$

where $0 < \eta\lambda_{\max}(C) < 1$, so that $k_t \in [0,1]$ is increasing in λ; the kernel $\kappa_x(t)$ of Fig. 3 is κ_{t+1}. We write $C = \sum_j \lambda_j \hat{u}_j \hat{u}_j^\top$.

B.1 Exact solution and retrieval geometry

Proposition B.1 (Exact solution and compression-only gauge)**.** *Let $W_{t+1} = H_t + W_t\Omega_t$ with $\Omega_t = \mathbb{I} - \eta_t C_t$, and $\Phi_{s,t} = \Omega_s \cdots \Omega_t$ ($\Phi_{t+1,t} = \mathbb{I}$).*

(a) $W_T = W_0\Phi_{0,T-1} + \sum_{k<T} H_k\Phi_{k+1,T-1}$. If every Ω_k is invertible, $W_T = \widetilde{W}_0(T)\,\Phi_{0,T-1}$ with $\widetilde{W}_0(T) = W_0 + \sum_{k<T} H_k\Phi_{0,k}^{-1}$: at every horizon, a storage–compression trajectory is a compression-only trajectory from a modified initialization.

(b) For HOP on a fixed batch ($C_t = C$, $H_t = \eta C_{yx}$), also when C is singular,

$$W_T = \underbrace{W_0\,\Omega^T}_{\textit{retention}} + \underbrace{\widehat{W}\kappa_T}_{\textit{acquisition}}, \qquad \widetilde{W}_0(T) = W_0 + \widehat{W}\big(\Omega^{-T} - \mathbb{I}\big). \tag{7}$$

The same holds for $\mathbb{E}[W_T]$ under SGD with independent minibatches drawn with replacement.

Proof. (a) Induction on $W_{T+1} = H_T + W_T\Omega_T$, and $\Phi_{k+1,T-1} = \Phi_{0,k}^{-1}\Phi_{0,T-1}$. (b) On $\mathrm{range}(C)$, $\eta\sum_{k<T}\Omega^k = \kappa_T C^{-1}$; on $\ker C = \ker X^\top$ the sum is $T\,\mathbb{I}$, but C_{yx} vanishes there. Hence $\eta C_{yx}\sum_{k<T}\Omega^k = \widehat{W}\kappa_T$, and $\widetilde{W}_0(T) = W_T\Omega^{-T}$. For SGD, H_k is independent of the later Ω_j, so $\mathbb{E}[H_k\Phi_{k+1,T-1}] = \eta C_{yx}\Omega^{T-1-k}$. □

The equivalence is pointwise in T and has a cost, since $\|\widetilde{W}_0(T)\|$ grows like $(1-\eta\lambda_{\max})^{-T}$. In the time-varying case the survival operators $\Phi_{k+1,T-1}$ drive the dynamics, and forgetting accumulates through them (Eq. equation 11, App. D); the factorization equation 7 needs a stationary batch, whose eigenbasis is reweighted and never rotated.

Definition B.2 (Retrieval metric)**.** A layer $W = C_{yx}M$ has *retrieval metric* $M \succeq 0$: it answers a query v with the stored targets, weighted by their alignment with v measured in M,

$$Wv = \tfrac{1}{P}\sum_i y_i\, x_i^\top M v. \tag{8}$$

A *spectral* metric is $M = C^+ s(C)$ with a gain $s : [0,\infty) \to [0,1]$, $s(0) = 0$. By equation 7, the SCT trajectory from $W_0 = 0$ has $M_t = C^+\kappa_t$ and gain $s = k_t$: the eigenvalues of κ_t in Fig. 3(a) are the gains of the retrieval geometry. A deep compression-only stack builds the ridge gain $\lambda/(\lambda+\rho)$ from a single Hebbian write (Thm. B.9(c)).

Proposition B.3 (The two endpoints)**.** *(a)* Sample geometry. *If X has full column rank, $X^\top C^+ X/P = \mathbb{I}_P$, so $\widehat{W}x_j = y_j$ for every j; and $C_{yx}MX = Y$ for every Y iff $\Pi M\Pi = C^+$. On the span of the data, C^+ is the only metric in which the stored patterns are orthonormal, i.e. retrieved as individual memories.*

(b) Feature geometry. *If $x = F\alpha + \xi$ and $y = L\alpha$ with $\alpha \sim (0, \mathbb{I}_r)$ and $\xi \sim (0, \sigma^2\mathbb{I})$, then $\Sigma = FF^\top + \sigma^2\mathbb{I}$, and the population association read in Σ^{-1} is the linear MMSE predictor $W^* = LF^\top\Sigma^{-1} = L(\sigma^2\mathbb{I} + F^\top F)^{-1}F^\top$.*

(c) The path. *$k_t(\lambda)/\lambda \to \eta t$ as $\eta t\lambda \to 0$ and $\to 1/\lambda$ as $\eta t\lambda \to \infty$: $M_t \approx \eta t\,\Pi$ (Hebbian) early and $M_t \to C^+$ late, and at time t whitening is applied to the modes above $\lambda_{\mathrm{cut}}(t) = 1/(\eta t)$ and withheld below. The peak $\lambda^*(t) = \beta/(1+t)$ of the band-pass $G_t(\lambda) = \lambda\,\omega_t(\lambda)$ tracked in Fig. 3 is this cutoff, since $k_t(\lambda^*) \to 1 - e^{-1}$.*

Proof. (a) With $X = UDV^\top$, $U^\top U = \mathbb{I}_P$, $D \succ 0$: $C^+ = PUD^{-2}U^\top$ and $X^\top C^+ X/P = \mathbb{I}_P$. Conversely, $C_{yx}MX = Y$ for all Y iff $X^\top(\Pi M\Pi)X = P\,\mathbb{I}_P$, which determines $\Pi M\Pi$ because X maps $\mathbb{R}^P$ bijectively onto $\mathrm{range}(X)$. (b) Push-through identity. (c) Expand $1-(1-\eta\lambda)^t$; $\partial_\lambda[\lambda\,\omega_t(\lambda)] = 0$ at $\eta\lambda^* = 1/(1+t)$, where $\omega_t = (t/(1+t))^t \to e^{-1}$. □

B.2 GENERALIZATION BEFORE MEMORIZATION

Assumption B.4 (Spiked latent-feature model)**.** The columns of X are i.i.d. $\mathcal{N}(0,\Sigma)$ with $\Sigma = \sigma^2(\mathbb{I} + \sum_{h\le r}\rho_h u_h u_h^\top)$, orthonormal u_h, fixed r and $\rho_h > 0$. Targets are $y = W^* x + \varepsilon$ with $W^* = \sum_h w_h u_h^\top$ and $\varepsilon \sim \mathcal{N}(0,\Sigma_\varepsilon)$ independent of x, and $n, P \to \infty$ with $n/P \to \gamma \neq 1$. μ_{MP} is the Marchenko–Pastur law of ratio γ and scale σ^2 (with an atom $1-1/\gamma$ at 0 if $\gamma > 1$) and edges $\lambda_\pm = \sigma^2(1 \pm \sqrt{\gamma})^2$.

This is the latent model of Prop. B.3(b) with $f_h = \sigma\sqrt{\rho_h}\, u_h$ and $y = \sum_h b_h \alpha_h$, for which $w_h = b_h\sqrt{\rho_h}/(\sigma(1+\rho_h))$ and $\Sigma_\varepsilon = \sum_h b_h b_h^\top/(1+\rho_h)$. The targets depend on the features, so fresh samples can be predicted better than the stored ones are recalled.

Lemma B.5 (Spectral measures)**.** *Let $\mu_v = \sum_j \langle v, \hat{u}_j\rangle^2 \delta_{\lambda_j}$ be the spectral measure of a unit vector v in C. Almost surely: (i) if $\rho_h > \sqrt{\gamma}$, C has an eigenvalue $\hat{\lambda}_h \to \sigma^2(1+\rho_h)(1+\gamma/\rho_h) > \lambda_+$ with $\langle \hat{u}_h, u_h\rangle^2 \to q_h = (1-\gamma/\rho_h^2)/(1+\gamma/\rho_h)$, and otherwise we set $q_h = 0$; all other eigenvalues lie asymptotically in $\{0\} \cup [\lambda_-, \lambda_+]$; (ii) $\mu_{u_h} \to \mu_h = q_h \delta_{\hat{\lambda}_h} + \nu_h$, where ν_h has mass $1-q_h$ on $\{0\} \cup [\lambda_-, \lambda_+]$ and kernel atom $\nu_h(\{0\}) = \pi_h = (\gamma-1)_+/(\gamma+\rho_h)$; (iii) $\mu_v \to \mu_{\mathrm{MP}}$ for unit $v \perp u_1, \dots, u_r$, and $v^\top g(C) v' \to 0$ for bounded continuous g and orthonormal v, v', each a feature direction or orthogonal to all of them.*

Proof. (i) is the BBP transition (Baik et al., 2005; Baik & Silverstein, 2006; Paul, 2007; Benaych-Georges & Nadakuditi, 2012). For (ii)–(iii), quadratic forms of the resolvent in deterministic vectors follow the deterministic equivalent $(C-z)^{-1} \asymp -z^{-1}(\mathbb{I} + \tilde{m}(z)\Sigma)^{-1}$ (Bai et al., 2007; Knowles & Yin, 2017), where $\tilde{m}$ is the Stieltjes transform of the limiting law of $X^\top X/P$. Since Σ is diagonal in a basis containing the u_h, $\int(\lambda - z)^{-1} d\mu_h = -[z(1+\sigma^2(1+\rho_h)\tilde{m}(z))]^{-1}$ and the off-diagonal forms vanish. The kernel atom is $\lim_{z\to 0}(-z)\int(\lambda - z)^{-1} d\mu_h = [1+\sigma^2(1+\rho_h)\tilde{m}(0)]^{-1}$, with $\tilde{m}(0) = [\sigma^2(\gamma-1)]^{-1}$ for $\gamma > 1$. □

Theorem B.6 (Feature geometry before sample geometry; formal version of Thm. 2.3)**.** *Under Assumption B.4, let $\widehat{W}_s = C_{yx} C^+ s(C)$ for a continuous gain s (the SCT trajectory from $W_0 = 0$ is $s = k_t$), and let $\mathcal{R}(s) = \mathbb{E}\|\widehat{W}_s x - y\|^2$ for a fresh pair (x, y).*

(a) Localization. *κ_t has an eigenvalue above the image $[0, k_t(\lambda_+)]$ of the noise bulk, carrying a fraction q_h of feature h, iff $\rho_h > \sqrt{\gamma}$. For gradient flow ($k_\tau(\lambda) = 1 - e^{-\tau\lambda}$) the largest gap is $D(R_h) = R_h^{-1/(R_h-1)}(1 - R_h^{-1})$, $R_h = \hat{\lambda}_h/\lambda_+$, which vanishes at the threshold. A subcritical feature is never acquired faster than the fastest noise mode: $\limsup u_h^\top \kappa_t u_h \le k_t(\lambda_+)$.*

(b) Exact risk. *For any design X, averaging over the training noise,*

$$\mathcal{R}(s) = \operatorname{tr}\Sigma_\varepsilon + \underbrace{\operatorname{tr}\big(W^*(\mathbb{I} - s(C))\,\Sigma\,(\mathbb{I} - s(C))W^{*\top}\big)}_{\mathcal{B}(s)} + \underbrace{\tfrac{1}{P}\operatorname{tr}(\Sigma_\varepsilon)\operatorname{tr}\big(\Sigma\, s(C)^2 C^+\big)}_{\mathcal{V}(s)}, \quad (9)$$

and almost surely $\mathcal{B} \to \sigma^2 \sum_h \|w_h\|^2 \big[\langle (1-s)^2\rangle_h + \rho_h \langle 1 - s\rangle_h^2\big]$ and $\mathcal{V} \to \operatorname{tr}(\Sigma_\varepsilon)\,\sigma^2\gamma \int_{(0,\infty)} s(\lambda)^2 \lambda^{-1} d\mu_{\mathrm{MP}}(\lambda)$, where $\langle g\rangle_h = \int g\, d\mu_h$.

(c) Feature phase. *If $s \ge 1 - \varepsilon$ at every supercritical $\hat{\lambda}_h$ and $s \le \delta$ on $[0, \lambda_+]$, then asymptotically $(1-\delta)^2\mathcal{B}_* \le \mathcal{B} \le \mathcal{B}_* + \sigma^2 \sum_h \|w_h\|^2 q_h(1+2\rho_h)\,\varepsilon$, with $\mathcal{B}_* = \sigma^2 \sum_h \|w_h\|^2 \big[(1-q_h) + \rho_h(1-q_h)^2\big]$, and $\mathcal{V} \le \delta^2 \mathcal{V}_\infty$; if moreover $s(\lambda) \le c\lambda$ (for $s = k_t$, $c = \eta t$), then $\mathcal{V} \le \operatorname{tr}(\Sigma_\varepsilon)\,\gamma\,\sigma^4 c^2$.*

(d) Sample phase. *As $t \to \infty$, $\widehat{W}_{k_t}$ tends to the least-squares ($\gamma < 1$) or minimum-norm interpolating ($\gamma > 1$) solution, and $\mathcal{R} \to \operatorname{tr}\Sigma_\varepsilon + \sigma^2 \sum_h \|w_h\|^2(\pi_h + \rho_h \pi_h^2) + \mathcal{V}_\infty$, with $\mathcal{V}_\infty = \operatorname{tr}(\Sigma_\varepsilon)\,\gamma/(1-\gamma)$ for $\gamma < 1$ and $\operatorname{tr}(\Sigma_\varepsilon)/(\gamma-1)$ for $\gamma > 1$.*

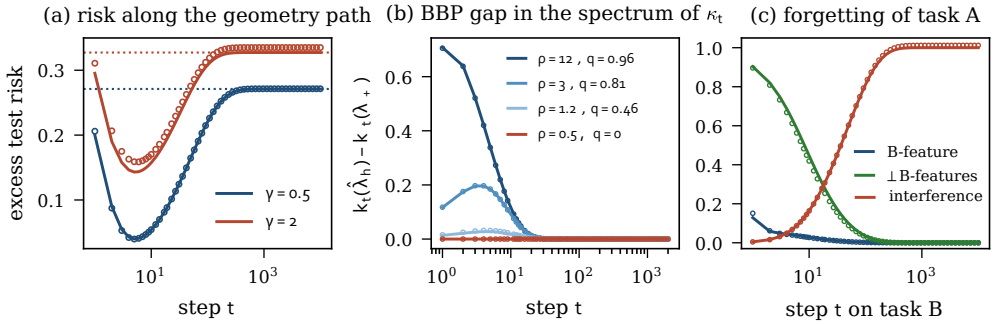


Figure 5: Theory (lines) against simulation (markers; $n = 900$, $\sigma^2 = 1$, $\eta = 0.9/\lambda_{\max}$). **(a)** Excess test risk along the SCT trajectory, Thm. B.6(b), for $\gamma = 0.5$ and $\gamma = 2$ ($\rho = (8, 3)$); dotted lines: sample-phase limits, Thm. B.6(d). **(b)** Gap between the eigenvalue of κ_t carried by a feature and the image of the bulk edge ($\gamma = 0.5$, $\sqrt{\gamma} \approx 0.71$); it vanishes for the subcritical feature. **(c)** Forgetting of a task-A map while training on task B ($\gamma_B = 0.5$): retention along a B-feature, $\langle \omega_t \rangle_h$, and orthogonal to B's features, $1 - b_B(t)$, and interference on generic memories, normalized by $\mathcal{V}_{B,\infty}$ (Thm. B.8).

Hence, if all features are supercritical, the SCT trajectory passes through a feature phase with excess risk $\mathcal{B}_* + O(\varepsilon)$ plus a variance that stays bounded as $\gamma \to 1$ (for $\rho_h > 1$), whereas its sample phase has excess risk at least $\mathcal{V}_\infty$, which diverges at the interpolation threshold. A subcritical feature has $q_h = 0$: it contributes its full null risk to $\mathcal{B}_*$ and can only be learned together with the noise bulk.

Proof. (a) $\kappa_t = k_t(C)$ has eigenvalues $k_t(\lambda_j)$ and k_t is increasing, so Lemma B.5(i) gives the criterion; maximizing $e^{-\tau\lambda_+} - e^{-\tau\hat{\lambda}_h}$ over τ gives D. By Lemma B.5(ii), $u_h^\top \kappa_t u_h \to q_h k_t(\hat{\lambda}_h) + \int k_t \, d\nu_h$, and $0 \le k_t \le k_t(\lambda_+)$ on the support of ν_h. (b) Write $Y = W^* X + E$. As $s(C)$ vanishes on $\ker C$, $C_{yx} C^+ s(C) = W^* s(C) + N$ with $N = EX^\top C^+ s(C)/P$, $\mathbb{E}N = 0$ and $\mathbb{E}[N\Sigma N^\top] = P^{-1} \operatorname{tr}(\Sigma s^2 C^+) \, \Sigma_\varepsilon$, which gives equation 9. Expanding Σ, the forms $u_h^\top g(C) u_{h'}$ tend to $\delta_{hh'} \langle g \rangle_h$ (Lemma B.5). In $\mathcal{V}$, the rank-r part of Σ contributes $O(1/P)$ because the non-zero spectrum stays above $\lambda_-/2$, and the rest converges to the Marchenko–Pastur integral. (c) On the support of ν_h, $1 - s \in [1 - \delta, 1]$, and at $\hat{\lambda}_h$, $1 - s \le \varepsilon$; inserting the resulting ranges of $\langle 1 - s \rangle_h$ and $\langle (1 - s)^2 \rangle_h$ in (b) gives the bias bounds. For $\mathcal{V}$, use $s^2 \le \delta^2$ on the bulk together with $\sigma^2 \gamma \int_{(0,\infty)} \lambda^{-1} d\mu_{\mathrm{MP}} = \gamma/(1 - \gamma)$ or $1/(\gamma - 1)$; or use $s^2/\lambda \le c^2 \lambda$ and $\int \lambda \, d\mu_{\mathrm{MP}} = \sigma^2$. (d) On the spectrum, which has a gap at 0, $k_t \to \mathbf{1}_{(0,\infty)}$, so (b) applies with $\langle 1 - s \rangle_h = \langle (1 - s)^2 \rangle_h = \pi_h$. For the final claims: when $\rho_h > 1$, R_h stays above 1 as $\gamma \to 1$, so a fixed window (ε, δ) remains admissible at a bounded time and (c) applies with bounded c. □

Remark B.7 (Reading and scope). Theorem B.6 holds for every spectral gain. By Prop. B.1, κ_t is also the response that compression alone removes from a stored map, and a deep compression-only stack produces the ridge gain with ρ in the role of $1/(\eta t)$. An initialization W_0 enters only through $(W^* - W_0)(\mathbb{I} - s(C))$ in the bias: along each mode it is erased at the rate at which the mode is acquired. Part (a) concerns localization, not recoverability: through C_{yx} a subcritical feature is still partly predictable by optimal shrinkage, but never ahead of the noise.

B.3 Forgetting: old memories are re-read through the new geometry

Let a layer that holds a map W_{A} be trained on a batch of task B. The retention channel of equation 7 now carries task A.

Theorem B.8 (Forgetting through the compression kernel). *Let $\widehat{W}_{\mathrm{B}} = C_{yx}^{\mathrm{B}} C_{\mathrm{B}}^+$ and $\kappa_{\mathrm{B},t} = k_t(C_{\mathrm{B}})$.*

(a) Identity. *The forgetting of an old input x is*

$$F_t(x) = (W_t - W_{\mathrm{A}})\, x = \big(\widehat{W}_{\mathrm{B}} - W_{\mathrm{A}}\big)\, \kappa_{\mathrm{B},t}\, x. \tag{10}$$

Only the component of x in the span of the new data can be forgotten, and inputs in $\ker C_{\mathrm{B}}$ never are. In B's feature phase, $\kappa_{\mathrm{B},t}$ is within $\max(\varepsilon, \delta)$ of the projector on B's r_{B} spike eigenvectors; in its sample phase it tends to the projector on $\operatorname{range} C_{\mathrm{B}}$. Without storage on B, $F_t(x) = -W_{\mathrm{A}}\kappa_{\mathrm{B},t}x$: compression alone erases what storage would overwrite.

(b) Three channels. *Let task B satisfy Assumption B.4, with features u_h and target map $\sum_h w_h u_h^\top$, and let $W_{\mathrm{A}} = \sum_a e_a v_a^\top + \sum_h a_h u_h^\top$ with orthonormal e_a and orthonormal $v_a \perp u_1, \dots, u_r$. In probability:*

- *(i) knowledge shared with a B-feature is overwritten as B acquires that feature, $W_t u_h \to a_h + (w_h - a_h)\langle k_t\rangle_h$, where $\langle k_t\rangle_h \geq q_h(1-\varepsilon)$ in B's feature phase;*
- *(ii) knowledge orthogonal to B's features decays only as B acquires its noise bulk, $W_t v_a \to (1 - b_{\mathrm{B}}(t))\, e_a$ with $b_{\mathrm{B}}(t) = \int k_t\, d\mu_{\mathrm{MP}} \leq k_t(\lambda_+^{\mathrm{B}})$, and it reaches $(1 - 1/\gamma_{\mathrm{B}})_+\, e_a$ as $t \to \infty$;*
- *(iii) on a memory orthogonal to B's features with $\|x\|^2 = n\zeta^2$, the interference due to B's label noise, $\mathbb{E}\|F_t(x)\|^2 - \|(W_{\mathrm{B}}^* - W_{\mathrm{A}})\kappa_{\mathrm{B},t}x\|^2$, tends to $(\zeta^2/\sigma^2)\, \mathcal{V}_{\mathrm{B}}(t)$, task B's own variance in Thm. B.6.*

The onset of overfitting on B and the erasure of A's knowledge orthogonal to B's features are therefore one spectral event: $\kappa_{\mathrm{B},t}$ entering the bulk of C_{B} (Fig. 5(c)). Generalizing on B is cheap for A; memorizing B is not.

Proof. (a) By equation 7 with $W_0 = W_{\mathrm{A}}$, $W_t - W_{\mathrm{A}} = (\widehat{W}_{\mathrm{B}} - W_{\mathrm{A}})\kappa_{\mathrm{B},t}$; $\kappa_{\mathrm{B},t}$ vanishes on $\ker C_{\mathrm{B}}$, and in the feature phase its eigenvalues are at least $1 - \varepsilon$ on the spikes and at most δ elsewhere. (b) Write $\widehat{W}_{\mathrm{B}}\kappa_{\mathrm{B},t} = W_{\mathrm{B}}^*\kappa_{\mathrm{B},t} + N_{\mathrm{B}}\kappa_{\mathrm{B},t}$ with $N_{\mathrm{B}} = E_{\mathrm{B}}X_{\mathrm{B}}^\top C_{\mathrm{B}}^+/P_{\mathrm{B}}$. The second term has second moment $\operatorname{tr}(\Sigma_\varepsilon^{\mathrm{B}})\, x^\top \kappa_{\mathrm{B},t}^2 C_{\mathrm{B}}^+ x/P_{\mathrm{B}}$, which is $O(1/P_{\mathrm{B}})$ for unit x and gives (iii) for $\|x\|^2 = n\zeta^2$ by Lemma B.5(iii). By Lemma B.5, $u_{h'}^\top g(C_{\mathrm{B}})u_h \to \delta_{hh'}\langle g\rangle_h$ and $v_a^\top g(C_{\mathrm{B}})v_{a'} \to \delta_{aa'}\int g\, d\mu_{\mathrm{MP}}$, and all mixed forms vanish; this gives (i) and (ii), with $b_{\mathrm{B}}(\infty) = \mu_{\mathrm{MP}}((0,\infty)) = \min(1, 1/\gamma_{\mathrm{B}})$. □

B.4 Depth and nonlinearity

In a deep network each layer compresses against the covariance of its own input, which the layers below keep changing. For linear layers this feedback can be solved exactly.

Theorem B.9 (Linked geometries)**.** *Let L linear layers evolve by compression only, $\dot{W}_\ell = -\eta W_\ell C_{\ell-1}$, $C_\ell = W_\ell C_{\ell-1} W_\ell^\top$, $C_0 = C$, and let $A_\ell = W_\ell \cdots W_1$.*

(a) Second-order blindness. *For any initialization the trajectory depends on the data only through C_0, and $C_\ell = A_\ell C_0 A_\ell^\top$; in particular $\operatorname{rank} C_\ell \leq \operatorname{rank} C_0$.*

(b) Band selection. *If $W_\ell(0) = O_\ell$ is orthogonal for $\ell < L$ and $W_L(0) = M$, then, with $Q_\ell = O_\ell \cdots O_1$, $A_\ell = Q_\ell\, d_\ell(C_0, t)$, $A_L = MQ_{L-1}d_L(C_0,t)$ and $C_\ell = Q_\ell\, \Lambda_\ell(C_0,t)\, Q_\ell^\top$, where*

$$d_\ell(\lambda, t) = \exp\Big[-\tfrac{1}{2}\textstyle\sum_{j<\ell} r_j(2\eta\lambda t)\Big], \qquad r_0(z) = z, \quad r_{j+1} = 1 - e^{-r_j}, \qquad \Lambda_\ell = \lambda\, d_\ell^2.$$

$\Lambda_\ell(\cdot, t)$ is unimodal with peak $\lambda_\ell^* = z_\ell/(2\eta t)$, where $z_1 = 1 > z_2 \approx 0.659 > z_3 \approx 0.494 > \cdots$ and $\ell z_\ell \to 2$. Each layer thus selects a band of the input spectrum, deeper layers select bands of smaller variance ($\lambda_\ell^* \approx 1/(\eta t \ell)$), and directions of equal input variance receive identical gains at every layer and time.

(c) Ridge retrieval without storage. *At* $t_L = \tau/(\eta L)$, $\|d_L(C_0, t_L) - (\mathbb{I} + \tau C_0)^{-1}\|_2 \le (5U^3/24 + U^2/8)/L$ *with* $U = 2\tau\|C_0\|_2$. *With identity hidden layers, a single Hebbian write* $M = C_{yx}/\rho$ *and* $\tau = 1/\rho$, $A_L \to C_{yx}(C + \rho\,\mathbb{I})^{-1}$.

Proof sketch. (a) Given C_0 the system is closed, and $C_\ell = W_\ell C_{\ell-1} W_\ell^\top$ with $C_{\ell-1} = A_{\ell-1} C_0 A_{\ell-1}^\top$ by induction. (b) The ansatz $W_\ell = Q_\ell\, w_\ell(C_0, t)\, Q_{\ell-1}^\top$ reduces the flow to one scalar equation per mode, $\dot{w}_\ell = -\eta w_\ell \Lambda_{\ell-1}$. With $I_j = \int_0^t \Lambda_j$, one has $w_\ell^2 = e^{-2\eta I_{\ell-1}}$, $\Lambda_\ell = w_\ell^2 \Lambda_{\ell-1}$ and $2\eta I_\ell = 1 - e^{-2\eta I_{\ell-1}}$ (both sides vanish at $t = 0$ and have the same derivative), so $2\eta I_j = r_j(2\eta\lambda t)$ and $\Lambda_\ell = \lambda\, r_\ell'(2\eta\lambda t)$. The logarithmic derivative of $z r_\ell'(z)$ is $(1 - S_\ell)/z$ with $S_\ell = z\sum_{j<\ell} r_j'$, and $S_\ell' = \sum_{j<\ell} r_j'(1 - S_j) > 0$ wherever $S_\ell = 1$, because $S_j < S_\ell$ for $j < \ell$; the crossing z_ℓ is therefore unique, and $S_{\ell+1}(z_\ell) > 1$ gives $z_{\ell+1} < z_\ell$. (c) With $u = 2\tau\lambda$ and $v_j = L r_j(u/L)$, the recursion $v_{j+1} = L(1 - e^{-v_j/L})$ is within $5U^3/(12L^2)$ per step of the time-$1/L$ map of $\dot{v} = -v^2/2$, whose solution $u/(1 + uq/2)$ integrates to $2\log(1 + u/2)$; both maps are 1-Lipschitz, which gives the bound, since $e^{-\log(1+u/2)} = (1 + \tau\lambda)^{-1}$. The same scaling gives $S_\ell(u/\ell) \to u/(1 + u/2)$, hence $\ell z_\ell \to 2$. □

Depth therefore reparametrizes the filter, from $e^{-\eta t C}$ at $L = 1$ to the reciprocal $(\mathbb{I} + \tau C)^{-1}$, and stratifies it into bands; but it never leaves the commutant of C_0, so spectral separation is the only separation a linear hierarchy can perform, and no layer can orthonormalize more than $\operatorname{rank} C_0$ memories. A nonlinear layer changes this, because the next layer compresses against a covariance that is not a function of C_0.

Theorem B.10 (Lifted geometries)**.** *Let a hidden layer compute* $\psi(z) = f(Gz)$, *with rows* g_r *of* G, *and let* $C_\psi = \mathbb{E}[\psi\psi^\top]$.

(a) Tensor-power lift. *If* $z \sim \mathcal{N}(0, \tilde{\Lambda})$ *and* $g_r^\top \tilde{\Lambda} g_r = 1$, *then* $C_\psi = \sum_{k\ge 0} c_k^2\, \mathcal{G}_k \tilde{\Lambda}^{\otimes k} \mathcal{G}_k^\top$, *where* c_k *are the normalized Hermite coefficients of* f *and* $\mathcal{G}_k$ *has rows* $(g_r^{\otimes k})^\top$. *The term* $k = 1$ *is the linked geometry of Thm. B.9(a); the term* k *is a congruent image of* $\tilde{\Lambda}^{\otimes k}$, *whose spectrum consists of the products* $\prod_i \tilde{\lambda}_i^{k_i}$ *of transmitted variances. For ReLU,* $c_2^2 = 1/(4\pi)$ *is* 16% *of the hidden second moment and* 88% *of its nonlinear part.*

(b) What linear compression cannot see. *Let* $x = s\,u + x_\perp$ *with* u *a unit vector,* $x_\perp \sim \mathcal{N}(0, \mathbb{I} - uu^\top)$, *and* s *symmetric and independent of* $x_\perp$, *with* $\mathbb{E}s^2 = 1$ *and* $\mathbb{E}s^4 = 3 + \kappa_4$. *Then* $C_0 = \mathbb{I}$, *so at the population level every deep linear compression network evolves as for* $x \sim \mathcal{N}(0, \mathbb{I})$, *independently of* u. *The quadratic lift* $xx^\top$, *realized linearly by a layer with* $f(u) = u^2$, *has covariance* $2\,\mathrm{Id} + \kappa_4\,|uu^\top\rangle\langle uu^\top|$ *on symmetric matrices: the non-Gaussian direction is an isolated eigenvalue* $2 + \kappa_4$. *For* $y = s^2 - 1$, $C_{yx} = 0$, *so no linear layer stores anything, whereas* $\langle uu^\top, xx^\top - \mathbb{I}\rangle = y$ *exactly; every linear predictor has risk at least* $\mathrm{Var}(y) = 2 + \kappa_4$.

Proof. (a) For each pair (r, r'), $(g_r^\top z, g_{r'}^\top z)$ is a standard Gaussian pair with correlation $Q_{rr'} = g_r^\top \tilde{\Lambda} g_{r'}$, so Mehler's formula gives $\mathbb{E}[f(g_r^\top z) f(g_{r'}^\top z)] = \sum_k c_k^2 Q_{rr'}^k$ (Daniely et al., 2016), and $Q_{rr'}^k = \langle g_r^{\otimes k}, \tilde{\Lambda}^{\otimes k} g_{r'}^{\otimes k}\rangle$. For ReLU, Gaussian integration gives $(c_0, \dots, c_3) = (1/\sqrt{2\pi}, 1/2, 1/(2\sqrt{\pi}), 0)$, $\mathbb{E}\max(g, 0)^2 = 1/2$ and $\sum_{k\ge 2} c_k^2 = 1/4 - 1/(2\pi)$. (b) $C_0 = \mathbb{E}[s^2]uu^\top + \mathbb{I} - uu^\top = \mathbb{I}$. In $\mathbb{E}[x_i x_j x_k x_l]$ the only departure from the Gaussian value is $\kappa_4 u_i u_j u_k u_l$, because terms with an odd power of s contain an odd number of factors of $x_\perp$; the Gaussian part acts as $M \mapsto 2M$ on symmetric matrices. Finally $C_{yx} = \mathbb{E}[(s^2 - 1)s]\, u = 0$ by symmetry, $u^\top x = s$, and $\mathbb{E}(y - w^\top x)^2 = \mathrm{Var}(y) + \|w\|^2$. □

Consecutive nonlinear layers therefore compress genuinely different geometries. A band of the second layer, $\{(i, j) : \tilde{\lambda}_i \tilde{\lambda}_j \in I\}$, combines input modes of different scales, which no band of

one spectrum can express, and the quadratic lift squares the separation that the first layer achieved between feature–feature and noise–noise pairs.

Proposition B.11 (Forgetting at a nonlinear layer follows the feature kernel)**.** *One HOP step of a readout B on features ψ, $B^+ = B - \frac{\alpha}{m}(B\psi(x)-y)\psi(x)^\top$, changes the response to an old input z by $-\alpha\,(B\psi(x)-y)\,K_m(x,z)$, with $K_m(x,z) = \psi(x)^\top\psi(z)/m$. For ReLU features with i.i.d. $\mathcal{N}(0,1)$ weights, $K_m(x,z) \to \frac{\|x\|\|z\|}{2\pi}[\sin\theta + (\pi-\theta)\cos\theta]$ almost surely (Cho & Saul, 2009): orthogonal inputs interfere with strength $\|x\|\|z\|/(2\pi)$, whereas antipodal inputs $z = -cx$ ($c > 0$) are exactly immune at every width, because their gates are disjoint.*

Proof. Evaluate $B^+\psi(z)$; apply the strong law to the i.i.d. rows of G; and note that $\max(g^\top x, 0)\max(-cg^\top x, 0) = 0$ for every g. □

Remark B.12 (Capacity and the Gaussian-equivalent regime). The sample geometry orthonormalizes P patterns only if they are linearly independent at that layer: at most $\operatorname{rank} C_0$ of them in a linear network, but up to $\min(P, n(n+1)/2)$ after a quadratic lift, and any $P \le m$ pairwise non-parallel inputs for wide ReLU features, whose limiting Gram matrix $\sum_k c_k^2 G^{\circ k}$ is positive definite by the Schur product theorem. The interpolation threshold of Thm. B.6(d) moves accordingly, as in random-feature double descent (Mei & Montanari, 2022). In the proportional regime with random first-layer weights, the terms $k \ge 2$ of Thm. B.10(a) act as a multiple of the identity (El Karoui, 2010; Hu & Lu, 2023), so the nonlinearity enters only through c_1^2 and an isotropic floor. The lifted structure becomes visible when the transmitted representation is low-dimensional relative to width, which is what upstream compression produces in its feature phase; we state this as a mechanism, not a theorem.

C Extensions to SCT and Deep learning

C.1 Polynomial Form

An instructive realization of the storage and compression rules is to consider the following parameterization:

$$H(x,y) = yx^T\rho_s(x) \qquad \pi(W \mid x) = W(\mathbb{I} - \rho_c(x)).$$

where ρ_c and ρ_s are polynomial functions of the data covariance:

$$\rho_s(x) = \sum_{k=0}^{K_s} s_k\beta^{-k}C_x^k, \qquad \rho_c(x) = \sum_{k=0}^{K_c} g_k\beta^{-k}C_x^k.$$

This yields the update rule

$$\Theta(x,y;W) = W - \big(W\rho_c(x) - yx^T\rho_s(x)\big).$$

As a concrete example, suppose $g_0 = 0$, $g_1 = 1$, and $s_k = \beta^{-1}g_{k+1}$. Defining the prediction error $e = Wx - y$, the update simplifies to an error-driven form:

$$\Theta(x,y;W) = W - \beta^{-1}ex^T\rho_c(x) = W - \beta^{-1}\partial_x W\rho_c(x).$$

If we further restrict the compression polynomial to $\rho_s(x) = \mathbb{I}$, several familiar update rules are recovered:

- HOP operator / gradient descent: $g_0 = 0$, $g_1 = 1$, $g_{k>1} = 0$
- HOP with weight decay $\gamma|W|^2$: $g_0 = \gamma$, $g_1 = 1$, $g_{k>1} = 0$
- Muon-like update (Jordan et al., 2024): $g_0 = 0$, $g_1 = 2$, $g_2 = -1.5$, $g_3 = 0.5$, $g_{k>3} = 0$
- Newton-Raphson: obtained via a Neumann series for C_x^{-1}

As discussed in B.2, this polynomials can be seen as a conditioning on the spectrum of C_x. For example, in the case of Muon, the polynomial pushes all singular values close to 1.

C.2 Optimization Tweaks

Momentum and Regualrizers

To extend Θ to momentum-based we can simply consider

$$m_{t+1} = \epsilon m_t + (1-\epsilon)(\Theta(x,t;W_t) - W_t)$$
$$W_{t+1} = W_t + m_{t+1}$$

When $\Theta(x,y;W_t) = \bar{\Theta}(x,y;W_t)$ and $y = \hat{y} - \eta\partial_y\mathcal{L}$ this is formally equivalent to classic momentum equations as

$$\bar{\Theta}(x,\hat{y} - \eta\partial_y\mathcal{L};W_t) - W_t = -\eta\partial_y\mathcal{L}x^T = -\eta\partial_W\mathcal{L}$$

With the same argument, we can extend this to most known optimizers. As for the weight-based regularization loss $R(W)$, we can directly add the regularizing gradients $\partial_W R(W)$ to the operator, for instance:

$$\Theta(x,y;W) = H(x,y) + \pi(W|x) - \eta_{reg}\partial_W R(W)$$

Biases

For what regards biases, we can seemingly extend this work by considering

$$\Theta(x,y;W,b) = \Theta(x,y-b;W)$$

with $b_i^{t+1} = b_i^t - \beta^{-1}\mathbb{E}_{k\sim batch}[y_i^k - W_i^t x_i^k]$. Its role is to essentially capture the average target activity, so that the signal of interest for computation becomes the deviation from the baseline neural activity (and so it can better use non-linearities).

Generalized HOP Parameterization

The storage–compression framework admits a broader class of update operators obtained by introducing linear preconditioning maps acting on both the representation and gradient geometry. Let

$$Q \in \mathbb{R}^{n\times n}, \qquad R \in \mathbb{R}^{m\times m},$$

with Q positive definite unless otherwise stated. We define the generalized HOP operator

$$\Theta_{Q,R}(x,y;W) = \frac{yQx^T}{\beta} + \bar{\pi}_{Q,R}(W|x),$$

where the generalized compression term is

$$\bar{\pi}_{Q,R}(W|x) = RW\left(\mathbb{I} - \frac{xQx^T}{\beta}\right).$$

Substituting yields

$$\Theta_{Q,R}(x,y;W) = RW - \frac{1}{\beta}(RWx - y)Qx^T.$$

This expression reveals two distinct geometric actions:

1. R acts directly on the parameter space by preconditioning the current weights;
2. Q modifies the effective input metric, thereby conditioning the gradient direction.

In particular, when $R = \mathbb{I}$, the update becomes

$$\Theta_Q(x,y;W) = W - \frac{1}{\beta}(Wx - y)Qx^T,$$

which corresponds to gradient descent under an input-space metric induced by Q.

Input-space geometry. Suppose $Q \succ 0$. Then there exists a matrix L such that

$$Q = L^T L,$$

for instance via the Cholesky decomposition. The update can then be rewritten as

$$\Delta W = -\alpha(Wx - y)(L^T L)x^T = -\alpha(Wx - y)L^T(Lx)^T.$$

Defining the transformed coordinates

$$x' = Lx,$$

the learning dynamics become equivalent to optimization in the transformed feature space x'. Thus, Q induces a learned geometry over inputs analogous to whitening or feature-metric transformations.

Metric-weighted optimization. We may further introduce a positive operator

$$U \in \mathbb{R}^{m \times m},$$

acting on the output geometry. The fully generalized update becomes

$$\Theta_{Q,R,U}(x, y; W) = U\frac{yQx^T}{\beta} + RW - URW\frac{xQx^T}{\beta},$$

or equivalently

$$\Theta_{Q,R,U}(x, y; W) = RW - \frac{U}{\beta}(RWx - y)Qx^T.$$

Defining

$$\tilde{W} = RW,$$

this update corresponds to gradient descent on the quadratic form

$$\mathcal{L}_U = (\tilde{W}x - y)^T U(\tilde{W}x - y).$$

Hence, U specifies the output-space metric used to measure prediction error. In particular, choosing

$$U = F^{-1},$$

where F is the Fisher information matrix, recovers a natural-gradient update.

Since the transformation $\tilde{W} = RW$ can be interpreted as a reparameterization of the weight matrix, we absorb R into W in the remainder of the derivation and focus on the pair (Q, U).

C.3 Nonlinear Updates and Associative Memories

The decomposition introduced in Eq. 2 naturally extends beyond linear maps. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable nonlinear model and let $J_x f$ denote its Jacobian at input x. Around x, the model admits the first-order approximation

$$f(x + \delta) \approx f(x) + J_x f\, \delta.$$

We can therefore interpret $J_x f$ as the local linearization of the nonlinear dynamics and apply the same storage–compression principle directly in tangent space.

To do so, we seek a first-order compression operator $\pi^{(1)}(f|x)$ satisfying

$$\pi^{(1)}(f|x)x = J_x f\, x - f(x),$$

such that the response of the local linear model along x is canceled relative to the nonlinear prediction. A minimal realization is

$$\pi^{(1)}(f|x) = J_x f - \beta^{-1} f(x)x^T.$$

The resulting nonlinear update operator becomes

$$\Theta(x, y; f) = \beta^{-1} yx^T + \pi^{(1)}(f|x) = J_x f - \beta^{-1}(f(x) - y)x^T.$$

Hence, learning in the nonlinear model can be interpreted as performing a storage–compression update on the local Jacobian representation.

In particular, defining the local quadratic loss

$$\mathcal{L}(x, y; f) = \frac{1}{2}\|f(x) - y\|^2,$$

we obtain

$$\Theta(x, y; f) = J_x f - \beta^{-1} \frac{\partial \mathcal{L}}{\partial (J_x f)},$$

showing that the nonlinear operator is equivalent to a gradient step on the linearized dynamics around x. Thus, the SCT decomposition extends naturally from weight-space optimization to optimization in the tangent bundle of nonlinear function classes.

Associative memories and Hopfield dynamics. One important application of this formulation is the training of associative memories, such as Hopfield networks. In particular, the nonlinear compression formalism recovers the pseudo-likelihood learning dynamics studied in modern Hopfield models (D'Amico et al., 2025).

Consider binary variables $x_i \in \{-1, +1\}$ with pairwise interaction energy

$$E(x) = -\sum_{i \neq j} J_{ij} x_i x_j.$$

Instead of maximizing the full likelihood, we approximate the joint distribution by the product of conditionals,

$$P_{\text{joint}}(x) \approx \prod_i P(x_i | x_{\setminus i}),$$

leading to the negative log pseudo-likelihood objective

$$\mathcal{L} = -\sum_{\mu=1}^{P} \sum_{i=1}^{N} \log p_i(\xi_i^\mu | \xi_{\setminus i}^\mu).$$

For the two-body energy above, this becomes

$$\mathcal{L} = -\sum_{\mu=1}^{P} \sum_i \left[\xi_i^\mu \sum_{j(\neq i)} J_{ij} \xi_j^\mu - \frac{1}{\lambda} \log 2\cosh\left(\lambda \sum_{j(\neq i)} J_{ij} \xi_j^\mu \right) \right].$$

Extremizing the loss yields the Callen identities,

$$0 = \frac{\partial \mathcal{L}}{\partial J_{kl}} = \sum_{\mu=1}^{P} \left[\xi_k^\mu - \tanh\left(\lambda \sum_{j(\neq k)} J_{kj} \xi_j^\mu \right) \right] \xi_l^\mu.$$

In the limit $\lambda \to \infty$, each term vanishes independently when

$$\xi_k^\mu = \operatorname{sgn}\left(\sum_{j(\neq k)} J_{kj} \xi_j^\mu \right), \qquad \forall\, \mu, k,$$

which is precisely the fixed-point stability condition for an associative memory storing the patterns $\{\xi^\mu\}$.

For clarity, consider the single-pattern case $P = 1$. The gradient reduces to

$$\frac{\partial \mathcal{L}}{\partial J_{kl}} = \left[\xi_k - \tanh\left(\lambda \sum_{j(\neq k)} J_{kj} \xi_j \right) \right] \xi_l.$$

Defining the nonlinear retrieval dynamics

$$\psi_k(\xi, J) = \tanh\left(\lambda \sum_{j \neq k} J_{kj} \xi_j \right),$$

the update takes the form

$$\Theta(\xi, \xi; J) = \frac{\xi\xi^T}{\beta} + J_\xi \psi - \psi(\xi, J)\frac{\xi^T}{\beta},$$

which is exactly the nonlinear storage–compression decomposition applied to the Hopfield flow.

This perspective reveals a continuous interpolation between Hebbian learning and perceptron-like maximal stability learning, as discussed in (D'Amico et al., 2025). For small λ,

$$\tanh(\lambda z) \approx \lambda z,$$

and the update reduces to a Hebbian outer-product rule. In contrast, as $\lambda \to \infty$, the dynamics converge toward the maximally stable perceptron solution, emphasizing the correction of low-margin patterns. Increasing λ therefore progressively shifts the learning objective from average correlation storage toward robust attractor stabilization.

From the SCT viewpoint, this transition corresponds to an increasing role of the compression term. Early in learning, storage dominates and the dynamics are approximately Hebbian. As training proceeds, the nonlinear compression operator reorganizes existing representations by subtracting the discrepancy between the nonlinear retrieval dynamics $\psi(\xi, J)$ and its local linearization. This mechanism increases storage capacity while preserving attractor consistency, enabling the network to generalize to corrupted or previously unseen patterns through internal reorganization of the memory landscape.

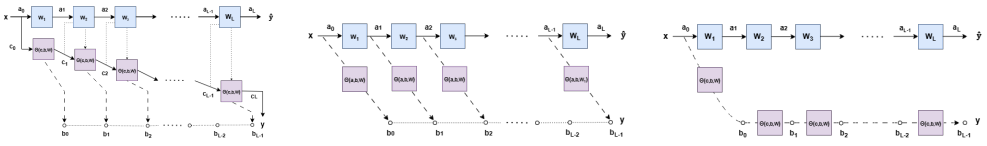
C.4 Deep Learning

Let's consider a Neural Network with L layers $\{W_i\}_{i=1,\dots,L}$ and activation functions ϕ_i. For sake of notation simplicity, we will ignore the biases, but everything that follows can be directly extended to include them as well (see appendix C.2). For the forward inference $\hat{y} = \text{NNet}(x) = \phi_L \circ W_{L-1}\phi_{L-1} \circ ...\phi_1 \circ W_0 x$ we will use the following notation

Forward Inference: $a_{i+1}^t = W_i \phi_i(a_i^t)$, where $\phi_0 = id(\cdot)$, $a_0^t = x^t$, $a_L^t = \hat{y}^t$.

Here, at a given time t, a_i^t and $\phi_i(a_i^t)$ are the pre-activation and post-activation states. In order to learn a task, the network needs to learn a target, either obtained in a supervised fashion or as a minimization of an unsupervised loss with respect to the current activity. To be as general as possible, we define a set of targets b_i^t, with y as the desired output.

Targets: $\{b_i^t\}_{i=0,\dots,L-1}$, where $b_{L-1}^t = y^t$



(a) General interpolation (b) $\lambda = 0$ (c) $\lambda = 1$

Figure 6: Interpolation Schemes for various λ.

We can use the notation of the previous section to define the i-th layer update at time t as

$$W_i^{t+1} = (1-\alpha)W_i^t + \alpha\Theta(a_i^t, b_i^t; W_i^t)$$

For a Hebb-like storage rule and $a_i^t = a^i$, $b_i^t = b^i$, this converges to some $W_*^i \in \mathcal{M}_{a_i}^{b_i}$ for $0 < \alpha\beta^{-1} < \min_t \frac{2}{||a_i^t||^2}$. Moreover, this can be easily extended to batched training ($a_i^t \in A_i^t, b_i^t \in B_i^t$) as follows:

$$W_i^{t+1} = (1-\alpha)W_i^t + \alpha\mathbb{E}_{A_i^t, B_i^t}\Big[\Theta(a_i^t, b_i^t; W_i^t)\Big]$$

Interpolative dynamics In a multi-layer network each layer's local storage-compression rule is a local learning rule that nudge it over encoding input-target maps while different layers interact by shaping the inputs (through forward pass) and, possibly, the targets. We can generalize the concept of layer-wise optimization inputs by introducing an *interpolative dynamic* within the network. At training time t, instead of simply chaining forward inference activations (a_i^t) to targets (b_i^t), we can define an intermediate signal c_i^t that blends the forward pass with the desired targets (Figure 6a). Let $c_0^t = x^t$ and $c_L^t = y^t$. For an intermediate layer i, we define the interpolative dynamic c_t^{i+1} as a weighted average:

$$c_t^{i+1} = (1-\lambda)W_i\phi_i(c_i^t) + \lambda b_i^t$$

where $0 \leq \lambda \leq 1$. The update rule for the weights W_i then uses this interpolative dynamic as its input to the Θ operator:

$$W_i^{t+1} = (1-\alpha)W_i^t + \alpha\Theta(\phi_i(c_i^t), b_i^t; W_i^t)$$

This introduces a spectrum of learning behaviors based on λ (Fig 6a), where with $\lambda = 0$ (Fig 6b) each layer learns to map $a_i \mapsto b_i$ whereas with $\lambda = 1$ the layer is instead nudged to encode $b_{i-1}^t \mapsto b_i^t$ (Fig 6c).

Further additions like momentum, Adam, and weight-based regularizers (L2, EWC, ...) are reported in the Appendix C.

C.5 Equivalence with known algorithms

Under specific storage, compression, and target rules, the SCT framework seamlessly recovers established deep learning algorithms. Specifically, backpropagation, predictive coding, and target propagation all rely on the *HOP* operator (orthogonal projection $W_i P_{x_i}$ for compression; normalized Hebb rule $\beta^{-1} y_i x_i^T$ for storage).

What structurally differentiates these algorithms is the interpolation factor λ and the definition of the targets b_i, which dictate cross-layer credit assignment. Table 1 summarizes the target rules necessary to recover Backpropagation (BP) (Linnainmaa, 1976; Rumelhart et al., 1986), Predictive Coding (PC) (Rao & Ballard, 1999; Millidge et al., 2022; Whittington & Bogacz, 2017), and Target Learning (TL) (Bengio, 2014; Meulemans et al., 2020).

Method	**Target Rule** b_i	λ
BP	$a_{i+1} - \eta \left(\prod_{k=i}^{L-1} J_{\phi_k}^T W_k^T \right) J_{\phi_L}^T (\hat{y} - y)$	0
PC	$\left\{ \underset{a_1,\ldots,a_{L-1}}{\arg\min}\ \mathcal{F}_t(a_1,\ldots,a_{L-1}\|x,y) \right\}_i$	1
TL	$g_i(b_{i+1})$	0/1

Table 3: Equivalence of the SCT framework with popular training rules.

This equivalence can naturally be extended to other algorithms as Feedback Alignment and Equilibrium Propagation. The case of Contrastive Hebbian Learning (Detorakis et al., 2019) is particularly interesting, as it uses a variant of the HOP operator where storage is performed on the *clamped* dynamics while compression is performed on the *free* ones.

We will now report the steps required to obtain the equivalence with known training algorithms.

▶ Equivalence with Backpropagation

In general, for multiple layers (W_i, ϕ_i), with $\tilde{z}_i = \phi_i(z_i)$ and being the final inference $\tilde{z}_L = \hat{y}$. Given $\beta \Delta W_i = \xi_i \tilde{z}_i^T - \phi_i(W_i z_i)\tilde{z}_i^T$ the backpropagation-equivalent target is:

$$\xi_i = z_{i+1} - \eta \frac{\partial \mathcal{L}}{\partial z_{i+1}} = W_i \tilde{z}_i - \eta \frac{\partial \mathcal{L}}{\partial W_i z_i}$$

By exploiting the chain rule (for simplicity, assume MSE loss $\mathcal{L} = \frac{1}{2}e^t e$, $e = \hat{y} - y^*$) we obtain the same update rule as backpropagation:

$$\frac{\partial \mathcal{L}}{\partial z_{i+1}} = (\prod_{k=i}^{L} J_{\phi_k}^T W_k^T)\frac{\partial \mathcal{L}}{\partial \hat{y}} = (\prod_{k=i}^{L} J_{\phi_k}^T W_k^T)e$$

$$\xi_i = z_{i+1} - \eta(\prod_{k=i}^{L} J_{\phi_k}^T W_k^T)e$$

Then

$$\Theta(\xi_i, \tilde{z}_i, W_i) = \beta^{-1}\xi_i \tilde{z}_i{}^T + W_i(\mathbb{I} - \beta^{-1}\tilde{z}_i\tilde{z}_i{}^T)$$

$$= \beta^{-1}(z_{i+1} - \eta(\prod_{k=i}^{L} J_{\phi_k}^T W_k^T)e)\tilde{z}_i{}^T + W_i(\mathbb{I} - \beta^{-1}\tilde{z}_i\tilde{z}_i{}^T)$$

$$\beta^{-1}z_{i+1} - W_i\tilde{z}_i\tilde{z}_i{}^T + W_i - \eta(\prod_{k=i}^{L} J_{\phi_k}^T W_k^T)e\tilde{z}_i{}^T = W_i - \beta^{-1}\eta\frac{\partial \mathcal{L}}{\partial W_i}$$

which recovers backpropagation for the i-th layer.

▶ EQUIVALENCE WITH TARGET LEARNING

If we define the targets as in (Meulemans et al., 2020; Whittington & Bogacz, 2017), we get exactly TP:

$$\xi_i = g_i(\xi_{i+1}) = t_i(Q_i\xi_{i+1})$$

We will see that if we take $\xi_i = \hat{z}_i - \eta\frac{\partial \mathcal{L}}{\partial \hat{z}_i}$, for some downstream loss $\mathcal{L}$, IFI is formally equivalent to backpropagation. Indeed, for

$$g_i(\xi_{i+1}|\hat{z}_i) = \hat{z}_i - \eta b_i(\xi_{i+1})$$
$$b_i(\xi_{i+1}) = J_{\phi_i}^T W_i^T \xi_{i+1}^T$$
$$\xi_L = \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

the two algorithms equate.

▶ EQUIVALENCE WITH PREDICTIVE CODING

Following the definition of PC of (Millidge et al., 2022), we recover PC by taking the following targets:

$$\xi_i = \left\{ \underset{x_2,...,x_{L-1}}{\arg\min}\ \mathcal{F}_t(x_2, ..., x_{L-1}|\bar{x}_1, \bar{x}_L) \right\}_i$$

$$\mathcal{F}_t(x_1, ..., x_L) = \frac{1}{2}\sum_l ||\varepsilon_{l,t}||^2 = \frac{1}{2}\sum_l ||x_l - f_{\theta_t}(x_{l-1})||^2$$

Here, $\bar{x}_1, \bar{x}_L$ are the input and desired output (clamped) and $\mathcal{F}_t$ the energy function to be minimized to find the intermediate targets.

D FORGETTING-FREE UPDATES: THEORY AND PROOFS

This appendix answers two questions raised by the forgetting identity equation 4: which layer-wise updates leave stored associations intact (App. D.1), and what such an update must know about the past (App. D.2). Table 4 summarizes the answers. HOPc is the member of the forgetting-free family that keeps the gradient step on the current sample (App. D.3). Proofs are collected in App. D.4.

Notation. A layer $W \in \mathbb{R}^{m\times n}$ receives an input x (a hidden representation in a deep network) with target y and error $e = Wx - y$. The HOP step is $\bar{\Theta}(x, y; W) = W - G$ with the interference operator $G = \beta^{-1} e\, x^\top$, and a correction R gives $\Delta W = -G + R$. Memories (x^μ, y^μ) satisfy $Wx^\mu = y^\mu$, $\mathcal{M} = \mathrm{span}\{x^\mu\}$, $\Pi_\mathcal{M}$ is the orthogonal projector onto $\mathcal{M}$ and $\Pi_\mathcal{M}^\perp = \mathbb{I} - \Pi_\mathcal{M}$. By equation 4, the update is forgetting-free iff $(R - G)x^\mu = 0$ for every memory. Accumulated over any sequence of updates $W_{k+1} = H_k + W_k\Omega_k$, $\Omega_k = \mathbb{I} - \eta_k C_k$, started at $W_0 = W_\mathrm{A}$, the identity becomes exact by Prop. B.1(a) and the telescoping sum $\Phi_{0,T-1} - \mathbb{I} = -\sum_k \eta_k C_k \Phi_{k+1,T-1}$:

$$W_T - W_\mathrm{A} = -\sum_{k<T} G_k(W_\mathrm{A})\, \Phi_{k+1,T-1}, \qquad G_k(W) = \eta_k W C_k - H_k. \tag{11}$$

For HOP storage, $G_k(W_\mathrm{A}) = \eta_k \nabla L_k(W_\mathrm{A})$: the forgetting of an old input is the sum of the new-task gradients evaluated at the *old* map, each transported to the present by the survival operator $\Phi_{k+1,T-1}$ of all later compressions. For a fixed batch it reduces to Thm. B.8.

D.1 THE SPACE OF FORGETTING-FREE UPDATES

Lemma D.1 (Exact-retention family)**.** *The update is forgetting-free on $\mathcal{M}$ iff $R\,\Pi_\mathcal{M} = G\,\Pi_\mathcal{M}$, that is,*

$$R = \beta^{-1} e\,(\Pi_\mathcal{M} x)^\top + Z\,\Pi_\mathcal{M}^\perp, \qquad Z \in \mathbb{R}^{m\times n} \text{ arbitrary}. \tag{12}$$

The two terms are Frobenius-orthogonal, and the minimum-norm member ($Z\Pi_\mathcal{M}^\perp = 0$) is gradient projection, $\Delta W_\mathrm{GP} = -\beta^{-1} e\,(\Pi_\mathcal{M}^\perp x)^\top$ (Zeng et al., 2019; Farajtabar et al., 2020; Saha et al., 2021).

Proposition D.2 (HOPc is the minimal forgetting-free drift)**.** *Let $p = \Pi_\mathcal{M} x$, $q = \Pi_\mathcal{M}^\perp x \neq 0$ and $\sin\theta = \|q\|/\|x\|$.*

(i) Exactly one minimum-norm member of equation 12 is an admissible drift ($Rx = 0$, Thm. 2.2):

$$R_\mathrm{H} = \beta^{-1} e\Big(p - \tfrac{\|p\|^2}{\|q\|^2}\, q\Big)^\top, \tag{13}$$

so the current sample moves exactly as under gradient descent, $W^+ x = \bar{\Theta}(x, y; W)\, x$.

(ii) $\Delta W_\mathrm{H} = \csc^2\theta\, \Delta W_\mathrm{GP}$: projection shrinks the step on the current output by $\sin^2\theta$, and HOPc restores it at weight cost $\|\Delta W_\mathrm{H}\|_F = \csc\theta\, \|G\|_F$.

(iii) For a batch X with errors E, $K = X^\top X$, $\tilde{X} = \Pi_\mathcal{M}^\perp X$ of full column rank and $\tilde{K} = \tilde{X}^\top \tilde{X}$,

$$\Delta W_\mathrm{GP} = -\beta^{-1} E\, \tilde{X}^\top, \qquad \Delta W_\mathrm{H} = -\beta^{-1} E\, K \tilde{K}^{-1} \tilde{X}^\top, \tag{14}$$

and ΔW_H is the minimum-norm forgetting-free update with $\Delta W_\mathrm{H} X = -\beta^{-1} EK$, the unconstrained function-space step of the batch.

Gradient projection and HOPc live in the same admissible set: the first is faithful to the gradient in weight space, the second to the step in function space. The stability–plasticity trade-off is relocated into conditioning: as x approaches $\mathcal{M}$ ($\theta \to 0$) projection stops learning, whereas the HOPc step grows as $\csc^2\theta$, which the threshold α_max of Sec. 3.1 caps. Every forgetting-free update also reroutes each aligned memory $x^\mu = \frac{\alpha_\mu}{\|x\|^2} x + z_\perp^\mu$, $\alpha_\mu = \langle x, x^\mu\rangle$: from $\Delta W x^\mu = 0$, whatever is learned along x is compensated along the private component, $\Delta W z_\perp^\mu = -\frac{\alpha_\mu}{\|x\|^2} \Delta W x$.

D.2 What exact retention requires

Let $\mathcal{F}_t$ collect what is available before the new pair (x_t, y_t) is drawn: the history, the current weights and any stored statistic of the past (Fisher matrices, anchors, subspace bases, buffers, generators).

Definition D.3 (Sample-blind and proactive corrections)**.** A correction R_t is *sample-blind* if it is $\mathcal{F}_t$-measurable, and *proactive* otherwise. Weight penalties (EWC, SI, MAS, L_2 anchoring) and replay or distillation gradients added to the update (ER, DER, LwF) are sample-blind; gradient projection, OWM and HOPc are proactive, since they act on the current G_t.

Theorem D.4 (Sample-blind corrections cannot be exact)**.** *Let the memory inputs x^μ be fixed (e.g. at the first layer), $\bar{G}_t = \mathbb{E}[G_t \mid \mathcal{F}_t]$, $\Sigma_\mathcal{M} = \sum_\mu x^\mu x^{\mu\top}$ and $\delta_t^\mu = W_t x^\mu - y^\mu$ with $\delta_0^\mu = 0$.*

(a) A sample-blind R_t is forgetting-free for almost every new sample iff $G_t \Pi_\mathcal{M} = \beta^{-1} e_t (\Pi_\mathcal{M} x_t)^\top$ is a.s. constant given $\mathcal{F}_t$, which requires all $\Pi_\mathcal{M} x_t$ with $e_t \neq 0$ to be parallel to one unit vector u and $e_t \langle x_t, u\rangle$ to be constant.

(b) The one-step forgetting of any sample-blind correction is at least

$$\mathcal{F}_t = \beta^{-2} \sum_\mu \operatorname{tr} \operatorname{Cov}\left[e_t \langle x_t, x^\mu\rangle \,\middle|\, \mathcal{F}_t\right], \tag{15}$$

with equality iff $(R_t - \bar{G}_t)\Sigma_\mathcal{M} = 0$; minibatches of size b divide it by b. Write $\mathcal{F}_t^\mu$ for the term of memory μ.

(c) The floor accumulates. At every T, $\mathbb{E}\|\delta_T^\mu\|^2 \geq \mathbb{E}\,\mathcal{F}_{T-1}^\mu$; a correction that does not react to the drift forgets diffusively, $\mathbb{E}\|\delta_T^\mu\|^2 = \sum_{t<T} \mathbb{E}\,\mathcal{F}_t^\mu$; and a restoring correction $-\epsilon\,\delta_t^\mu$, $0 < \epsilon < 2$, bounds forgetting by $\sup_t \mathbb{E}\,\mathcal{F}_t^\mu / (\epsilon(2-\epsilon))$ but needs the memories' current outputs $W_t x^\mu$.

Weight penalties and replay gradients are thus on the same side: both are blind to the alignment of the incoming sample, and they differ only in whether and how they restore. Part (c) is the discrete-time counterpart of the Ornstein–Uhlenbeck picture of Thm. 3.4. The obstruction is not how much is stored about the past, but how it enters the update.

Remark D.5 (EWC and OWM use the same statistic). For a linear layer with Gaussian likelihood the Fisher of the memories is $\Sigma_\mathcal{M} \otimes \mathbb{I}$. The explicit EWC step adds $-\lambda(W - W^\star)\Sigma_\mathcal{M} = -\lambda \sum_\mu (W x^\mu - W^\star x^\mu) x^{\mu\top}$, a distillation-replay gradient: sample-blind and restoring. The implicit (proximal) step of the same penalty is $\Delta W = -G(\mathbb{I} + \lambda \Sigma_\mathcal{M})^{-1}$, the OWM projector (Zeng et al., 2019) with $\alpha = \lambda^{-1}$, which is proactive and tends to gradient projection as $\lambda \to \infty$ (Kao et al., 2021). The divide between regularization and projection is the divide between explicit and implicit use of one quadratic form.

A static statistic thus suffices at a single layer. In a deep network it suffices only while the representations of the memories stay put. Let $f = f_{<i}(\cdot; \theta_t)$ map inputs to the input of layer i, let $A = \{x^\mu\}$ be the memory inputs, and $S_f(A) = \operatorname{span} f(A)$ the subspace that layer i must protect. A learner that stores only a statistic $\omega = \Omega(A)$ cannot tell A from any A' with $\Omega(A') = \omega$, so it must protect the *envelope* $\mathcal{E}_f(\omega) = \sum_{A' :\, \Omega(A') = \omega} S_f(A')$.

Theorem D.6 (Tracking the past requires the past)**.** *(a)* Missing information costs plasticity. *A correction computed from ω is forgetting-free at layer i iff it annihilates $\mathcal{E}_f(\omega)$. Its progress on the current input per unit update norm is then at most a fraction $\|\Pi_\mathcal{E}^\perp x_i\| / \|x_i\|$ of that of gradient descent, which decreases as the envelope grows.*

(b) Frozen or linear representations. *The envelope is tight, and stored statistics suffice (as in OWM, GPM and Adam-NSCL), if every layer has been forgetting-free on the spans stored when the memories were learned, or if f is linear and a basis of $\operatorname{span} A$ is propagated through the current network.*

Information about the past	Correction	Per-layer retention	Examples
none	—	diffusive, $\sum_t \mathcal{F}_t$	SGD
anchors, importances or replayed targets, entering additively	sample-blind, restoring	bounded, $\geq \mathcal{F}_t$ (Thm. D.4)	EWC, SI, ER, DER
stored span or second moment, entering multiplicatively	proactive, static geometry	exact while representations are frozen (Thm. D.6b)	OWM, GPM, Adam-NSCL, NCL
memory inputs (buffer or generator)	proactive, current geometry	exact under reorganization (Thm. D.6d)	LPR (soft), HOPc
memory inputs and outputs	proactive and restoring	exact, with repair of past drift	HOPc ($\epsilon > 0$)

Table 4: What a layer-wise correction must know about the past, and what it buys.

(c) Nonlinear reorganization. *If $f = \phi(W\,\cdot)$ with ϕ the ReLU, second moments do not suffice: for $\omega = \Sigma_A$, $\mathcal{E}_f(\omega) \supseteq S_f(A) + W\,\mathrm{span}(A)$, strictly whenever $W\,\mathrm{span}(A) \not\subseteq S_f(A)$, e.g. for one memory whenever Wx^μ has a negative entry.*

(d) Universal reorganization. *If the lower maps reachable by training are point-separating (as for ReLU networks with biases and at least two layers below i), ω gives a tight envelope for every memory set iff ω determines the memory inputs. Tight protection under reorganization requires the past inputs themselves, replayed or generated (Ha & Schmidhuber, 2018).*

Together, Theorems D.4 and D.6 are the precise content of Thm. 3.2. Exactness requires a proactive correction, coupled to the current sample through its alignment with the protected span; a stored statistic can supply that span only while past representations stay frozen or move linearly; once they reorganize nonlinearly, as they must for feature learning, every piece of missing information about past inputs is paid for in plasticity. This is the formal content of the observation that replay lets downstream layers track drift in upstream ones (van der Veldt et al., 2025), and it complements Knoblauch et al. (2020) and Peng et al. (2023) by making the plasticity price explicit layer by layer. The protective part never needs the memory targets y^μ; restoring past drift does.

Proposition D.7 (Stale and estimated geometry)**.** *(i) If layer i protects a stored subspace $\hat{\mathcal{M}}_i \ni x_i^\mu(0)$ while the memory's representation has drifted to $x_i^\mu(t) = x_i^\mu(0) + \delta_i^\mu(t)$, the change inflicted by the layer's own update is*

$$\Delta W_i\, x_i^\mu(t) = \Delta W_i\, \Pi^\perp_{\hat{\mathcal{M}}_i}\, \delta_i^\mu(t), \tag{16}$$

the out-of-subspace part of the upstream drift, i.e. of the forgetting integral of Prop. 3.3. It compounds with depth and vanishes if the subspace is recomputed from the current representations of stored or generated inputs.

(ii) If $\Delta W\, \Pi_{\hat{\mathcal{M}}} = 0$ for an estimate $\hat{\mathcal{M}}$ of $\mathcal{M}$, the leakage on the true span is $\|\Delta W \Pi_{\mathcal{M}}\|_F \leq \|\Delta W\|_F\, \|\Pi_{\mathcal{M}} - \Pi_{\hat{\mathcal{M}}}\|_2$, with $\|\Delta W_{\mathrm{H}}\|_F = \csc\hat{\theta}\, \|G\|_F$ for HOPc. A leakage ε on a top-r eigenspace with spectral gap g_r therefore needs $N \gtrsim r\, \mathbf{r}(\Sigma)\, \|\Sigma\|_2^2/(g_r\varepsilon)^2$ stored or generated samples (Yu et al., 2015; Koltchinskii & Lounici, 2017), where $\mathbf{r}(\Sigma) = \mathrm{tr}\,\Sigma/\|\Sigma\|_2$ is the effective rank of the protected representation.

Part (i) ties the necessity of memories to feature learning. When past representations move by a vanishing fraction of their norm, as in the lazy regime, stored statistics become exact asymptotically (Bennani et al., 2020; Doan et al., 2021); when they move at $O(1)$, as in the rich regime driven by the compression of Thm. 2.3, the leak is non-zero and past inputs become necessary. Part (ii) says that the buffer size is set by the effective rank of the protected representation, not by the complexity of the past distribution.

Algorithm 1 HOP Correction

Require: $\beta > 0, \alpha_{\text{max}} > 0, \epsilon \geq 0, \mathcal{X}, \mathcal{Y}, \mathcal{X}^\mu, \mathcal{Y}^\mu$
  while $t \leq T$ **do**
    for each layer **do**
      $X, Y \leftarrow \{x_i\}_b \sim \mathcal{X}, \{y_i\}_b \sim \mathcal{Y}$
      $M, \Psi \leftarrow \{x_\mu, y_\mu\}_P \sim p(\mu|\langle X, \mathcal{X}^\mu\rangle)$
      $v, \xi \leftarrow \text{GramSchmidt}(X, M)$
      $\alpha_{i\mu} \leftarrow \langle x_i, M_\mu\rangle$
      $g_\mu \leftarrow \langle \xi_\mu, M_\mu\rangle$
      $\Delta\mu \leftarrow \mathbb{E}_i\Big[\alpha_{i,\mu}(W_t x_i - y_i)\Big] - \epsilon(W_t x_\mu - y_\mu)$
      $H_\mu \leftarrow g_\mu^{-1}\Delta\mu \otimes \xi_\mu$
      $W_{t+1} \leftarrow (\sum_i^b \beta^{-1} y_i \otimes x_i + W_t P_{x_i}) + \sum_\mu^P H_\mu$
    end for
  end while

D.3 HOPc in practice

Consider 1 as the algorithm for HOPc; this method applies the member of equation 12 selected by Prop. D.2 to the actual update of each layer, and adds a restoring term. Let ΔW_0 be the base update (SGD, Adam, or the full DER++ step), X the current batch inputs to the layer, M the *current* representations of the protected memories and Y_M their targets. With $A = \Pi_X^\perp M$,

$$\Delta W = \Delta W_0 - \big(\Delta W_0 M + \epsilon\,(WM - Y_M)\big)\,A^\dagger. \tag{17}$$

Since $A^\dagger X = 0$ and $A^\dagger M = \mathbb{I}$, the batch moves exactly as under the base optimizer, $\Delta W X = \Delta W_0 X$, and the memories move by $\Delta W M = -\epsilon(WM - Y_M)$: they are held fixed for $\epsilon = 0$ and pulled back otherwise, which is the restoring correction of Thm. D.4(c) confined to the kernel of the current batch. The protective term needs only memory inputs, the restoring term also their targets. For one sample, one memory and $\epsilon = 0$, equation 17 reduces to

$$\Delta\pi(x^\mu) = \beta^{-1}\alpha\,(\hat{y} - y)\,\frac{z_\perp^\top}{\|z_\perp\|^2}, \qquad z_\perp = x^\mu - \frac{\alpha}{\|x\|^2}x, \quad \alpha = \langle x, x^\mu\rangle, \tag{18}$$

and with several memories the pseudo-inverse is needed unless the $z_\perp^\mu$ are orthogonal. The size of the correction is governed by $\sigma_{\min}(A)$. For unit x and one unit memory the amplification is $\csc^2\theta = 1/(1-\alpha^2)$, so excluding memories with $\alpha^2 > \alpha_{\max}^2$ caps it at $1/(1-\alpha_{\max}^2)$ (about 17 for $\alpha_{\max} = 0.97$) and deliberately allows forgetting of memories that the current input re-uses. With many memories the ridge form $A^\dagger \to (A^\top A + \gamma\mathbb{I})^{-1}A^\top$ still gives $\Delta W X = \Delta W_0 X$ and protects each singular direction of A with weight $s^2/(s^2+\gamma)$.

D.4 Proofs

Lemma D.1. As $\Pi_{\mathcal{M}} x^\mu = x^\mu$, $Gx^\mu = G\Pi_{\mathcal{M}} x^\mu$, so the update is forgetting-free iff $R - G$ vanishes on $\mathcal{M}$, i.e. $(R-G)\Pi_{\mathcal{M}} = 0$; write $R = R\Pi_{\mathcal{M}} + R\Pi_{\mathcal{M}}^\perp$. The terms are orthogonal since $\text{tr}(\Pi_{\mathcal{M}} G^\top Z \Pi_{\mathcal{M}}^\perp) = \text{tr}(G^\top Z \Pi_{\mathcal{M}}^\perp \Pi_{\mathcal{M}}) = 0$, so $\|R\|_F$ is minimal at $Z\Pi_{\mathcal{M}}^\perp = 0$, where $\Delta W = -G\Pi_{\mathcal{M}}^\perp$.

Proposition D.2. (i) Admissibility $Rx = 0$ and $R\Pi_{\mathcal{M}} = G\Pi_{\mathcal{M}}$ give $Rq = -Gp = -\beta^{-1}e\|p\|^2$. On $\mathbb{R}^n = \mathcal{M} \oplus \text{span}\{q\} \oplus \mathcal{N}$, R is fixed on the first two summands and free on $\mathcal{N}$, and the Frobenius norm is minimized uniquely by $R|_{\mathcal{N}} = 0$, which is equation 13. (ii) $\Delta W_{\text{H}} = -G + R_{\text{H}} = -\beta^{-1}\frac{\|x\|^2}{\|q\|^2}e\,q^\top = \csc^2\theta\,\Delta W_{\text{GP}}$, and $\|\Delta W_{\text{H}}\|_F = \beta^{-1}\|e\|\|x\|^2/\|q\| = \csc\theta\|G\|_F$. (iii) A forgetting-free update satisfies $\Delta W = \Delta W \Pi_{\mathcal{M}}^\perp$, so the requirement $\Delta W X = -\beta^{-1}EK$ reads $\Delta W \tilde{X} = -\beta^{-1}EK$, whose minimum-norm solution $-\beta^{-1}EK\tilde{X}^\dagger = -\beta^{-1}EK\tilde{K}^{-1}\tilde{X}^\top$ has rows in $\text{col}(\tilde{X}) \subseteq \mathcal{M}^\perp$. For projection, $\Delta W_{\text{GP}} = -G\Pi_{\mathcal{M}}^\perp = -\beta^{-1}E\tilde{X}^\top$.

Theorem D.4. (a) By Lemma D.1 the step is exact for a sample iff $R_t\Pi_{\mathcal{M}} = G_t\Pi_{\mathcal{M}}$. If this holds a.s., $G_t\Pi_{\mathcal{M}}$ equals the $\mathcal{F}_t$-measurable $R_t\Pi_{\mathcal{M}}$; conversely a constant $G_t\Pi_{\mathcal{M}} = C_t$ is matched by $R_t = C_t$. If $C_t \neq 0$, all non-zero $e_t(\Pi_{\mathcal{M}}x_t)^\top$ equal one rank-one matrix, so the $\Pi_{\mathcal{M}}x_t$ are parallel to one u and $e_t\langle x_t, u\rangle$ is constant. (b) Given $\mathcal{F}_t$, the cross term has zero mean, so $\mathbb{E}[\|(R_t - G_t)x^\mu\|^2 \mid \mathcal{F}_t] = \|(R_t - \bar{G}_t)x^\mu\|^2 + \mathbb{E}[\|(G_t - \bar{G}_t)x^\mu\|^2 \mid \mathcal{F}_t]$. Summed over μ, the first term is $\mathrm{tr}[(R_t - \bar{G}_t)\Sigma_{\mathcal{M}}(R_t - \bar{G}_t)^\top]$ and the second is $\mathcal{F}_t$. (c) For fixed x^μ, $\delta^\mu_{t+1} = \delta^\mu_t + m^\mu_t + \zeta^\mu_t$ with predictable $m^\mu_t = (R_t - \bar{G}_t)x^\mu$ and martingale differences $\zeta^\mu_t = -(G_t - \bar{G}_t)x^\mu$ that no sample-blind correction can alter, with conditional second moment $\mathcal{F}^\mu_t$. Hence $\mathbb{E}\|\delta^\mu_T\|^2 = \mathbb{E}\|\delta^\mu_{T-1} + m^\mu_{T-1}\|^2 + \mathbb{E}\,\mathcal{F}^\mu_{T-1}$, which gives the lower bound, the diffusive case ($m \equiv 0$) and, for $m^\mu_t = -\epsilon\delta^\mu_t$, the recursion $\mathbb{E}\|\delta^\mu_{t+1}\|^2 = (1-\epsilon)^2\mathbb{E}\|\delta^\mu_t\|^2 + \mathbb{E}\,\mathcal{F}^\mu_t$. For all memories at once, $m_t = -\epsilon\delta_t$ is realized by $R_t = \bar{G}_t\Pi_{\mathcal{M}} - \epsilon(W_tM - Y_M)M^\dagger$, which needs the memory inputs and their reference outputs.

Remark D.5. For $p(y \mid x, W) = \mathcal{N}(Wx, \sigma^2\mathbb{I})$ the Fisher quadratic form of a perturbation V over the memories is $\sigma^{-2}\,\mathrm{tr}(V\Sigma_{\mathcal{M}}V^\top)$. The penalty gradient is $\lambda(W - W^\star)\Sigma_{\mathcal{M}}$. The proximal step is stationary at $(V - W + G) + \lambda(V - W^\star)\Sigma_{\mathcal{M}} = 0$; with intact memories, $W\Sigma_{\mathcal{M}} = W^\star\Sigma_{\mathcal{M}}$, so $(V - W)(\mathbb{I} + \lambda\Sigma_{\mathcal{M}}) = -G$, and the Woodbury identity gives the OWM form.

Theorem D.6. (a) A correction computed from ω is the same for every consistent A'; by Lemma D.1 it is exact for all of them iff it annihilates every $S_f(A')$, i.e. $\mathcal{E}_f(\omega)$. Then $\|\Delta W x_i\| = \|\Delta W\Pi^\perp_{\mathcal{E}}x_i\| \leq \|\Delta W\|_F\|\Pi^\perp_{\mathcal{E}}x_i\|$, whereas gradient descent attains $\|G\|_F\|x_i\|$. (b) Inputs of a consistent A' lie in the stored input span; if layer 1 has been exact on it, their pre-activations are unchanged and lie in the stored span of layer 2, and induction over layers and steps gives $f^{(j)}_t(A') = f^{(j)}_0(A')$. For linear f, $\mathrm{span}\, f(A') = f(\mathrm{span}\, A)$ for every A' with the same span. (c) Sign-flipped sets $\{s_a a\}_{a\in A}$, $s_a = \pm 1$, have the same second moment, so $\phi(-Wa) \in \mathcal{E}_f(\omega)$; as $\phi(z) - \phi(-z) = z$, $\mathrm{span}\{\phi(Wa), \phi(-Wa)\} = \mathrm{span}\{\phi(Wa), Wa\}$. For one memory with a negative entry in Wa, either Wa also has a positive entry and $Wa \not\parallel \phi(Wa)$, or $\phi(Wa) = 0 \neq Wa$. (d) If $\Omega(A) = \Omega(A')$ with some $a' \in A' \setminus A$, point separation gives f with $f(A) = \{0\}$ and $f(a') \neq 0$, so $S_f(A) = \{0\}$ while $\mathcal{E}_f(\omega) \ni f(a')$. For ReLU networks with biases, choose u with $\langle u, a'\rangle \neq \langle u, b\rangle$ for all $b \in A$, set $c = \langle u, a'\rangle$ and $\rho = \min_{b\in A}|\langle u, b\rangle - c|$; the non-negative hat function $\phi(\langle u, x\rangle - c + \rho) - 2\phi(\langle u, x\rangle - c) + \phi(\langle u, x\rangle - c - \rho)$ equals ρ at a' and 0 on A, and one more ReLU unit passes it on to layer i.

Proposition D.7. (i) $\Delta W_i x^\mu_i(0) = 0$, and $\Delta W_i\delta = \Delta W_i(\Pi_{\hat{\mathcal{M}}_i} + \Pi^\perp_{\hat{\mathcal{M}}_i})\delta = \Delta W_i\Pi^\perp_{\hat{\mathcal{M}}_i}\delta$. (ii) $\Delta W\Pi_{\mathcal{M}} = \Delta W(\Pi_{\mathcal{M}} - \Pi_{\hat{\mathcal{M}}})$; the sample size follows from the Davis–Kahan bound $\|\Pi_r - \hat{\Pi}_r\|_2 \leq 2\sqrt{r}\,\|\hat{\Sigma}_N - \Sigma\|_2/g_r$ and $\mathbb{E}\|\hat{\Sigma}_N - \Sigma\|_2 \lesssim \|\Sigma\|_2\sqrt{\mathbf{r}(\Sigma)/N}$ for $N \gtrsim \mathbf{r}(\Sigma)$.

Remark D.8 (Scope). Zero forgetting at every layer is sufficient, not necessary, for zero forgetting of the outputs: $W_2W_1 = (W_2T^{-1})(TW_1)$ for any invertible T, and positive rescalings play the same role for ReLU units. Exploiting this freedom requires evaluating the current network on the memory inputs, as in Thm. D.6. Prop. 3.3 keeps the direct term $\Delta W_i\,\phi_i(a^\mu_i)$ of the multi-layer drift; with simultaneous updates of all layers the propagated term $W_i(t+1)[\phi_i(a^\mu_i(t+1)) - \phi_i(a^\mu_i(t))]$ must be added, unless activities are patched layer by layer as in Fig. 7. The price of sign-blindness in Thm. D.6(c) is concrete: for Gaussian W and P memories in general position, $\dim(S_f(A) + W\,\mathrm{span}\,A) = P + r$ against $\dim S_f(A) = P$.

D.5 Multi-layer Forgetting

Forgetting Dynamics

This section explores the dynamics of forgetting and, as a sanity check, how forgotten memories can be reintroduced into the model. Given a memory x^μ from a task $\mathcal{T}_A$, with x^μ_i as the latent representation for the i-th layer, and a dataset $\mathcal{T}_B = \{(x^i, y^i)\}_{i=1,\ldots,P}$, we want to determine how

much of the memory is forgotten after training on $\mathcal{T}_B$. In light of equation (4), at each training step t, we can track the updated memory output for a single layer:

$$x^{\mu}_{i+1}(t+1) = x^{\mu}_{i+1}(t) - \beta^{-1}\langle x^{\mu}_i(t), x^t_i\rangle e^t$$

Where $e^t = f(x^t;\theta^t) - y^t$ is the target error at time t.

When considering multilayer networks, the forgetting dynamics becomes more intricate, as the forgetting drift compounds as we go deeper in the NNet. On the other hand, on a layer-wise perspective, the only thing that differs is that the i-th layer's memory input is not static under non-zero forgetting, and thus $\langle x^{\mu}_i, x^t_i\rangle$ varies even when presenting a constant input $x^t = \bar{x}$. To show this, we now consider a more general L-multi-layer network. The layer-wise forgetting drift for the tasks $A \to B$ can be defined after P steps as follows:

Let us assume that we have a memory $x^{\mu}_0 \in \mathcal{T}_A$, with $x^{\mu}_i = f_i(x^{\mu}_{i-1}|\theta^A_i)$, on the other hand, we train on elements x^t_0 from task $\mathcal{T}_B$, with $x^t_i = f_i(x^t_{i-1}|\theta_i) = \phi_i(W^t_{i-1}x^t_{i-1})$ we have:

$$x^{\mu}_{i+1}(t+1) = x^{\mu}_{i+1}(t) - \frac{\langle x^{\mu}_i(t), x^t_i\rangle}{\beta}\prod_{k=0}^{i+1} J_{\phi_{L-k}}W^T_{L-k}(x^t_L - y^t)$$

Where $x^{\mu}_0(t) = x^{\mu}_0\ \forall t$. If we define $\Delta^t_i = x^{\mu}_i(t) - x^{\mu}_i(t-1)$ and

$$\delta^t_i = x^{\mu}_i(t) - x^{\mu}_i(0) = \sum_s^t \Delta^s_i$$

From the equation above and the target definitions for backpropagation of tab. 3 (see appendix C.5) we get:

$$\Delta^t_{i+1} = -\beta^{-1}\Big(< x^{\mu}_i(0), x^t_i > + < \sum_{s=0}^{t-1}\Delta^s_i, x^t_i > \Big)\Big(\prod_{k=0}^{i+1} J_{\phi_{L-k}}W^T_{L-k}\Big)(x^t_L - y_t)$$

$$\delta^P_i = -\beta^{-1}\sum_t^P\Big(< x^{\mu}_i(0), x^t_i > + < \delta^{t-1}_{i-1}, x^t_i > \Big)\Big(\prod_{k=0}^{i+1} J_{\phi_{L-k}}W^T_{L-k}\Big)(x^t_L - y_t)$$

where $\Delta^t_0 = 0$ and $\delta^t_0 = 0\ \forall t$.

As a sanity check, in Figure 7 it is depicted an example of "memory recovery", where we numerically compute these nested integrals as the network is trained on the new task and we use it to recover the performance on the past memory.

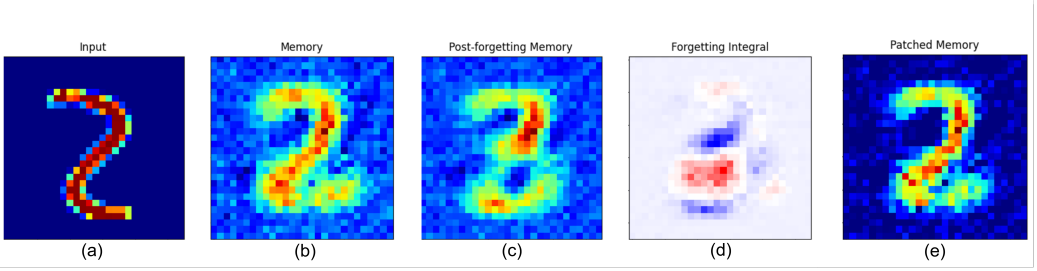


Figure 7: Frogetting Integral for 4-layer AutoEncoder (AE). (a-b) The AE learns to reconstruct the digit 2. (c) After training exclusively on the digit 3 the AE, with the same input as in (a) it now outputs a 3 (memory override). (d) Visualization of the forgetting Integral in the output space. (e) After patching the activities with the forgetting integral from Thm. 3.3 we get back the original memory, showing that the integral really captures the forgetting drift.

D.6 Continuous forgetting under random Gaussian inputs

Assumption D.9 (Memory Anchored Gaussian Equivalence)**.** Given a noisy d-dimensional input $x(t) \sim \mathcal{N}(\bar{x}, \mathbb{I}_d)$, a memory x^μ and their dot product $\alpha(t) = \langle x^\mu, x(t) \rangle \sim \mathcal{N}\left(\omega, \frac{1}{d}\right)$, we make the assumption that a a neural network $f(\cdot, \theta)$ such that $f(x^\mu, \theta) = \mu$, follows:

$$f(x(t); \theta) = \alpha(t)^K \mu + \rho(1 - \alpha(t))^K \chi$$

for some $K \in \mathbb{R}^+$ sensitivity factor, $\chi \in \mathcal{N}(0, \mathbb{I}_m)$ a random Gaussian variable, and $\rho \in \mathbb{R}^+$ the spectral radius of f.

Assumption D.9 is an extended version of a Gaussian Equivalence assumption. Here we consider the case where f acts Gaussian (χ) in its last layer with respect to random gaussian input. The extension comes from the fact that, since $f(x^\mu, \theta) = \mu$, we assume the interpolation between it's behavior on x and x^μ scales as a power of their alignment. Therefore, the K factor encodes the non-linearity of the network with respect to the alignment between x and x^μ. For 1 layer and $K = 1$ this is always exact.

Theorem D.10 (Forgetting Under Random Gaussian Inputs)**.** *Under the Gaussian Equivalence assumption D.9, let $\mu(t) \in \mathbb{R}^m$ represent the memory trace evolving under the stochastic differential equation:*

$$\dot{\mu}(t) = -\beta^{-1}\alpha(t)\left(\alpha(t)^K \mu(t) - y(t)\right) - \beta^{-1}\rho\alpha(t)(1 - \alpha(t))^K \chi(t)$$

where $\chi(t) \sim \mathcal{N}(0, \mathbb{I}_m)$, K is a positive even integer, and $\alpha(t) = \langle x^\mu, x(t) \rangle$ is a scalar random variable encoding input alignment.

Assume the inputs are drawn with controlled noise variance such that $x(t) \sim \mathcal{N}(\bar{x}, \epsilon\mathbb{I}_d)$. For a large input dimension d, the memory trace $\mu(t)$ in expectation over the fast variable $\alpha(t)$ follows a multidimensional Ornstein-Uhlenbeck process:

$$d\mu_t = \left(-\lambda_\epsilon \mu_t + \frac{\omega}{\beta} y(t)\right) dt + \sigma_{eff,\epsilon} dW_t$$

where W_t is an m-dimensional Wiener process. The effective decay rate λ_ϵ and diffusion coefficient $\sigma_{eff,\epsilon}$ are approximated by:

$$\lambda_\epsilon \approx \beta^{-1}\left(\omega^{K+1} + \frac{\epsilon K(K+1)}{2d}\omega^{K-1}\right)$$

$$\sigma^2_{eff,\epsilon} \approx \left(\frac{\rho}{\beta}\right)^2 \left(\omega^2(1-\omega)^{2K} + \frac{\epsilon}{2d} g''(\omega)\right)$$

with $\omega = \langle x^\mu, \bar{x} \rangle$, and the curvature of the gating term $g''(\omega)$ given by:

$$g''(\omega) = (1-\omega)^{2K-2}\left[2(1-\omega)^2 - 8K\omega(1-\omega) + 2K(2K-1)\omega^2\right]$$

Consequently, the distribution of the memory trace $P(\mu, t)$ is Gaussian, with the mean trajectory and variance matrix given by:

$$\mathbb{E}[\mu(t)] = e^{-\lambda_\epsilon t}\mu(0) + \frac{\omega}{\beta}\int_0^t e^{-\lambda_\epsilon(t-s)} y(s) ds$$

$$\mathbb{V}[\mu(t)] = \frac{\sigma^2_{eff,\epsilon}}{2\lambda_\epsilon}\left(1 - e^{-2\lambda_\epsilon t}\right)\mathbb{I}_m$$

Proof. Given $x(t) \sim \mathcal{N}(\bar{x}, \epsilon\mathbb{I}_d)$, the projection $\alpha(t) = \langle x^\mu, x(t) \rangle$ is a linear combination of Gaussian variables and is therefore normally distributed. Its expectation is:

$$\mathbb{E}[\alpha(t)] = \langle x^\mu, \mathbb{E}[x(t)] \rangle = \langle x^\mu, \bar{x} \rangle = \omega$$

From the problem's prior assumption that $x(t) \sim \mathcal{N}(\bar{x}, \mathbb{I}_d)$ yields $\text{Var}(\alpha(t)) = \frac{1}{d}$, we deduce that $\|x^\mu\|^2 = \frac{1}{d}$. Consequently, scaling the variance by ϵ gives:

$$\text{Var}(\alpha(t)) = (x^\mu)^\top(\epsilon\mathbb{I}_d)x^\mu = \epsilon\|x^\mu\|^2 = \frac{\epsilon}{d}$$

Thus, $\alpha(t) \sim \mathcal{N}\left(\omega, \frac{\epsilon}{d}\right)$.

We define the corresponding stochastic Fokker-Planck equation* for $\mu(t)$ averaged over the distribution of $\alpha(t)$. The effective drift vector $\bar{A}(\mu, t)$ is:

$$\bar{A}(\mu, t) = \mathbb{E}_{\alpha(t)}\left[-\beta^{-1}\alpha(t)^{K+1}\mu + \beta^{-1}\alpha(t)y(t)\right] = -\beta^{-1}\mathbb{E}[\alpha^{K+1}]\mu + \beta^{-1}\omega y(t)$$

For a Gaussian random variable $Z \sim \mathcal{N}(\omega, \sigma^2)$, its moments can be expanded for small variance ($\sigma^2 = \frac{\epsilon}{d} \ll 1$) using Taylor expansion $f(Z) \approx f(\omega) + \frac{1}{2}f''(\omega)\sigma^2$. Applying this to $f(\alpha) = \alpha^{K+1}$:

$$\mathbb{E}[\alpha^{K+1}] \approx \omega^{K+1} + \frac{1}{2}(K+1)K\omega^{K-1}\left(\frac{\epsilon}{d}\right)$$

We define the effective decay rate $\lambda_\epsilon \triangleq \beta^{-1}\mathbb{E}[\alpha^{K+1}]$, confirming the formulation in the theorem.

The effective diffusion matrix $\bar{D}(\mu, t)$ represents the variance of the stochastic driving force, scaled by the gating variables:

$$\bar{D}(\mu, t) = \frac{1}{2}\mathbb{E}_{\alpha(t)}\left[\left(\beta^{-1}\rho\alpha(t)(1-\alpha(t))^K\right)^2 \mathbb{I}_m\right] = \frac{\rho^2}{2\beta^2}\mathbb{E}[\alpha^2(1-\alpha)^{2K}]\mathbb{I}_m$$

Let $g(\alpha) = \alpha^2(1-\alpha)^{2K}$. Applying the second-order Taylor approximation around ω:

$$\mathbb{E}[g(\alpha)] \approx g(\omega) + \frac{\epsilon}{2d}g''(\omega)$$

We compute $g''(\omega)$ by taking derivatives. The first derivative is:

$$g'(\omega) = 2\omega(1-\omega)^{2K} - 2K\omega^2(1-\omega)^{2K-1}$$

The second derivative evaluates to:

$$g''(\omega) = 2(1-\omega)^{2K} - 8K\omega(1-\omega)^{2K-1} + 2K(2K-1)\omega^2(1-\omega)^{2K-2}$$

Factoring out $(1-\omega)^{2K-2}$ yields:

$$g''(\omega) = (1-\omega)^{2K-2}\left[2(1-\omega)^2 - 8K\omega(1-\omega) + 2K(2K-1)\omega^2\right]$$

Since $\bar{D}(\mu, t) = \frac{1}{2}\sigma^2_{eff,\epsilon}\mathbb{I}_m$, substituting the expectation gives the effective diffusion coefficient $\sigma_{eff,\epsilon}$.

Replacing the fast-fluctuating variable $\alpha(t)$ with its average effect reduces the dynamics to a standard linear stochastic differential equation*:

$$d\mu_t = \left(-\lambda_\epsilon\mu_t + \frac{\omega}{\beta}y(t)\right)dt + \sigma_{eff,\epsilon}dW_t$$

This is an exact formulation of the multidimensional Ornstein-Uhlenbeck (OU) process. Using Itô's Lemma, we solve for $\mu(t)$ using the integrating factor $e^{\lambda_\epsilon t}$. The solutions for the first two moments (expectation and variance) of the standard OU process complete the proof. □

Corollary D.11 (Equilibrium Distribution for a Constant Target)**.** *Assume the setting of Theorem 1. Let the target signal be constant over time, such that $y(t) = y$ for all $t \geq 0$, and assume $\omega > 0$ so that $\lambda_\epsilon > 0$. As $t \to \infty$, the probability distribution of the memory trace $\mu(t)$ converges to a unique stationary multivariate Gaussian distribution $P_{eq}(\mu) = \mathcal{N}(\mu_{eq}, \Sigma_{eq})$, with the equilibrium mean μ_{eq} and covariance matrix Σ_{eq} given by:*

$$\mu_{eq} = \frac{\omega}{\beta\lambda_\epsilon}y \approx \left(\omega^K + \frac{\epsilon K(K+1)}{2d}\omega^{K-2}\right)^{-1}y \tag{19}$$

$$\Sigma_{eq} = \frac{\sigma^2_{eff,\epsilon}}{2\lambda_\epsilon}\mathbb{I}_m \approx \frac{\rho^2\left(\omega^2(1-\omega)^{2K} + \frac{\epsilon}{2d}g''(\omega)\right)}{2\beta\left(\omega^{K+1} + \frac{\epsilon K(K+1)}{2d}\omega^{K-1}\right)}\mathbb{I}_m \tag{20}$$

where $g''(\omega)$ is the curvature of the gating term defined in Theorem 1.

Proof. By Theorem 1, the memory trace μ_t follows a multidimensional Ornstein-Uhlenbeck process. For a constant target $y(t) = y$, the stochastic differential equation* simplifies to:

$$d\mu_t = \lambda_\epsilon \left(\frac{\omega}{\beta \lambda_\epsilon} y - \mu_t \right) dt + \sigma_{eff,\epsilon} dW_t$$

The closed-form solutions for the mean and variance of $\mu(t)$ from Theorem 1 can be explicitly evaluated. For the expected value, the integral of the constant target y becomes:

$$\mathbb{E}[\mu(t)] = e^{-\lambda_\epsilon t}\mu(0) + \frac{\omega}{\beta} y \int_0^t e^{-\lambda_\epsilon (t-s)} ds$$

Evaluating the integral yields:

$$\int_0^t e^{-\lambda_\epsilon (t-s)} ds = \left[\frac{1}{\lambda_\epsilon} e^{-\lambda_\epsilon (t-s)} \right]_s = 0^{s=t} = \frac{1 - e^{-\lambda_\epsilon t}}{\lambda_\epsilon}$$

Substituting this back into the expectation:

$$\mathbb{E}[\mu(t)] = e^{-\lambda_\epsilon t}\mu(0) + \frac{\omega}{\beta \lambda_\epsilon} y \left(1 - e^{-\lambda_\epsilon t}\right)$$

Because the mean alignment $\omega > 0$ and $\epsilon \geq 0$, the effective decay rate λ_ϵ is strictly positive. Taking the limit as $t \to \infty$, the transient terms $e^{-\lambda_\epsilon t}$ vanish:

$$\mu_{eq} = \lim_{t \to \infty} \mathbb{E}[\mu(t)] = \frac{\omega}{\beta \lambda_\epsilon} y$$

Similarly, the variance from Theorem 1 is given by:

$$\mathbb{V}[\mu(t)] = \frac{\sigma_{eff,\epsilon}^2}{2\lambda_\epsilon} \left(1 - e^{-2\lambda_\epsilon t}\right) \mathbb{I}_m$$

Taking the limit as $t \to \infty$:

$$\Sigma_{eq} = \lim_{t \to \infty} \mathbb{V}[\mu(t)] = \frac{\sigma_{eff,\epsilon}^2}{2\lambda_\epsilon} \mathbb{I}_m$$

To obtain the explicit approximations, we substitute the expressions for λ_ϵ and $\sigma_{eff,\epsilon}^2$ established in Theorem 1. For the mean:

$$\mu_{eq} \approx \frac{\omega}{\beta \left[\beta^{-1} \left(\omega^{K+1} + \frac{\epsilon K(K+1)}{2d} \omega^{K-1} \right) \right]} y = \frac{1}{\omega^K + \frac{\epsilon K(K+1)}{2d} \omega^{K-2}} y$$

For the equilibrium covariance, substituting $\sigma_{eff,\epsilon}^2$ and λ_ϵ yields:

$$\Sigma_{eq} \approx \frac{\left(\frac{\rho}{\beta}\right)^2 \left(\omega^2 (1-\omega)^{2K} + \frac{\epsilon}{2d} g''(\omega) \right)}{2\beta^{-1} \left(\omega^{K+1} + \frac{\epsilon K(K+1)}{2d} \omega^{K-1} \right)} \mathbb{I}_m$$

Simplifying the constants by canceling β^{-1} from the numerator and denominator gives the final stated expression for Σ_{eq}. Since the SDE is linear with additive Gaussian noise, the invariant measure is uniquely defined by this stationary mean and covariance, concluding the proof. □

In Figure 8 we confront the empirical evaluation of the equilibrium distribution of forgotten memories against theoretical predictions. The implications of this theorem are that even the noise of novel tasks training distribution can hinder memories. We will therefore look for a solution that is resilient to possible drifts of the memories. In other words, mitigating forgetting is not enough, we need an elastic force that pulls them back.

It is worth observing that, as a consequence of proposition **??**, when looking at the intralayer level the bias in the α_i distribution between each layer is not stationary. If $\langle x_i^\mu, x_i \rangle \sim \mathcal{N}(\omega_i, \frac{1}{d_i})$, and $\langle x_i^\mu, x_i \rangle = \langle x_i^\mu, \bar{x}_i \rangle + \langle x_i^\mu, \delta x_i \rangle = \omega + \langle x_i^\mu, \delta x_i \rangle$, it means that $\omega(t) = \langle x_i^\mu(t), \bar{x}_i \rangle$, with $x_i^\mu(t)$ following the dynamics of equation D.5.

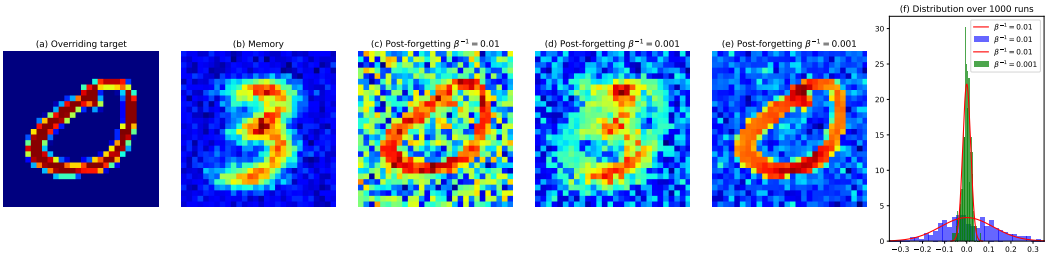


Figure 8: Continuous forgetting under random Gaussian inputs. For an overriding target (a) and a memory (b) we can have different "diffusive" forgetting dynamics. For an high learning rate we get (c) while for a lower one we get first (d) and then (e). Finally, (f) shows how the distribution of the stable memory matches the theoretical predictions.

E Experimental methodology

All methods, baselines included, are run with the same learner, the same code path and the same evaluation protocol; they differ only in the terms listed in §E.3. Hyperparameters are selected on a held-out validation split and the final numbers are measured once on the official test sets with seeds that were never used for selection (§E.5).

E.1 Benchmarks and input representations

Table 5: Benchmarks. Classes are split into tasks in their natural order (MNIST digits, CIFAR-10 label order, Tiny-ImageNet sorted WordNet ids), identical for all seeds. The protocol is class-incremental: a single output head over all classes, no task identity at test time. Features: frozen ImageNet ResNet-18, post-ReLU, same as in de Haan et al. (2026). †The labelled official validation set serves as the test set.

	Split-MNIST	Split-CIFAR-10	Split-Tiny-ImageNet
tasks × classes	5×2	5×2	10×20
train / test images	60k / 10k	50k / 10k	100k / 10k†
input x	pixels, $[0,1]^{784}$	features, $\mathbb{R}^{512}_{\geq 0}$	features, $\mathbb{R}^{512}_{\geq 0}$
network	784–100–100–10	512–1000–10	512–1000–200
parameters	89.6k	523k	713k

Frozen encoder. Following de Haan et al. (2026), for CIFAR-10 and Tiny-ImageNet the input is the 512-dimensional global-average-pooled output of the torchvision ImageNet ResNet-18 (`IMAGENET1K_V1`) He et al. (2016), applied once to every image (resized to 224×224, ImageNet normalisation) and cached; no gradient ever reaches the encoder, so all forgetting happens in the trained network. The features are post-ReLU, L2-normalised per sample and scaled by 10 (the mean norm of a flattened MNIST digit, the regime the learning rule was developed in). A ridge linear probe on the CIFAR-10 features reaches 86.0% test accuracy (87.1% train). Split-MNIST uses the raw pixels.

E.2 Network and learner

The trained network is an MLP with ReLU hidden units, biases, a linear readout and a softmax output $p = \mathrm{softmax}(z)$. Every method is trained with the same layer-wise rule, the HOP operator of §2 with backpropagated targets (the $\lambda = 0$ end of the interpolation of §2.2), followed by Adam:

1. *Targets.* For a mini-batch of N rows b (stream and replay rows together) with per-row weights w_b, the output error is $\delta_{L,b} = \frac{2}{D_{\text{out}}} w_b (p_b - t_b)$ applied directly at the logits (t_b one-hot or a stored distillation target; the softmax Jacobian is not applied, i.e. the cross-entropy gradient scaled by $2/D_{\text{out}}$). It is backpropagated through the current weights and the ReLU

derivatives, and every layer receives the target $b_{l,b} = a_{l,b} - \eta\,\delta_{l,b}$, where $a_{l,b}$ is its current pre-activation.

2. *HOP step.* Each layer, with input $h_{l,b}$ (the post-activation of the previous layer; $h_{0,b} = x_b$), computes

$$\widetilde{W}_l \;=\; W_l - \frac{1}{N}\sum_b (W_l h_{l,b} - b_{l,b})\, h_{l,b}^\top \;=\; W_l - \frac{\eta}{N}\sum_b \delta_{l,b}\, h_{l,b}^\top, \tag{21}$$

i.e. a step of size η along the backpropagated error signal (Lemma 2.1); the bias follows the mean residual. All layers use the same forward pass.

3. *Optimiser.* $G_l = W_l - \widetilde{W}_l$ is passed to Adam as the gradient (step η, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay 0, moments reset at the start of every epoch). Corrections (§E.3.1) are added to $\widetilde{W}_l$ before this step, so Adam normalises the sum.

Each task is trained for 20 epochs (one epoch = one pass over the task's training data); mini-batches hold 32 current-task rows, to which each method appends its replay rows (§E.3). The learning rate is benchmark-specific and selected on the validation split (§E.5): $\eta = 10^{-4}$ (Split-MNIST), $3 \cdot 10^{-5}$ (Split-CIFAR-10), $3 \cdot 10^{-4}$ (Split-Tiny-ImageNet).

E.3 Methods

Buffer. Methods that store samples use a reservoir buffer of total capacity B over the input stream (Buzzega et al., 2020); it stores the input, the label and the network output $p(x)$ at the time x was seen, and holds samples of the current task as well. $B \in \{10, 50, 200, 500, 1000\}$. Each step draws replay rows uniformly without replacement.

- **ER**: 32 buffer rows with their one-hot labels, weight 1.
- **DER++**: 32 buffer rows distilled towards their stored outputs $p(x)$ (MSE, weight $\alpha = 0.5$) and 32 further rows with their labels (weight $\beta = 1$). The HOPc variant is built on $\alpha = \beta = 0.5$, which is also reported as its paired control.
- **OGD** (Farajtabar et al., 2020): at every task boundary the gradients $\nabla_\theta z_c(x)$ of all output units (Split-MNIST, Split-CIFAR-10) or of the true-class unit (Split-Tiny-ImageNet, 200 outputs) at n samples per class of the finished task are Gram–Schmidt orthonormalised into a basis of at most B directions, with n chosen to spread the budget evenly over the task boundaries; the step G is projected onto the basis' orthogonal complement before Adam. OGD adds no replay rows.
- **Class-Gaussian replay** (a generative proxy, cf. Theorem 3.2): at each task boundary, for every class c of the finished task, the mean μ_c and the top-k eigenpairs $(\lambda_{c,i}, u_{c,i})$ of the covariance of *all* its training inputs are stored and the samples discarded. Each step appends 32 fresh draws $\tilde{x} = \mu_c + \sum_{i \le k} \zeta_i \sqrt{\lambda_{c,i}}\, u_{c,i}$, $\zeta \sim \mathcal{N}(0, I)$, c uniform over the stored classes, labelled c (weight 1). Draws are clipped at 0 (the inputs are non-negative), at 1 for pixels, and, for the L2-normalised features, rescaled to the class' mean norm. k is the largest value whose storage fits the buffer budget (§E.4).

E.3.1 HOPc with boundary principal components

The memories of HOPc are, per layer, the principal directions of the past tasks' representations, estimated from all their training samples at the task boundary, together with the layer's action on them at that time (Algorithm 2). The corrector preserves each layer's input–output map on this subspace as it was learned, rather than a subspace re-estimated during training.

Construction (end of task t). All training inputs of task t are passed through the network $W^{(t)}$. For each layer l the top-k eigenvectors $u_{l,j}$ of the uncentred second moment $\frac{1}{n}\sum h_l h_l^\top$ of its inputs are stored with their eigenvalues $s_{l,j}$ and their images $y_{l,j} = W_l^{(t)} u_{l,j}$ (no bias: these are directions). The sets of successive tasks are merged into one orthonormal basis per layer: with $A_l = [\sqrt{s_{l,j}}\, u_{l,j}]_j$ stacked over all stored components, the new components are the top-k right singular vectors $v_{l,m}$ of A_l with weights $\sigma_{l,m}^2$, and their targets are given by the weighted least-squares map through the stored

pairs, $y_{l,m} = \big(\sum_j s_{l,j}\, y_{l,j} u_{l,j}^\top\big) v_{l,m}/\sigma_{l,m}^2$, which reproduces every stored image exactly whenever nothing is discarded. $k = 10$ per layer (smaller for HOPc alone at the smallest buffers, §E.4).

Correction (every step, every layer). With the orthonormal components u_j, their stored images y_j, the batch inputs h_b of the layer, the alignments $\alpha_{bj} = \langle h_b, u_j \rangle$ and the residuals $r_b = (W h_b - b_b)/N$ of equation 21, the HOP step is corrected as $\widetilde{W} \leftarrow \widetilde{W} + \lambda H$ with

$$H = \sum_j d_j\, u_j^\top, \qquad d_j = \underbrace{\sum_b \frac{\alpha_{bj}^2}{\sum_{b'} \alpha_{b'j}}\, r_b}_{\text{drift cancellation}} \; - \; \underbrace{\frac{1}{\beta_e}\big(W u_j - y_j\big)}_{\text{elastic pull}}, \tag{22}$$

$\lambda = 0.1$, $\beta_e = 64$. Eq. equation 22 (HOP correction of §3.1 with the elastic term of Algorithm 1) is specialised to orthonormal memories; it is exact when the batch rows and the memories are linearly independent, and otherwise batch rows without a residual after Gram–Schmidt are dropped from the first term. The first term removes a small alignment-weighted share of the step's drift on u_j (a few percent, measured); the elastic term, which restores the layer's action on u_j to its value at the end of the task that produced it, carries the effect. Without it the gain over the base largely disappears (Split-CIFAR-10 validation, HOPc + DER++: $+0.8$ to $+1.8$ pp at $B = 50$ and ≈ 0 at $B = 200$, against $+11$ to $+15$ pp and $+3$ pp with it). Because the elastic term does not scale with η while the HOP residual does, λ and β_e are tied to the learning rate; the values above were selected at $\eta = 3 \cdot 10^{-5}$ on Split-CIFAR-10.

Algorithm 2 HOPc with boundary principal components (per layer l; `reduce=boundary-eig`)

Require: k, λ, β_e; stored (U_l, S_l, Y_l) (components, weights, images)
  for each task t **do**
    for each mini-batch **do**
      build the batch of the base method (stream rows + replay rows)
      compute targets b_l and the HOP solution $\widetilde{W}_l$ (Eq. 21)
      if $U_l \neq \emptyset$ **then**
        $\widetilde{W}_l \leftarrow \widetilde{W}_l + \lambda H(U_l, Y_l)$ ▷ Eq. 22
      end if
      $W_l \leftarrow \mathrm{Adam}(W_l,\ W_l - \widetilde{W}_l)$
    end for
    $H_l \leftarrow$ inputs of layer l for all training samples of task t under $W^{(t)}$
    $(U', S') \leftarrow$ top-k eigenpairs of $\frac{1}{n} H_l^\top H_l$; $\quad Y' \leftarrow U' W_l^\top$
    $(U_l, S_l, Y_l) \leftarrow \textsc{Merge}\big((U_l, S_l, Y_l), (U', S', Y')\big)$ ▷ weighted SVD + least-squares images
  end for

Variants. *HOPc*: the corrector alone; the loss sees only the current batch (the reservoir is kept only for the stopping rule of §E.5). *HOPc + ER* and *HOPc + DER++*: the corrector on top of the ER / DER++ ($\alpha = \beta = 0.5$) batch. *HOPc + Gaussian replay*: the corrector on top of class-Gaussian replay, $k \in \{5, 10\}$ selected on validation.

E.3.2 HOPc (anchor) with Gaussian replay

Here the class Gaussians serve both as the generator and as the memory whose input–output map is preserved at the readout layer. For each stored class, the $k_c + 1$ affine points $P_c = \{\mu_c,\ \mu_c + \sqrt{\lambda_{c,i}}\, u_{c,i}\}_{i \le k_c}$ of its input Gaussian are kept with their logits $Z_c = z^{(t)}(P_c)$ at the end of task t ($k_c = 4$, or the largest value that fits the budget). At every step the points are passed through the current network to the readout input H_P, and the readout step receives a restoring pull of fixed relative size,

$$\widetilde{W}_L \leftarrow \widetilde{W}_L + \kappa \left\|\widetilde{W}_L - W_L\right\|_F \frac{-E^\top [H_P, \mathbf{1}]}{\left\|E^\top [H_P, \mathbf{1}]\right\|_F}, \qquad E = H_P W_L^\top + \mathbf{b}_L - Z, \tag{23}$$

$\kappa = 0.3$. Because the pull is a fixed fraction of the plain step it is independent of η and of Adam's normalisation. The stored logits are counted in the memory budget, and the replay Gaussians are shrunk to pay for them.

E.4 MEMORY ACCOUNTING

Storage is counted in floats and reported in *sample-equivalents*, stored floats divided by the input dimension D (784 or 512). A reservoir of B samples costs B.

- *Gaussian replay* is matched to the budget: with n_{old} classes stored before the last task (8, 8, 180), k is the largest integer with $n_{\text{old}}\big((k+1)D + k\big) \le BD$, i.e. $k = 0, 5, 23, 61, 123$ on the 10-class benchmarks for $B = 10, \dots, 1000$ and $k = 0, 1, 4$ on Split-Tiny-ImageNet for $B = 200, 500, 1000$. Below $B \approx 180$ Split-Tiny-ImageNet cannot store even the class means, and Gaussian replay is not run there. The anchored variant uses the same budget including its stored logits.
- *Boundary components* cost $k\,(d_l + d_{l+1} + 1)$ floats per layer (direction, image, weight): 10 components per layer are 15 (Split-MNIST), 49 (Split-CIFAR-10) and 53 (Split-Tiny-ImageNet) sample-equivalents. For HOPc + ER / DER++ / Gaussian replay they are stored *in addition* to the buffer and are reported next to every result; HOPc alone spends the budget on them,
$$k = \min\Big(10,\ \Big\lfloor \frac{BD}{\sum_l (d_l + d_{l+1} + 1)} \Big\rfloor\Big).$$
- *OGD* stores one parameter-sized vector per direction (89.6k–713k floats, i.e. 114–1393 sample-equivalents each); its budget is B directions, which gives it by far the largest memory.

E.5 MODEL SELECTION, STOPPING RULES AND EVALUATION

Splits and seeds. A stratified 10% of every training set (fixed split) is held out as validation. All selection uses the validation split, with 3 seeds for method hyperparameters (seed 8, plus seed 42 for ER at $B = 200$, for the learning-rate scan). The reported numbers train on the full training set and are measured on the official test set with 4 seeds. The seed controls initialisation, data order, buffer sampling and Gaussian draws; the task split is fixed.

Learning rate (shared by all methods of a benchmark). Scanned over $\{10^{-3}, 3\cdot10^{-4}, 10^{-4}, 3\cdot10^{-5}, 10^{-5}\}$ with ER at $B \in \{50, 200, 1000\}$, DER++ at $B = 200$ and JOINT. We take the rate with the best mean ER/DER++ accuracy, excluding rates whose JOINT accuracy is more than ≈ 1 pp below the best JOINT (underfitting); ties within 0.5 pp go to the higher JOINT. With a fixed epoch budget the rate strongly controls forgetting (ER at $B = 200$ on Split-CIFAR-10: 61% at 10^{-3}, 72% at $3\cdot10^{-5}$, while JOINT moves from 88.5% to 87.8%), so it is tuned on the reference baselines, not on the correctors.

Method hyperparameters. DER++ (α, β), OGD's variant, the HOPc settings (k, λ, β_e, the component source) and κ were selected on the Split-CIFAR-10 validation split with 3 seeds per buffer size and transferred unchanged to the other two benchmarks; for HOPc + Gaussian replay $k \in \{5, 10\}$ is selected on each benchmark's validation split.

Stopping rules. The result is reported under an early stop protocol. *Balanced rollback*: after every epoch the score $\mathrm{CE}_{\text{buffer}}(\text{previous classes}) + \mathrm{CE}_{\text{held-out}}(\text{current classes})$ is computed, where the first term uses the stored buffer rows of earlier tasks (for Gaussian replay a fixed set of 100 draws per class) and the second a held-out 5% of the current task's *training* data (removed from training in this protocol only). Training stops at the first epoch whose score is worse than the best so far and the best weights are restored. The rule never reads validation or test data. Its buffer term reads the B stored samples even for methods that do not replay them (HOPc alone, OGD). We do not use rollback on the buffer loss alone: at small learning rates it restores a first epoch that has not yet learned the new task.

Metrics. The headline metric is the top-1 class-incremental accuracy over the whole test set after the last task (argmax over all classes). We also record the task-accuracy matrix R_{ij} (accuracy on task j after task i) and

$$\text{forgetting} = \tfrac{1}{T-1}\textstyle\sum_{j<T}\big(\max_{i\ge j} R_{ij} - R_{Tj}\big), \qquad \text{plasticity} = \tfrac{1}{T}\textstyle\sum_j R_{jj}, \qquad \text{old-task accuracy} = \tfrac{1}{T-1}\textstyle\sum_{j<T} R_{Tj}.$$

Tables report mean $\pm$ sample standard deviation over the four test seeds and, for comparisons, the mean per-seed paired difference with the number of seeds on which it is positive. JOINT (all classes as one task) and naive fine-tuning bound every table.

E.6 Implementation and compute

The learner and all correctors are implemented in NumPy (float64). Every run is single-threaded, with the BLAS thread count pinned to one, because the reduction order of multithreaded BLAS measurably changes results (up to 2 pp through Adam's normalisation); at a fixed thread count runs are deterministic, and re-running a configuration reproduces its accuracy, task matrix and per-class accuracies exactly. A GPU is used only once, to encode the image datasets (an RTX 3090, about two minutes for CIFAR-10); a GPU training step was measured to be slower than the CPU step at these matrix sizes. The final tables (792 runs, plus 312 for HOPc + Gaussian replay) were run as independent single-core jobs on five 128-core CPU nodes, 120 at a time per node; the 792-run sweep took 5.3 h wall-clock. The median trial took 27 / 130 / 189 min (Split-MNIST / Split-CIFAR-10 / Split-Tiny-ImageNet) with fixed epochs and 11 / 100 / 144 min under balanced rollback.

Table 6: Hyperparameters of the final tables.

shared	20 epochs/task; 32 stream rows (+32 replay, +32 distillation for DER++); Adam $(0.9, 0.999)$, $\epsilon = 10^{-8}$, reset every epoch; weight decay 0; η per benchmark: 10^{-4} / $3 \cdot 10^{-5}$ / $3 \cdot 10^{-4}$ (MNIST / CIFAR-10 / Tiny)
DER++	$\alpha = 0.5$, $\beta = 1$ (HOPc base and control: $\alpha = \beta = 0.5$)
OGD	B directions; all output units (10 classes) / true class (200); projection before Adam
Gaussian replay	k storage-matched; draws clipped at 0 (pixels also at 1), features renormalised
HOPc	boundary components, $k = 10$ per layer, all layers; $\lambda = 0.1$, $\beta_e = 64$
HOPc + Gaussian replay	as HOPc, $k \in \{5, 10\}$ by validation
HOPc (anchor)	readout layer, $k_c = 4$ points per class (budget permitting), $\kappa = 0.3$

E.7 Results

Split-MNIST

Learning rate: 10^{-4}; **naive:** 19.74; **JOINT:** 97.62 ± 0.14.

Table 7: Split-MNIST, protocol = bal. Accuracy (%) $\pm$ SD.

Method	Buf. 10	Buf. 50	Buf. 200	Buf. 500	Buf. 1000
ER	27.84 ± 1.26	50.36 ± 1.85	75.01 ± 2.09	84.34 ± 1.11	89.05 ± 0.93
DER++	29.92 ± 0.66	57.79 ± 2.53	79.17 ± 1.92	$\mathbf{89.14 \pm 0.49}$	$\mathbf{91.12 \pm 0.58}$
DER++ $\alpha = \beta = .5$	29.81 ± 1.91	58.75 ± 2.53	79.35 ± 2.02	88.08 ± 0.98	90.84 ± 0.32
OGD	19.05 ± 0.10	20.21 ± 0.96	23.53 ± 3.01	25.51 ± 1.80	28.92 ± 3.23
HOPc	$\mathbf{75.40 \pm 3.06}$	$\mathbf{83.12 \pm 1.53}$	$\mathbf{83.49 \pm 1.21}$	83.36 ± 0.65	83.01 ± 0.09
HOPc + ER	56.01 ± 1.84	59.54 ± 0.56	72.29 ± 2.00	81.11 ± 1.39	85.84 ± 0.38
HOPc + DER++	55.76 ± 0.84	69.80 ± 4.58	78.10 ± 1.68	84.54 ± 1.03	88.89 ± 0.53
Generative methods					
Gaussian replay	74.10 ± 1.44	83.58 ± 0.44	88.12 ± 0.59	87.25 ± 0.49	85.80 ± 0.91
HOPc + Gaussian replay	$\mathbf{76.93 \pm 1.33}$	$\mathbf{86.04 \pm 0.20}$	$\mathbf{89.56 \pm 0.46}$	$\mathbf{89.51 \pm 0.57}$	$\mathbf{89.28 \pm 0.89}$

Split-CIFAR-10

Learning rate: 3×10^{-5}; **naive:** 19.76; **JOINT:** 87.21 ± 0.13.

Split-TinyImageNet

Learning rate: 3×10^{-4}; **naive:** 16.87; **JOINT:** 66.14 ± 0.24.

Limitations: To Sample or to Constrain?

While standard replay methods interleave old inputs and targets into the training batch to brute-force an approximation of the previous data distribution, HOPc explicitly patches the mathematical drift

Table 8: Split-CIFAR-10, protocol = bal. Accuracy (%) ± SD.

Method	Buf. 10	Buf. 50	Buf. 200	Buf. 500	Buf. 1000
ER	32.03 ± 5.06	57.60 ± 0.11	71.54 ± 1.58	77.48 ± 1.12	80.09 ± 0.27
DER++	36.95 ± 4.18	65.94 ± 1.69	76.04 ± 1.34	80.91 ± 0.62	$\mathbf{82.73 \pm 0.23}$
DER++ $\alpha = \beta = .5$	36.68 ± 4.93	65.41 ± 1.83	76.77 ± 0.43	80.29 ± 0.56	82.44 ± 0.28
OGD	21.84 ± 0.74	41.00 ± 1.85	46.25 ± 1.61	52.81 ± 4.01	59.35 ± 1.74
HOPc	69.84 ± 1.05	$\mathbf{81.02 \pm 0.65}$	80.10 ± 0.38	80.61 ± 0.15	80.31 ± 0.03
HOPc + ER	73.34 ± 0.22	75.34 ± 0.92	78.45 ± 0.46	80.60 ± 0.52	81.95 ± 0.50
HOPc + DER++	$\mathbf{76.91 \pm 1.45}$	77.61 ± 1.22	$\mathbf{80.47 \pm 0.39}$	$\mathbf{82.10 \pm 0.33}$	82.70 ± 0.15
Generative methods					
Gaussian replay	76.09 ± 0.15	80.63 ± 0.16	84.21 ± 0.13	$\mathbf{85.05 \pm 0.44}$	$\mathbf{84.90 \pm 0.44}$
HOPc + Gaussian replay	$\mathbf{78.20 \pm 0.30}$	$\mathbf{81.88 \pm 0.12}$	$\mathbf{84.65 \pm 0.20}$	84.95 ± 0.15	84.78 ± 0.33

Table 9: Split-TinyImageNet, protocol = bal. Accuracy (%) ± SD.

Method	Buf. 10	Buf. 50	Buf. 200	Buf. 500	Buf. 1000
ER	12.41 ± 0.23	16.55 ± 0.07	26.79 ± 0.74	35.06 ± 0.29	42.49 ± 0.50
DER++	15.55 ± 0.75	26.87 ± 0.28	40.01 ± 0.76	47.21 ± 0.24	50.80 ± 0.76
DER++ $\alpha = \beta = .5$	16.03 ± 1.69	27.28 ± 1.71	39.16 ± 0.29	46.58 ± 0.94	51.39 ± 0.69
OGD	8.30 ± 0.24	8.42 ± 0.24	9.64 ± 1.28	10.66 ± 0.69	–
HOPc	$\mathbf{27.04 \pm 0.77}$	$\mathbf{37.16 \pm 0.50}$	38.11 ± 0.44	37.94 ± 0.79	38.19 ± 0.51
HOPc + ER	22.67 ± 1.28	25.68 ± 0.29	32.18 ± 0.45	38.90 ± 1.21	44.52 ± 0.46
HOPc + DER++	25.91 ± 3.42	33.63 ± 1.28	$\mathbf{42.09 \pm 0.45}$	$\mathbf{48.56 \pm 0.64}$	$\mathbf{52.26 \pm 0.28}$
Generative methods					
Gaussian replay	–	–	–	53.30 ± 0.62	55.04 ± 1.02
HOPc + Gaussian replay	–	–	–	$\mathbf{57.26 \pm 0.48}$	$\mathbf{58.20 \pm 0.09}$

introduced by training. By orienting the null space of the weight update (Fig. 4), HOPc only requires enough samples to locate the specific *latent subspace* encoding the memories. This geometrical constraint drastically reduces the number of samples required for stable retention compared to matching a full data distribution.

However, this strict preservation introduces a profound new challenge. HOPc effectively compartmentalizes old memories and new data, effectively blocking any cross-task generalization. True abstraction is inherently disruptive: existing features must be integrated, reorganized, and sometimes overwritten to accommodate broader representations. Therefore, completely eliminating forgetting by perfectly nullifying compressive interference leaves no room for cross-task compression (see App.B.3). In other words, absolute stability precludes abstract plasticity. Future work will address this limitation by studying cross-task compression with minimal samples, specifically focusing on adaptive mechanisms to dynamically tune the maximum alignment angle α_{max} such that a memory is preserved. This would allow the network to autonomously navigate the **stability-plasticity trade-off** (Mermillod et al., 2013), tuning the boundary between protecting essential task-specific subspaces and yielding to the compressive forces necessary for global generalization.