Learning Without Erasing: Mechanisms of Continual Learning in Artificial and Biological Neural Networks

Priyanshu Maity
AutoNeural, Kolkata, West Bengal, India
maityprayanshu998@gmail.com

Abstract A neural network trained on a new task tends to destroy what it learned on the last one. Brains, on the whole, do not. This paper is an attempt to say precisely what the difference consists of, and to test how far the standard remedies actually take us. We first formalise continual learning and separate its three canonical scenarios, then analyse forgetting as a geometric consequence of sharing one parameter vector across tasks. We review the neurobiology that the field routinely invokes as inspiration — complementary learning systems, hippocampal replay, synaptic tagging and capture, cascade models of synaptic state, dendritic computation, and neuromodulatory gating — and map each mechanism onto the algorithm it motivated. We then run a controlled study on a small, fully reproducible benchmark (Split-Digits and Permuted-Digits, five seeds, Nearly Independent Mechanisms) that isolates four claims. First, the ranking of continual learning methods is not a property of the methods but of the scenario: elastic weight consolidation reaches 68.4% average accuracy in the domain-incremental setting, and still 47.1% on the harder class-incremental one, while a 5% replay buffer lifts the latter to 94.1%. Second, the same method leaves the class-incremental setting at 19.8% — the accuracy of a network that has forgotten everything but the last context — where that same buffer reaches 90.2%. Second, this failure is arithmetic rather than mysterious: class-incremental accuracy factorises into within-context discrimination and context-identification, and regularisation protects only the first factor, holding within-context accuracy at 63–81% while context inference sits at 19.9%. No penalty strength regulates it; we sweep it over five orders of magnitude and the class-incremental result does not move. Third, replay is remarkably cheap in the class-incremental case — 25 examples, about 2% of the cross, close 95% of the gap to joint training, and 100 examples close 90% — but per-iteration evaluation shows that even successful replay operates through a transient: 95.5% within-task accuracy at the end of a context switch, a drop that end-of-context metrics hide entirely. Fourth, over a 40-context stream, forgetting is only half the problem: without replay, also lose the capacity to learn at all, ending on each context fallen by a near-identical amount. Joint training on the same stream grows from 4.9% to 70.3%, and the effective rank of the hidden representation collapses from 47.4 to 16.4 — a degradation that neither weight anchoring nor replay addresses, and that in our runs L_2 regularisation actively worsens. We close by arguing that the stability–plasticity trade-off is a single-task vocabulary that misnames the problem, that the foundation-model era has changed the economics of the problem rather than solving it, and that the most effective remedies are architectural rather than regularising.

Keywords: continual learning, catastrophic forgetting, complementary learning systems, memory replay, synaptic consolidation, loss of plasticity, neuro-inspired machine learning

1 Introduction

The observation that gives this field its name is over thirty years old. McCloskey and Cohen [1] trained a small network on one set of arithmetic problems, then trained the same network on a second set, and found that performance on the first collapsed almost to zero. They called it catastrophic interference. Ratcliff [2] reproduced it in recognition memory models, French [3] traced it to the overlap of distributed representations, and Goodfellow et al. [4] confirmed that neither dropout nor the switch to rectified units in deep networks made it go away. The phenomenon has survived every architectural revolution since.

What makes it more than a curiosity is that the systems we build it into are increasingly expected to operate in settings where the data does not arrive all at once. A clinical model meets a new imaging protocol. A recommender meets a product category that did not exist last quarter. A language model is asked about an event that postdates its pretraining corpus. The standard industrial answer — retrain from scratch on the union of old and new data — is correct, expensive, and increasingly impractical: it costs the full pretraining budget every time anything changes, and it presumes the old data is still available, which privacy law, licensing, and storage economics often prevent.

Much of the published comparison in this area is confounded: methods are compared at unequal compute [10], under scenarios definitions that are not held fixed [11], and with evaluation protocols that measure performance only at task boundaries [12]. A benchmark small enough to run five seeds of every method on a laptop CPU, with every comparison implemented from scratch, buys a kind of control that a larger benchmark makes expensive. It costs external validity, and we return to that cost in Section 6.8.

1.1 Contributions

(1) A unified formalisation of the continual learning problem (Section 2) that separates task, context, and explicit-memory variables (Section 2), followed by an analysis of forgetting as gradient interference and representational drift in a shared parameter space (Section 3).

(2) A structured mapping from six families of biological memory mechanisms to the algorithmic constructs they motivated, distinguishing mechanisms that genuinely constrain learning from those that store knowledge only as post-hoc framing (Section 4, Table 2).

(3) A taxonomy of algorithmic families with the mathematical stated in full and every symbol defined, including the exact objectives of EWC, SI, MAS, LwF, ER, A-GEM, and gradient-projection methods (Section 5).

(4) A controlled empirical study (Section 6) reporting: the scenario-dependence of method ranking; an arithmetic decomposition of class-incremental failure into within-context and context-inference terms; the replay-budget scaling curve; per-context measurement of the stability gap; a matched-budget comparison of interleaved versus offline ("sleep") replay; and a 40-task measurement of plasticity loss.

(5) An assessment of what the foundation-model era changes about the problem, and a research agenda that identifies where we believe current effort is misallocated (Sections 7–8).

All code, random seeds, and result files are self-contained; the full suite runs in well under an hour on a single CPU core.

2 Problem setting and notation

2.1 The learning problem

Ordinary supervised learning we assume a single fixed distribution $\mathcal{D}$ over pairs (x, y) and seek parameters θ that minimise the expected loss $\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_\theta(x), y)]$, where f_θ is a network with parameters $\theta \in \mathbb{R}^P$ and ℓ is a per-example loss such as the cross-entropy. Continual learning removes the assumption that $\mathcal{D}$ is fixed.

The learner instead receives a sequence of T *contexts* (we use "context" and "task" interchangeably, following [11], where context t supplies a dataset $\mathcal{D}_t = \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^{n_t}$ drawn from its own distribution. The contexts arrive one at a time. While context t is being learned, the data of contexts $1, \dots, t-1$ is no longer available except for whatever the learner has chosen to store. The goal after the final context is to minimise the loss averaged over all contexts seen so far:

$$\theta^* = \arg\min_\theta \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell(f_\theta(x), y)]. \quad (1)$$

Reading Eq. (1) carefully is worthwhile, because the difficulty is entirely in what is hidden in it. The symbol $\arg\min_\theta$ means "the parameter setting that makes the following quantity as small as possible"; the sum $\sum_{t=1}^{T}$ means "add up the contexts from 1 to T"; $\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\dots]$ is the average of the bracketed quantity over examples drawn from context t's distribution; and $\ell(f_\theta(x), y)$ is the penalty the network f_θ pays when it gives input x the label y but the correct answer was y. The objective is a sum over all contexts, but at any moment the learner can only compute the gradient of *one* term of that sum. Continual learning is therefore not a different objective from multi-task learning. It is the same objective under an information constraint about which terms are currently measurable.

2.2 Three scenarios, and why the distinction matters

Van de Ven and Tolias' continual learning" hides three problems of very different difficulty. The distinction is due to van de Ven et al. [11], van de Ven and Tolias [13] and turns entirely on what the learner is told at test time and what it is asked to discriminate between.

Task-incremental (Task-IL). The context identity is provided at test time, usually as a one-hot vector $c \in \{0,1\}^T$ with $\sum_t c_t = 1$. The network learns a separate output head for each context, and the task-ID input selects which head to use. Because the problem "which of these k classes?" is always a k-way choice, chance is $1/k$ regardless of how many contexts have been seen, and forgetting is entirely within-head. This is the easiest scenario by a wide margin.

Domain-incremental (Domain-IL). The label space is shared and constant, but the input distribution shifts. Recognising all ten digits under successively different sensor characteristics is domain-IL. The learner is not told which domain it is in and does not need to know.

Class-incremental (Class-IL). New classes appear over time, the label space grows, and at test time the learner must discriminate among *all* classes it has ever seen without being told the context. This is the hardest and the most realistic of the three, and it is the setting in which the regularisation family fails outright, for reasons we make quantitative in Section 6.5.

Much of the apparent disagreement in the literature dissolves once these are kept apart. A method reported as "solving" continual learning under Task-IL may be at chance under Class-IL, and vice versa [11]. We therefore evaluate every method in all three settings.

2.3 Metrics

Let $a_{t,i}$ denote test accuracy on context i measured immediately after the learner finishes training on context t, so $a_{t,i}$ is defined for $i \le t$. The standard summary statistics [15–17] are:

$$\bar{A}_T = \frac{1}{T}\sum_{i=1}^{T} a_{T,i}, \quad (2)$$

$$F = \frac{1}{T-1}\sum_{i=1}^{T-1} \left(a_{i,i} - a_{T,i}\right), \quad (3)$$

$$\mathrm{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1} \left(a_{T,i} - a_{i,i}\right), \quad (4)$$

$$\mathrm{LA} = \frac{1}{T}\sum_{i=1}^{T} a_{i,i}. \quad (5)$$

The first term is the ordinary mean over the current context's data. In the second, k indexes individual parameters, θ_k is the current value of parameter k, $\theta^*_{t-1,k}$ is its value when the previous context finished, $\Omega_k \ge 0$ is an estimate of how much that parameter matters for past contexts, and $\lambda > 0$ is a scalar controlling the strength of the whole penalty. Setting $\lambda = 0$ for all k recovers plain L_2 anchoring to the previous solution.

The families differ only in how Ω_k is computed.

Elastic weight consolidation [7]. EWC sets $\Omega_k = F_k$, the k-th diagonal entry of the Fisher information matrix,

$$F_k = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial \log p_\theta(y_i \mid x_i)}{\partial \theta_k}\right)^2, \quad (9)$$

evaluated at θ^*_{t-1}, on that context's data. Reading this: for each of n training examples we compute the derivative of the log probability the model assigns to the correct label with respect to parameter k, square it, and average. A parameter whose perturbation almost nothing changes the prediction gets a large F_k and is held in place. The Fisher approximates the Hessian of the loss there at a minimum, which connects this algorithm, showing that gating a task does not increase the memory load of units as it would if contexts were stored explicitly in the recurrent direction satisfying it.

Synaptic intelligence [56]. SI estimates importance online, during training, rather than at the end. It accumulates a path integral for each parameter,

$$\omega_k = \sum_{\text{time}} -g_k(\tau)\,\Delta\theta_k(\tau), \qquad \Omega_k = \frac{\omega_k}{(\Delta_k)^2 + \xi} \quad (10)$$

where $g_k(\tau)$ is the gradient of the loss with respect to parameter k at step τ, $\Delta\theta_k(\tau)$ is the update actually applied at that step, Δ_k is the total displacement of parameter k over the whole context, and ξ is a small constant preventing division by zero. The numerator credits a parameter for how much work it did; the denominator penalises it for how much total movement gives loss reduction per unit of displacement. SI's appeal is that it is a genuinely online, biologically plausible quantity, computable from information a synapse could plausibly have.

Memory aware synapses [64]. MAS uses the sensitivity of the network's own output, $\Omega_k = \frac{1}{N}\sum_{i=1}^{N} \|\partial\, l_2(f_\theta(x_i))/\partial\theta_k\|$, which has the practical advantage of using unlabelled data and therefore working on unlabelled deployment data.

Functional regularisation. A different choice is to constrain the function rather than the weights. Learning without Forgetting [65] adds a distillation term, i.e. it asks the network to output on current-context inputs close to the old model's outputs recorded before training, $\mathcal{L}_{\mathrm{LwF}} = \mathcal{L}_t + \lambda\,\mathrm{KL}(p_{\theta^*_{t-1}} \,\|\, p_\theta)$, where KL is the Kullback–Leibler divergence measuring how much one predictive distribution differs from another. This family needs no stored data and no Fisher, but it presumes that the current inputs are informative about the old function, and Gaussian-process treatments [61, 66, 67] put this on a Bayesian footing, where the penalty's dependence on what happens *afterwards*, whereas essentially every importance-based method fixes the importance estimate at the end of the task and never revises it.

2.5 Replay: constrain the data

Replay attacks the non-stationarity itself by reconstructing an approximation of the past, which is i.i.d. once more. The generic objective is

$$\mathcal{L}^{\mathrm{rep}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell(f_\theta(x), y)] + \alpha\,\mathbb{E}_{(x,y)\sim\mathcal{M}}[\ell(f_\theta(x), y)], \quad (11)$$

where $\mathcal{M}$ is a memory of examples and α balances the two terms. Variants differ in what $\mathcal{M}$ contains and how it is filled.

- **Experience replay** [53, 68] stores raw examples, typically by reservoir sampling so that the buffer stays a uniform sample of the stream. Simple, surprisingly hard to beat [11].
- **Constrained replay,** GEM [15] and A-GEM [69] use $\mathcal{M}$ less as an extra loss but as an inequality constraint, projecting the current gradient g so that it does not increase the memory loss, $\langle g, g_{\mathcal{M}}\rangle \ge 0$. A-GEM needs only a single average over the recent direction satisfying it.
- **Pseudo-rehearsal** [70] replaces the network's own past logits alongside the labels and matches them, which combines rehearsal with self-distillation.
- **Generative replay** [8] stores no data at all, training a generative model alongside the classifier and sampling pseudo-examples from it. This is the closest analogue of hippocampal replay and it sidesteps privacy and storage constraints, at the price of needing a generator good enough that its samples are useful — a condition that fails on complex inputs.
- **Latent replay** [14, 70, 71] replays internal representations rather than inputs. van de Ven et al.'s brain-inspired replay generates hidden activity through the network's own context-modulated feedback connections, which is both more scalable and closer to the biology, since the hippocampus does not store sensory retinal images.

- **Selection.** MIR [55] retrieves the stored examples whose loss would increase most under the impending update — maximally interfered-with retrieval — rather than ones merely uniformly, an algorithmic version of the reward- and novelty-modulated replay in Section 4.2 [72, 73].

2.3 Parameter isolation: constrain the architecture

If interference comes from sharing, allocate different parameters to different contexts. Progressive networks [72] add a new column per context with lateral connections to previous old columns, guaranteeing zero forgetting at the cost of linear parameter growth. PackNet [58] iteratively prunes and freezes, packing multiple contexts into one fixed network. HAT [59] learns per-context attention masks over units with a sparsity penalty. Supermasks in superposition [73] keep a fixed random backbone and learn only a binary mask per context, while hypernetworks [74] *generate* the network's own past logits alongside the labels and matches them, which combines rehearsal with self-distillation