Continual Learning Algorithms

Anonymous authors
Paper under double-blind review

Abstract

Continual learning studies how a model can keep acquiring knowledge from a non-stationary stream of data without retraining from scratch. The field has produced a large and fast-growing catalogue of algorithms, yet it is often unclear what they share, what problem they solve, and how much of it has been solved—a confusion that the new vocabulary introduced by recent language-model methods has only deepened. This review argues that most continual learning algorithms pursue the same objective: approximately minimizing the *historical risk*, the average loss over all data observed so far, under a limited computational budget. They differ mainly in how they approximate the past—through a *surrogate* term added to the objective, or a *trust region* that constrains the update—and we use this distinction to organize the literature, classical and recent alike. We complement this taxonomy with an accounting of computational cost that separates the cost of each update from the cost of consolidating the algorithm's memory. Under this accounting, no surveyed method keeps its excess risk bounded while paying a cost that grows *sublinearly* with the history: a closer approximation of the past is always paid for with more computation, memory, or model capacity. Reading recent foundation-model adaptation methods through the same lens shows that they largely reinstantiate classical mechanisms—replay, distillation, parameter isolation, low-rank constraints—under new names: the toolkit has been modernized, but what is computationally achievable has, so far, not changed. We distill from this analysis the open problems whose solution, we argue, would constitute real progress in the field.

1 Introduction

From its inception, the vision of artificial intelligence has been closely tied to *adaptability—the ability to acquire new skills over time*—as a fundamental component of intelligence (Legg & Hutter, 2007). According to these foundational definitions, intelligence is not the ability to solve difficult tasks, but the ability to continue acquiring knowledge and reasoning capabilities as the world changes. Yet in the classical machine learning paradigm knowledge is acquired during a training phase, after which the model is deployed as a fixed artifact. Under this paradigm, adaptation requires either retraining from scratch on the accumulated data, or, more recently, relying on the emergent capabilities of large language models through In-Context Learning (ICL) (Brown et al., 2020; Radford et al., 2021). The success of ICL has made modern machine learning useful in dynamic domains such as coding assistance and conversational agents. At the same time, it also reopens a basic question: how can we build systems with a greater capacity for ongoing, open-ended adaptation?

ICL enables adaptation by operating within the activation space of a frozen model. This mechanism is powerful, but inherently constrained by the finite size of the context window. Moreover, as the provided context grows, the computational cost of inference grows with it. Retrieval systems can improve the use of limited context, but they do not change the fact that relevant information must be supplied again at inference time. The field of *continual learning* (CL) takes a more radical position: learning itself should be an ongoing process. Continual learning algorithms aim to incrementally update a network's parameters as new data arrives over time. By embedding knowledge directly into the weight space, CL seeks to enable lifelong online knowledge acquisition without systematically increasing inference complexity.

The shift toward never-ending adaptation forces a fundamental re-evaluation of how we measure model performance. In traditional machine learning, generalization is evaluated on held-out data from a static distribution. When an agent operates in a non-stationary environment, however, there is no single fixed distribution that fully defines success. Historically, the continual learning literature has navigated this ambiguity by prioritizing historical retention: a model is expected to preserve performance on previous experiences, thereby avoiding *catastrophic forgetting* (McCloskey & Cohen, 1989; Ratcliff, 1990; French, 1999; De Lange et al., 2022; van de Ven et al., 2022). In optimization terms, this usually amounts to minimizing the average loss over all data observed so far, which we will call the *historical risk*.

This objective is natural, but it is not inevitable. The *Big World Hypothesis* (Javed & Sutton, 2024) challenges the assumption that the environment is small enough to be retained in full: open-ended environments will expose the system to more information than a finite-capacity model can possibly preserve. Under strict capacity constraints, the learner must decide what to preserve, what to compress, and what to forget (Kumar et al., 2025). This raises a question that is logically prior to the design of any algorithm: *what* is worth remembering. It is distinct from, and largely independent of, the question the algorithms in this review actually answer—*how* to retain whatever has been selected. We keep the two apart throughout, and develop the distinction in section 2.1.

Even if the world were compressible into a single model, continuous adaptation would still be bounded by computational cost. The most reliable way to integrate new knowledge is to retrain the model from scratch on all accumulated data. This procedure achieves adaptation by construction, but at each moment it requires reprocessing a history that grows with time. As the lifetime of the system increases, so does the cost of each update. At the scale of modern foundation models, this brute-force strategy quickly becomes economically and practically untenable. The long-term goal of continual learning is to provide a cheaper alternative to retraining: an algorithm that can integrate new knowledge while paying a cost that grows *sublinearly* with the length of the learning history.

As we will argue in this paper, existing continual learning algorithms have not fully achieved this goal. They either preserve performance by relying on computation, memory, or model capacity that grows with the history, or they use cheaper approximations whose performance degrades over long horizons. This is not a failure of individual methods as much as a sign that the field has often lacked a precise definition of its goals. For this reason, in this paper we adopt a theoretical perspective to describe continual learning algorithms: it allows us to abstract away implementation details and identify the axes along which real progress has been made. A theoretical understanding is also, we will argue, a precondition for practice: systems that keep learning after deployment can only be trusted with guarantees on what they retain, what they optimize, and when they fail (section 5.1).

Core thesis and contributions. This review is organized around four claims, which are also its main contributions; the discussion in section 5 returns to each of them after the review.

- *Most existing continual learning algorithms can be understood as approximate minimization of the historical risk—the average loss over the data observed so far—under a limited computational budget.* We formalize this objective in section 2.1, and our in-depth review in section 4 substantiates the claim.

- These algorithms fall into two broad classes with mirrored failure modes, distinguished by whether the past is *represented* through a surrogate objective or *protected* by a trust region: *surrogate-objective methods* optimize a cheaper surrogate for the historical risk, while *trust-region methods* restrict adaptation to regions where historical performance is preserved (section 3.4).

- Comparing methods fairly requires an explicit model of computational cost. We propose an accounting that separates the per-step cost from the per-interaction *consolidation* cost, making explicit when an algorithm's cost grows with the length of the history (section 3.3). It exposes the trade-off frontier that, we argue, is the real frontier of progress in the field: closer approximations of the historical risk cost more computation, more memory, or less reusable model capacity.

- *Existing methods provide no general mechanism for maintaining a bounded excess historical risk at a per-interaction cost sublinear in the history*; in this precise sense the field has not yet solved its core problem, and progress is more limited than the volume of published methods suggests. In particular, the recent wave of continual-adaptation methods for large foundation models reintroduces the same mechanisms under new names—replay, distillation, parameter isolation, low-rank constraints—without, so far, a fundamental change in what is computationally achievable. We take this as evidence that the field needs the kind of sharper theoretical organization this review attempts to provide.

2 The continual learning setup

We now introduce the minimal formal setup used throughout the review. The goal is not to prescribe a single continual learning benchmark, but to define notation broad enough to cover online, task-based, supervised, generative, and reinforcement-learning settings. This generality is important because the same algorithmic ideas often reappear across settings under different names. Table 1 collects the symbols used throughout for reference.

Table 1: Summary of notation.

Symbol	**Meaning**
o_t, $o_{i:j}$	observation at step t; sequence of observations over $[i, j]$
$f \in \mathcal{F}$, $\theta \in \Theta$	model and its parameters
$\ell(\theta, o)$, $\ell_t(\theta)$	loss on observation o; shorthand for $\ell(\theta, o_t)$
$\mathcal{R}_t(\theta) = \ell(\theta, o_{1:t})$	historical risk (average cumulative loss), eq. (1)
θ_t	parameters the algorithm *produces* at time t
$\theta_t^\star \in \arg\min_\theta \mathcal{R}_t(\theta)$	ideal historical-risk minimizer
$\mathcal{B}_t$	replay buffer (a stored subset of $o_{1:t}$)
s_t	algorithm memory state (buffer, statistics, masks, ...)
g_{t-1}, TR_{t-1}	surrogate for the past loss; trust region on updates
$\epsilon_t = \mathcal{R}_t(\theta_t) - \mathcal{R}_t(\theta_t^\star)$	excess historical risk, eq. (5)
c_t^{step}, c_t^{cons}, C_t	step cost, consolidation cost, total per-interaction cost
K_t, ν	inner-loop steps at time t; consolidation frequency, eq. (4)
$m_\epsilon(t)$	minimal information to retain for ϵ-accuracy (section 3.3)

Data. Data is received *sequentially* in time, and the learner must update its model continuously as new information becomes available. We denote by o_t the data or *observation* received at time-step t, and by $o_{i:j} = o_i, \dots, o_j$ the sequence of data observed in the interval $[i, j]$. The notation is intentionally generic: o_t may denote a single sample, a mini-batch, or an entire dataset, depending on the context. Each sample may itself be a tuple, such as input–output pairs in supervised learning or state–action–reward tuples in reinforcement learning. The time index t may index seconds, months, years, or semantically organized units such as *tasks* or skills (e.g., a new language). With a slight abuse of terminology, we may refer to o_t interchangeably as a *task*. Several of the algorithms we review retain an explicit subset of past observations; we denote such a *buffer* (or episodic memory) by $\mathcal{B}_t \subseteq o_{1:t}$, and use it as a running example of stored past information.

Model and learning algorithm. The *model* is a function $f \in \mathcal{F}$ parameterized by $\theta \in \Theta$. Following standard conventions, we focus on neural networks, where θ encompasses the trainable parameters of the model, including weights, biases, and normalization statistics. The model maps an input x to an output $f(\theta, x)$; both inputs and outputs are left generic, since their form depends on the application. The *learning algorithm* produces a sequence of models $(f_1, f_2, \dots)$ over time, updating the parameters, and possibly additional state such as memories or statistics, as new observations arrive.

Loss function. A *loss function* $\ell(\theta, o) \in \mathbb{R}^+$ is a non-negative scalar quantifying the model error on an observation o, where lower values correspond to better performance. When o is a set of observations (e.g., a batch or dataset), $\ell(\theta, o)$ denotes the average loss over the set. For brevity, we may denote $\ell(\theta, o_t)$ as $\ell_t(\theta)$. In supervised learning, for example, an observation may be a pair $o_t = (x_t, y_t)$, with x_t the model input and y_t the target, and the loss may be written as $\ell(f_\theta(x_t), y_t)$.

2.1 Historical risk

As mentioned before, the definition of a metric for continual learning performance is still a debated topic. For any given data sequence, learning algorithm, and loss function, there are many desirable characteristics to be encoded. Some examples, which have emerged in the continual learning literature, include: *backward transfer* or *forgetting*—the ability of the model to retain and improve previous skills; *forward transfer*—the ability of the model to exploit already acquired skills to acquire new ones; *plasticity*—the ability of the model to acquire new skills at all; and *real-time performance*—the ability of the model to deal with the current task at hand. The choice of metric is not merely a matter of evaluation as it implicitly determines which learning behaviors are desirable, and therefore shapes algorithmic design. Each of these choices represents a compromise: focusing on forward transfer might come at the cost of backward transfer, or vice-versa. The exact relation between these principles is still an open theoretical question in the field.

The most common choice of objective for continual learning algorithms in the literature is the average loss over the data observed so far. We call this quantity the *historical risk*. In task-based settings, the same objective is often called the *multi-task loss*. Given observations $o_{1:t}$, the historical risk is

$$\mathcal{R}_t(\theta) = \ell(\theta, o_{1:t}) = \tfrac{1}{t}\left(\ell(\theta, o_1) + \cdots + \ell(\theta, o_t)\right). \tag{1}$$

The normalization by $1/t$ keeps the scale of the objective comparable over time; it does not change the minimizers relative to the unnormalized cumulative loss. For compactness, we will sometimes write $\ell_{1:t}(\theta)$ or $\mathcal{R}_t(\theta)$. Thus, in what follows, *historical risk* denotes the conceptual objective, *average cumulative loss* denotes its mathematical form, *multi-task loss* denotes the same objective in task-based settings, and *cumulative loss* denotes the unnormalized variant. The historical risk captures how well the model performs across its entire learning history, and is closely related to *forgetting*. If θ_t denotes the model parameters at time t, forgetting of an observation o_i can be measured in terms of the loss difference:

$$\delta\ell(\theta_t, o_i) = \ell(\theta_t, o_i) - \min_{k \in [1, t-1]} \ell(\theta_k, o_i). \tag{2}$$

Under this definition, *positive forgetting* signifies a drop in performance relative to the best previous result, while *negative forgetting* indicates improvement or transfer. While alternative formulations exist depending on the specific task and metric (e.g., accuracy versus reward), they share this conceptual lineage.[1]

Forgetting is only one of the two terms that make up the historical risk. The other is *plasticity*: the model must fit an observation when it is first encountered for there to be any performance to retain later. A task the model never learned contributes no forgetting, yet still raises the historical risk. A low historical risk therefore requires both—learning each task and remembering it afterwards.

What to remember versus how to remember. Whether the historical risk is the right objective at all is an open and active question, pressed by the Big World Hypothesis (Javed & Sutton, 2024): a finite agent in a sufficiently large world cannot retain everything, and must choose what to keep. Our thesis is that this choice—*what* to keep—is largely orthogonal to the question the algorithms in this review address: *how* to keep it. These algorithms build a cumulative and modular memory of the past, and with minor modifications the same mechanisms apply to whatever one decides to retain, as we show with examples in section 5.2. They therefore remain useful even if the field moves away from the historical-risk objective.

[1] The data on which historical risk and forgetting are measured is itself a modeling choice. We write losses on the observed sequence for simplicity, but in practice forgetting is often evaluated on *test data*—unobserved samples from the same task or time-dependent distribution—and other choices are possible: related skills, a fixed application-specific skill set, or features intended to support future transfer (Zhang et al., 2022a). These choices matter for algorithm design: if the goal is transferring to a specific population or environment, that should be, implicitly or explicitly, the learning objective.

Whether anything is gained by addressing the two questions *together*—choosing what to keep and how to keep it jointly—is itself an open and potentially fruitful question.

Once historical risk is fixed as the reference objective, continual learning algorithms can be compared by how they approximate its minimizer over time. This is the perspective developed next.

3 A unifying formulation of continual learning algorithms

3.1 The ideal historical-risk solution

At any time t, the algorithm should produce model parameters that minimize[2] the historical risk:

$$\theta_t^\star \in \arg\min_\theta \; \mathcal{R}_t(\theta). \tag{3}$$

Of course, computing $\theta_t^\star$ exactly is generally impossible in a continual setting: it would require access to all past observations and enough computation to solve the historical-risk minimization problem after each interaction with the world. We therefore use $\theta_t^\star$ as an ideal reference point, and write θ_t (without a star) for the parameters the algorithm *actually produces* at time t. Continual learning algorithms differ in how closely θ_t approximates $\theta_t^\star$ while limiting access to past data, computation, memory, or model capacity.

There are many ways to search for a solution of the objective in eq. (1). This formulation covers methods that adapt a model through sequential optimization, which remain the dominant and most versatile class of continual learning approaches. It does not directly describe all possible adaptation mechanisms. For example, model merging, ensembling, retrieval-augmented systems, and some external-memory architectures may combine, query, or compose models without following a single parameter trajectory. We return to these cases later.

3.2 A generic CL algorithm

A continual learning algorithm couples *two processes that run at the same time*: an *environment process* that produces observations, and a *learning process* that spends computation to update the model. The formulation below is deliberately general. Because an observation o_t may be a single sample, a mini-batch, or an entire dataset, and the learner may take any number of update steps after it—including none, holding the model fixed—the same template covers strict online learning, periodic updating once every M observations, and task-based continual learning.

The learning process approaches the reference solution $\theta_t^\star$ incrementally, through a sequence of model updates. We capture it with a generic STEP operator. Let BASE denote a one-step optimization map that returns a proposed *update direction*: given an objective J and current parameters θ, $\mathrm{BASE}(J, \theta)$ is a step—stochastic-gradient, Adam, or natural-gradient—so that the unconstrained next iterate is $\theta + \mathrm{BASE}(J, \theta)$; running BASE repeatedly represents a longer optimization phase. The STEP operator wraps BASE with whatever information about the past the algorithm maintains, and returns the update actually applied to the model. We give the two concrete forms of STEP in section 3.4.

Algorithm 1 makes the shared skeleton explicit.

[2]The use of membership $\in$ rather than equality emphasizes that multiple solutions typically exist. From the historical-risk perspective, these solutions are equivalent; yet, they may differ substantially in other respects—such as their bias toward particular tasks or their generalization to unseen data.

Algorithm 1: Generic continual learning algorithm

Initialize parameters θ_0 and memory state s_0;
for $t = 1, \ldots, T$ **do** // one interaction
  Observe o_t; set $\theta \leftarrow \theta_{t-1}$;
  while *not finished* **do** // model update
    $\theta \leftarrow \theta + \mathrm{STEP}(\theta;\, s_{t-1})$;
  $\theta_t \leftarrow \theta$; update memory state s_t ; // consolidation
return θ_T, s_T

The two processes appear as the two loops of algorithm 1: the outer loop advances the environment one observation at a time, and the inner loop performs the model update that follows. The state s_t is the algorithm's *memory*—a replay buffer, a generator, sufficient statistics, a Fisher or Hessian approximation, a subspace basis, masks, router parameters, or prototypes—and is what STEP consults about the past.

3.3 Counting the cost

Historically, continual learning research has modeled operational constraints as limits on *data usage*, simulating privacy or storage restrictions. However, the more fundamental and universal barrier is *compute*: even given infinite storage and perfect access to past data, a scalable system simply cannot afford the processing time or energy to reprocess its entire history at every time step (Kumar et al., 2025; Prabhu et al., 2023). The binding constraint is that the per-interaction compute the learner can spend is bounded, regardless of how much data it is in principle allowed to revisit. In this work, we adopt this more general view and characterize continual learning algorithms in terms of how their computational cost grows with time.

In order to properly account for the total number of computations, we explicitly distinguish between two sources of cost: the *step cost* and the *consolidation cost*. Throughout this discussion, we assume for simplicity that each observation o_t contains a comparable amount of data; equivalently, t can be interpreted as cumulative observed data measured in fixed-size units.

- **Step cost** c_t^{step}: the cost of one execution of STEP inside the inner optimization loop. This includes forward and backward passes, replay batches, regularizer evaluations, projections, and optimizer updates.
- **Consolidation cost** c_t^{cons}: the cost paid once after receiving observation o_t, outside the repeated inner-loop updates, to fold it into the memory state s_t. This includes updating a replay buffer, computing a Fisher matrix, estimating a subspace, training a generator, or storing prototypes.

An algorithm makes two choices that govern its cost: the number of inner-loop steps K_t it takes after an observation—possibly zero, leaving the model untouched—and the *consolidation frequency* $\nu \in (0, 1]$ at which it folds new information into its memory—once every $1/\nu$ observations, with $\nu = 1$ after every one. Here *consolidation* comprises all the work done once per processed observation, outside the inner loop: updating the memory state s_t and committing the new parameters θ_t. Observations that arrive between consolidations are simply accumulated into the next observation the learner processes—this accumulation is part of the meaning of o_t—and, while accumulating, the system keeps responding to the environment with its current parameters, without learning. The cost of processing observation o_t is then

$$C_t = K_t\, c_t^{\text{step}} + \nu\, c_t^{\text{cons}}, \tag{4}$$

and the lifetime cost after T observations is $\sum_{t=1}^{T} C_t$. Together with the freedom to let o_t be a single sample, a mini-batch, or an entire dataset, these two knobs let the template of section 3.2 describe the full range of settings: strict online learning (o_t a sample, small K_t, $\nu = 1$), periodic updating (small ν; between consolidations $K_t = 0$, so the model is genuinely fixed while data accumulates), and task-based continual learning (o_t a dataset, large K_t).

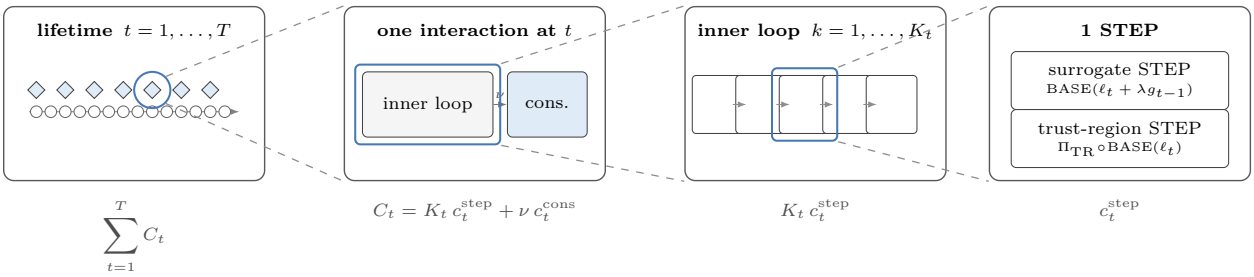

Figure 1: **Anatomy of the cost.** Progressive zoom-in on algorithm 1, with the cost contributed at each scale. From left to right: over the *lifetime*, each observation o_t (circle) triggers an update, while the memory is consolidated (diamond) once every $1/\nu$ observations, for a total cost $\sum_t C_t$; a single *interaction* is an inner loop followed—with frequency ν (dashed)—by a consolidation, at cost $C_t = K_t\, c_t^{\mathrm{step}} + \nu\, c_t^{\mathrm{cons}}$; the *inner loop* is K_t executions of STEP (cost $K_t\, c_t^{\mathrm{step}}$); and a single STEP is one of the two forms studied in this review—a surrogate-objective STEP (Definition 1) or a trust-region STEP (Definition 2)—both costing c_t^{step}.

Historically, many studies have evaluated the computational cost of CL methods based solely on step-cost efficiency, neglecting the consolidation overhead associated with maintaining expensive approximations or constraints. This oversight stems from the dominance of the task-based setting in the classic literature—or equivalently, of slow environment timescales—where each observation corresponds to a large dataset and the number of optimization steps is high, rendering the inner loop dominant. The balance changes in environments that demand fast acquisition of new knowledge—physical domains where the agent must adapt at the pace of the world around it. There, the update cannot be deferred: new information must be folded into the model almost as soon as it arrives ($\nu \to 1$), so the consolidation overhead becomes as relevant as the step cost. We return to the problem of scheduling updates against the environment's timescale in section 5.2. More generally, the lowest-cost algorithm is the one that achieves an optimal trade-off between step and consolidation cost—balancing immediate responsiveness against long-term stability.

Sublinear adaptation. The retraining baseline recomputes a solution to the historical risk using all data observed so far. If the amount of data per interaction is constant, then the cost of each retraining step grows at least linearly with t, since the optimizer must process a history of size t. We say that a continual learning algorithm achieves *sublinear adaptation* if

$$C_t = o(t)$$

while maintaining an acceptable approximation to the historical-risk optimum. For example, one may require

$$\mathcal{R}_t(\theta_t) - \mathcal{R}_t(\theta_t^\star) \leq \epsilon$$

for a tolerance ϵ. This criterion separates the statistical question of how well an algorithm approximates the historical risk from the computational question of how the cost of that approximation scales with the length of the learning history.

When is sublinear adaptation possible? Whether sublinear adaptation is achievable is a property of the data stream before it is a property of the algorithm. For a fixed tolerance ϵ, every stream carries a minimum amount of information that any learner must retain for the historical risk to stay within ϵ of the optimum; we denote its size by $m_\epsilon(t)$, as a function of the number of observations t. This informational floor translates into a computational one: a state of size $m_\epsilon(t)$ must at least be consulted at each update. At the other extreme, retaining and reprocessing the full history always suffices, at cost linear in t. The cost of an ϵ-accurate learner thus falls between a lower bound set by the stream and a trivial upper bound, $m_\epsilon(t) \lesssim C_t \lesssim t$, and the room between the two is governed by the growth of $m_\epsilon(t)$—the compressibility of the stream. If $m_\epsilon(t)$ is of order t—the *Big World* regime of Javed & Sutton (2024)— a bounded-computation learner must give up part of the past. The argument also runs in reverse: plain fine-tuning pays $O(1)$ per interaction by design, so it can maintain a bounded excess risk only when $m_\epsilon(t)$ is itself bounded—that is,

in effectively finite environments. This argument is informal, not a theorem: making it precise is one of the central open questions of the field (section 5.2).

3.4 Approximating the past: surrogate objectives and trust regions

The impossibility of computing $\theta_t^\star$ cheaply forces every continual learning algorithm to approximate it. We measure the quality of an approximation directly, by the *excess historical risk* of the solution it produces,

$$\epsilon_t \;:=\; \mathcal{R}_t(\theta_t) - \mathcal{R}_t(\theta_t^\star), \tag{5}$$

the same quantity used in the sublinear-adaptation criterion above. A smaller ϵ_t means a more faithful approximation. *The central tension of the field is that driving ϵ_t down generally costs more computation, memory, or model capacity.*

We propose an optimization-first view of the literature, which organizes the existing approximations around a familiar duality. At each step the learner minimizes the current loss while trying not to undo the past, and the literature enforces the second requirement in two ways: in *surrogate-objective methods*, the past enters as a penalty term g_{t-1} added to the objective; in *trust-region methods*, it enters as a constraint TR_{t-1} on the admissible update. These are the penalty and constrained forms of the same program. The two are related—a penalty can often be tightened into a constraint, and a constraint relaxed into a penalty—but they are not interchangeable in practice, and charting when they correspond, and at what cost in ϵ and computation, is still open (section 5.2). Keeping the distinction is useful because the two forms carry different technical advantages: how tightly forgetting can be guaranteed, what object is placed in long-term memory and how it is consulted, and how each tends to fail. We develop these contrasts method by method in section 4.

Definition 1 (Surrogate-objective STEP)**.** *Let g_{t-1} be an approximation of the past loss $\ell(\cdot, o_{1:t-1})$. It is combined with the current loss into the surrogate objective*

$$J_t^{\mathrm{surr}} \;:=\; \ell_t + \lambda\, g_{t-1}, \qquad \lambda > 0,$$

where λ trades plasticity against retention. The surrogate objective step applies:

$$\mathrm{STEP}(\theta) = \mathrm{BASE}(J_t^{\mathrm{surr}}, \theta). \tag{6}$$

The surrogate is ϵ-accurate *if optimizing J_t^{surr} yields a solution whose excess historical risk (5) is at most ϵ.*

Notation. When writing g_t we refer to an approximation of $\ell(\cdot, o_{1:t})$; when writing $g(\cdot, o)$ we refer to an approximation of $\ell(\cdot, o)$.

Definition 2 (Trust-region STEP)**.** *Let θ_r be a reference point and Π_A the projection operator onto a set A. The trust region is* ϵ-valid *if historical performance degrades by at most ϵ inside it,*

$$\mathrm{TR}_{t-1}(\theta_r) \subseteq \{\theta : \mathcal{R}_{t-1}(\theta) \le \mathcal{R}_{t-1}(\theta_r) + \epsilon\},$$

the case $\epsilon = 0$ recovering the ideal region where historical risk never increases. The trust-region STEP *projects the unconstrained update onto the trust-region:*

$$\mathrm{STEP}(\theta) = \Pi^{\theta}_{\mathrm{TR}_{t-1}(\theta_r)}[\mathrm{BASE}(\ell_t, \theta)]\,. \tag{7}$$

The tolerance ϵ and the cost C_t are at odds: driving $\epsilon \to 0$ requires preserving more information about the past—more replay, finer curvature, larger subspaces—which raises c_t^{step}, c_t^{cons}, memory, or model size. Figure 2 sketches this trade-off conceptually. Only the two endpoints of the landscape can be placed with confidence—fine-tuning without any retention mechanism, and full retraining—and we deliberately leave the interior open: locating each mechanism on this plane requires an assumption-explicit accounting of its cost and excess risk, which we identify as an open task (section 5.2); the tables of section 4 are a first, coarse step in that direction.

The remainder of the paper reads existing algorithms through this lens, asking, for each, how it defines g_{t-1} or TR_{t-1}, what tolerance ϵ it achieves, and at what cost.

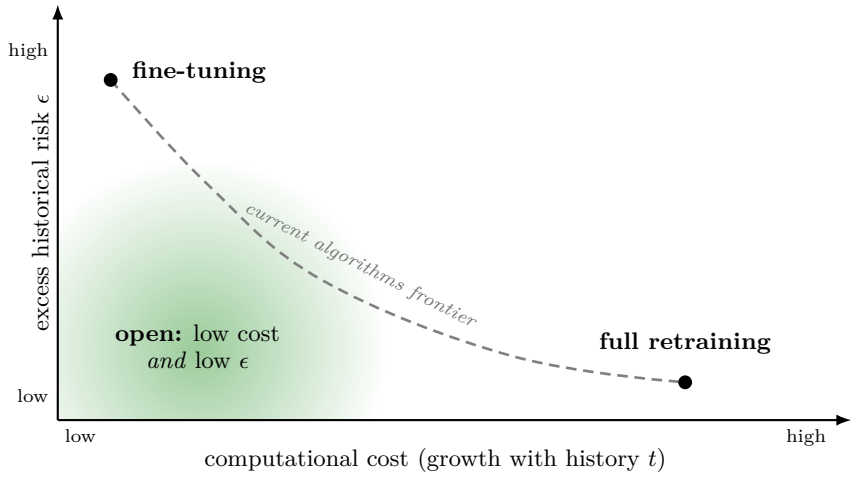

Figure 2: **Conceptual schematic.** The accuracy–cost trade-off that organizes the field, drawn for large t. Only the two extremes can be placed with confidence: plain fine-tuning, with no retention mechanism, is cheap but incurs a high excess risk ϵ; full retraining is faithful at a per-step cost of order t. Every mechanism reviewed in section 4 lies somewhere between these two anchors (dashed), but placing it rigorously requires an assumption-explicit account of its cost and excess risk that the field currently lacks—an open problem in itself (section 5.2). The highlighted lower-left region—low ϵ *at* low cost—is, to our knowledge, reached by no current mechanism. The axes are ordinal and show only scaling with the history t, model size held fixed.

4 Continual learning algorithms

In organizing the review we follow the common split used by existing surveys (De Lange et al., 2022; van de Ven et al., 2022), which our framework groups into the two categories identified above: surrogate-objective and trust-region methods. We connect existing continual learning methods to the framework introduced in algorithm 1, by specifying how each algorithm defines the STEP operator, and either the approximation g_{t-1} or the trust region TR_{t-1}. This organization is intentionally mechanism-based rather than name-based. Many recent adaptation methods for large pretrained models reintroduce the same computational choices under different terminology: what information from the past is retained, where it is stored, how expensive it is to consult, and which parameter directions are allowed to change. Focusing on the mechanism makes this continuity visible, even when the practical setting differs substantially. Table 2 previews this mapping.

4.1 Surrogate-objective methods

The continual learning algorithms discussed in this section all implement a surrogate-objective STEP operator, as defined in Definition 1. As we will see, they employ differing approximations of the past losses $g_{t-1} \approx \ell(\cdot, o_{1:t-1})$, resulting in distinct trade-offs between computational cost and performance.

4.1.1 Experience replay

Among the recognised families of continual learning algorithms, the oldest and most influential class is Experience Replay (ER). First introduced by Robins (1993; 1995), it was inspired by hippocampal replay mechanisms observed in biological memory consolidation (Hayes et al., 2021), from which it derives its name. Remarkably, despite its age, replay remains among the strongest baselines even today (Buzzega et al., 2021; 2020), and has also been applied to large-scale applications, including LLM pre-training and sequential fine-tuning (Ibrahim et al., 2024; Abbes et al., 2025; Hickok, 2025). The core idea is simple: instead of compressing the whole past into a statistic or constraint, the algorithm keeps some material trace of past observations and trains on it again. This trace may be raw data, generated data, latent features, model outputs, trajectories, or a large data mixture. Below, we review how replay-based methods vary along several key design axes. In

Table 2: The main algorithm families grouped by the generic STEP operator. Surrogate-objective families (blue) add a term g_{t-1} approximating the past loss; trust-region families (pink) restrict the update to a region preserving past performance. The memory state s_t is what each method stores of the past; the mechanism column names how that memory is turned into a STEP. The section column points to the discussion of each family.

Type	Family	Memory state s_t	Mechanism	§
surrogate	Experience replay	data buffer $\mathcal{B}$	replay loss $g_{t-1} = \ell(\cdot, \mathcal{B})$	4.1.1
	Generative replay	generator $\mathcal{G}$	generated-sample loss $g_{t-1} = \mathbb{E}_{\mathcal{G}}[\ell]$	4.1.1
	Regularization (EWC)	anchor + curvature Q	quadratic penalty g_{t-1}	4.1.2
	Variational (VCL)	posterior (μ, Σ)	KL-to-prior penalty g_{t-1}	4.1.2
	Functional / GP	inducing points	function-space penalty g_{t-1}	4.1.2
trust-region	GEM / A-GEM	data buffer $\mathcal{B}$	gradient-constraint projection	4.2.1
	OGD / GPM	gradient / repr. subspace	gradient projection ($\perp$ subspace)	4.2.1
	Orthogonal LoRA	low-rank subspaces	low-rank orthogonal construction	4.2.1
	Masking / expansion / prompts	masks / new modules	parameter freezing	4.2.2
	Mixture of Experts	experts + learned router	routing + expert freezing	4.2.2

the simplest form, ER builds a buffer $\mathcal{B}_{t-1} \subset o_{1:t-1}$ of B past observations (throughout, $B = |\mathcal{B}_{t-1}|$ denotes the buffer size), and approximates the loss over the full past data as

$$g_{t-1}(\theta) = \ell(\theta, \mathcal{B}_{t-1}).$$

The consolidation cost can be low when buffer maintenance is simple, but the step cost grows with the amount of replay used in each update. Experience replay therefore turns the problem of approximating the historical risk into a concrete resource allocation problem: which past observations should be stored, how often should they be revisited, and how much computation should be spent on them?

Generative ER. As an alternative to sampling from a fixed buffer, one may sample from a generative model of the past data: $g_{t-1}(\theta) = \mathbb{E}_{o\sim\mathcal{G}}[\ell(\theta, o)]$. Introduced by Shin et al. (2017), the first implementation employed GANs to synthesize pseudo-samples of previous tasks, while recent work explores diffusion-based generators (Gao & Liu, 2023; Kim et al., 2025). However, generative replay creates two main challenges of its own: (1) training the generator is costly, and (2) when trained continually, the generator itself suffers from forgetting. The cost of training and maintaining the generator is not incidental: it shifts part of the historical-risk approximation problem from the learner to the memory model. To address this issue, Liu et al. (2020) proposed generating features rather than raw inputs—but this still requires a CL mechanism to stabilize the feature extractor. With the rise of powerful generative models, the CL community has increasingly focused on the continual training of generators themselves (we refer the reader to Guo et al., 2025, for a recent survey).

Minimal buffers. Larger buffers naturally provide a better approximation of the past loss, but they incur higher computational complexity. Consequently, substantial research has focused on identifying more informative buffers, which can be formulated as a coreset selection problem: find the smallest buffer whose replay keeps the approximation within tolerance,

$$\min_{S \subseteq o_{1:t-1}} |S| \quad \text{s.t.} \quad \epsilon(S) \leq \epsilon,$$

where $\epsilon(S)$ is the excess historical risk (5) obtained when the surrogate is $g_{t-1} = \ell(\cdot, S)$. In other words, the goal is the minimum-size buffer that still achieves a target approximation quality. For complex models an exact solution is intractable, but several approximations have been proposed. When the optimizer BASE is gradient-based, this translates to choosing samples whose gradients best approximate those of the full dataset. Gradient-based coreset selection has been widely studied in continual learning (Aljundi et al.,

2019b; Borsos et al., 2020; Yoon et al., 2022; Tiwari et al., 2022), though often at high (consolidation) computational cost. Other methods employ meta-learning (Riemer et al., 2019), influence estimation (Sun et al., 2022), or bilevel optimization (Aljundi et al., 2019a; Chaudhry et al., 2021), which further increase computational complexity.

Is the extra computation spent in selecting a better buffer worth it? In other words, with the same computational budget, would a more expensive buffer selection method outperform a larger uniform buffer? Empirical results from Prabhu et al. (2023) show that, when fixing the total compute budget, "*costly sampling strategies perform significantly worse than simple uniform replay.*" This is also the lesson of several more recent replay and data-mixture studies (Prabhu et al., 2020; Zhang et al., 2022b; Ibrahim et al., 2024). Methods such as GDumb (Prabhu et al., 2020) and Repeated Augmented Rehearsal (Zhang et al., 2022b) show how much performance can be recovered by better using the same elementary mechanism: store data, revisit it, augment it, or mix it with new data at the right rate. In the online setting, refinements to *how* stored and incoming samples are combined—retrieving maximally-interfered samples (Aljundi et al., 2019a) or decoupling the loss to curb abrupt representation drift, as in ER-ACE (Caccia et al., 2022)—improve stability at negligible additional cost. In large language model continual pretraining, data replay and data mixing play the same role, except that the "buffer" is often a large retained corpus or a mixture schedule rather than a small episodic memory (Ibrahim et al., 2024; Abbes et al., 2025). In reinforcement learning, the same primitive appears as experience replay and prioritized replay, where the interaction between replay selection, policy-induced data distributions, and non-stationarity becomes even more pronounced (Schaul et al., 2016; Zha et al., 2019; Sun et al., 2020; Neves et al., 2024).

Knowledge Distillation. Surprisingly, another approach to reduce the buffer size B without sacrificing performance is to use Knowledge Distillation (introduced by Hinton et al., 2015), widely applied in CL (e.g., Rebuffi et al., 2017; Hou et al., 2019; Simon et al., 2021; Iscen et al., 2020). A particularly interesting case is *Dark Experience Replay* (DER) (Buzzega et al., 2020; Boschini et al., 2023), which stores, instead of the ground-truth label y_i, the output $\tilde{y}_i = f(\theta_i, x_i)$ that the model produced when it first saw x_i, and replays against this past prediction rather than the true target. This simple modification yields substantial gains—especially for small buffers—without increasing computational cost. Interestingly, earlier distillation-based methods such as LwF (Li & Hoiem, 2018) and iCaRL (Rebuffi et al., 2017) perform worse, likely due to less effective teacher selection: LwF and iCaRL use the latest model as a teacher, while DER uses snapshots from the model history. The true source of DER's improvement is still unclear, although recent analyses indicate that there may be more gains to be had from distillation in CL (Lanzillotta et al., 2025b).

Recent language-model adaptation methods extend this idea to *self-generated* memory objects: *context distillation* trains the model to reproduce its own context-conditioned outputs *without* the context, moving knowledge from the context window into the parameters (Snell et al., 2022); *on-policy distillation* trains the student on its own generations scored by a teacher (Agarwal et al., 2024); and "experiential learning" agents distill self-generated experience—insights, reasoning traces—into a reusable memory (Zhao et al., 2024). These methods blend two mechanisms of this section: the targets are distilled from a teacher, but the data is generated rather than stored, making them equally heirs of generative replay. In our framework the consolidation step is still a surrogate objective—the target stands in for a richer history of experience—and the open question is what that target preserves and discards: it may compress procedures far more efficiently than raw replay, yet lose counterfactuals, rare events, or alternative explanations. This question must be answered before systems that consolidate their own generated experience can be trusted in deployment (section 5.1).

4.1.2 Regularization

Instead of relying on past data or data distributions to approximate the past loss function, regularization methods construct a surrogate $g_{t-1} \approx \ell(\cdot, o_{1:t-1})$ as a function of the model parameters θ alone. The *regularizer* g_{t-1} is typically a smooth function designed to preserve essential geometric features of the past loss landscape. For example, when using gradient descent, it is useful if $\nabla g_{t-1}(\theta) \approx \nabla \ell(\theta, o_{1:t-1})$ for all θ. This gradient-matching view was made explicit in the *K-priors* framework (Khan & Swaroop, 2021; Daxberger et al., 2023), which designs g to reconstruct past gradients directly. Although standalone regularization

methods are often less competitive than replay on classical benchmarks, regularization remains pervasive as a component inside hybrid systems. It appears as adaptive uncertainty penalties (Ahn et al., 2019), auxiliary stability terms in adapter or LoRA-based methods (Biswas et al., 2026), and reference-policy penalties in reinforcement-learning-based post-training (Chen et al., 2026). In these settings, the regularizer rarely carries the whole burden of remembering the past; instead, it biases an update that is also shaped by routing, replay, or on-policy data. In fact, recent work on RL-based post-training suggests that retention can depend at least as much on the use of on-policy data as on the KL penalty itself (Chen et al., 2026).

Quadratic regularization. The most widely used regularizers are quadratic functions in θ, which typically consist of a Mahalanobis distance from an anchor point, usually chosen to be the model parameters at the end of the previous observation cycle θ_{t-1}. Formally:

$$g_{t-1}(\theta) = (\theta - \theta_{t-1})^\top Q_{t-1}(\theta - \theta_{t-1}), \tag{8}$$

where Q_{t-1} is a positive semi-definite matrix encoding the metric. The first use of quadratic regularization was in Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), where Q_{t-1} is the diagonal of the Fisher Information Matrix. Similar formulations such as Synaptic Intelligence (SI) (Zenke et al., 2017) and Memory Aware Synapses (MAS) (Aljundi et al., 2018) were shown by Benzing (2022) to be further instances of FIM-based approximations. The approximation in eq. (8) may seem too restrictive and inaccurate for neural networks; however, it can be derived from a second-order Taylor expansion of the historical risk $\mathcal{T}_2(\theta; \ell_{1:t-1}, \theta_{t-1})$, when assuming that θ_{t-1} is a local minimum thereof.

$$\mathcal{T}_2(\theta; \ell_{1:t-1}, \theta_{t-1}) = \nabla \ell_{1:t-1}(\theta_{t-1})^\top (\theta - \theta_{t-1}) + \tfrac{1}{2}(\theta - \theta_{t-1})^\top \nabla^2 \ell_{1:t-1}(\theta_{t-1})(\theta - \theta_{t-1}) + \text{cnst}.$$

When θ_{t-1} is a local minimum, the first term vanishes, and one recovers the quadratic form with $Q_{t-1} = \nabla^2 \ell_{1:t-1}(\theta_{t-1})$ (up to a constant term). Moreover, when the model fits the data well (i.e., the error is close to zero), the Hessian is dominated by its Generalized Gauss-Newton (GGN) component. For negative log-likelihood objectives in the exponential family, the GGN coincides exactly with the Fisher Information Matrix (Martens, 2020). The GGN offers two distinct advantages over the exact Hessian: (1) it is guaranteed to be positive-semi-definite, ensuring convexity and non-negativity and (2) it can be computed from first derivatives only, which are cheaper than second derivatives. For very large models, materializing the full $P \times P$ matrix in memory may be infeasible (where P is the size of the model); thus, practical implementations rely on diagonal or block-diagonal approximations (Ritter et al., 2018; Yin et al., 2020; Kirkpatrick et al., 2017).

The derivation from Taylor expansion exposes a fundamental limitation of all approximations in the form of eq. (8): like the expansion itself, they are accurate only within a local neighborhood of the center θ_{t-1}. In practice, regularization methods are often used "improperly", allowing the model to drift far outside the region where the quadratic approximation remains valid. This locality is a simple but central failure mode: it helps explain why regularization methods often underperform replay, and why they are sensitive to hyperparameters that influence the local geometry—learning rate, batch size, and model capacity (as shown by Lanzillotta et al., 2024). Locality also fixes the cost of regularization. The step cost is low: g_{t-1} is evaluated once per update, independent of how much past data it summarizes, so it is $\mathcal{O}(1)$ against the $\mathcal{O}(B)$ of replay. The cost reappears at consolidation, in the accuracy of Q_{t-1}. Capturing the curvature of the full past loss at the current parameters θ_t requires reprocessing the history, at cost $\mathcal{O}(t)$. Most methods avoid this by updating Q_{t-1} from the most recent observation alone, using the additivity of the Fisher/Hessian to keep consolidation at $\mathcal{O}(1)$; the curvature is then estimated at a past optimum θ_i and no longer matches the geometry at θ_t, which increases ϵ_t. The trade-off is the general one: a smaller ϵ_t requires a richer metric or re-estimation over more of the history, both of which raise the consolidation cost. Thus most regularization methods occupy a cheap, low-fidelity point on the cost–risk frontier. A promising and largely open direction is to make this locality a first-class design constraint: pairing the regularizer with an explicit safety region within which the surrogate is known to be valid—monitoring the drift from the anchor, and re-estimating or tightening the penalty when the boundary is approached—while keeping the consolidation cost bounded. Efficient mechanisms of this kind would attack the failure mode of regularization directly, rather than compensating for it with heavier memory.

Variational inference perspective. Many regularization methods can also be understood as approximate Bayesian updates (Opper & Winther, 1999; Khan & Rue, 2023). Bayesian inference prescribes a principled way to update knowledge as data arrive sequentially. In the Bayesian setting, the posterior after observing o_t is given by:

$$P(\theta \mid o_{1:t}) \propto P(\theta \mid o_{1:t-1})\, P(o_t \mid \theta).$$

Approximating this posterior via a variational distribution $q_t(\theta)$ leads to the following objective:

$$q_t(\theta) = \arg\min_{q(\theta)} \mathrm{D}_{\mathrm{KL}}(q(\theta) \,\|\, P(\theta \mid o_{1:t})) \tag{9}$$

$$= \arg\min_{q(\theta)} \underbrace{\mathrm{D}_{\mathrm{KL}}(q(\theta) \,\|\, P(\theta \mid o_{1:t-1}))}_{\text{prior term}} - \underbrace{\mathbb{E}_{q(\theta)}[\log P(o_t \mid \theta)]}_{\text{likelihood term}} \tag{10}$$

This *Variational Continual Learning* (VCL) framework (Nguyen et al., 2018) and its extensions (Swaroop et al., 2019; Loo et al., 2021; Tseran et al., 2018) formalize continual learning as recursive Bayesian inference. Typically, $q(\theta)$ is chosen to be Gaussian or another exponential-family distribution, balancing tractability and expressiveness. Richer posteriors yield more faithful updates but also higher cost—mirroring the trade-offs in regularization. The quadratic regularizer can be derived as a special case of this Bayesian formulation under a Laplace approximation and mean-field assumption, an instance of the *delta method* (Oehlert, 1992). Khan (2025) further formalizes this connection, showing that EWC and related methods correspond to specific choices of approximate variational posteriors.

An alternative Bayesian perspective arises when the posterior is expressed in *function space* rather than parameter space, as in Gaussian Processes (GPs). Bui et al. (2017) extended sparse GPs to the streaming setting, and Titsias et al. (2020) adapted this idea to neural networks by using their feature maps as GP kernels. When viewed as a regularizer, this induces a *functional regularization* term (Pan et al., 2020) that penalizes deviations of the network's outputs from previous posterior predictions (a form of knowledge distillation) at selected inducing points. This perspective bridges regularization and experience replay methods as the inducing points play the role of a learned replay buffer in function space. Recent work continues to make this link explicit: VAR-GPs construct auto-regressive variational posteriors for sequential tasks (Kapoor et al., 2021), memory-based dual GPs combine sparse posterior summaries with explicit memory mechanisms (Chang et al., 2023), and adaptive inducing-point methods adjust model size as data arrive (Pescador-Barrios et al., 2025). These methods are useful boundary examples for the present taxonomy because the "memory" is neither a raw replay buffer nor a parameter penalty alone: it is a structured posterior summary whose size controls both computational cost and approximation quality.

Summary. Table 3 summarizes the surrogate-objective family. A consistent pattern emerges: data-based methods (replay, distillation) pay a step cost that scales with the amount of replay but approximate the past loss faithfully, whereas parameter-based methods (regularization, variational) make the step essentially free but incur a higher excess historical risk, which they control only locally around the anchor.

4.2 Trust-region methods

The algorithms reviewed in this section instantiate the trust-region STEP operator defined in Definition 2. They are distinguished by their specific definitions of the trust region $\mathrm{TR}_{t-1}(\theta_r)$, which, much like the approximations in surrogate-objective methods, yield distinct trade-offs between computational overhead and performance stability. We discuss GEM, OGD, and GPM in some detail because they make the geometry of the trust region explicit. Many recent methods use the same principle in less explicit form, by restricting updates to prompts, adapters, LoRA subspaces, sparse memory slots, or routed experts.

4.2.1 Orthogonality-based methods

This class of methods enforces the trust region by *projecting* each update so that it (approximately) preserves the loss, or the predictions, on past data. One of the pioneering approaches in this category, Gradient Episodic

Table 3: Surrogate-objective methods (section 4.1): each adds a surrogate g_{t-1} for the past loss. Costs give the leading data/time dependence per inner step (c^{step}) and per interaction (c^{cons}) under the standard implementations, with the shared $O(P)$ model-size factor omitted; they are coloured cheap / moderate / expensive, and each entry is justified in the corresponding paragraph of the text. Row shading follows table 2 (blue = surrogate-objective family), with the darker shade for data-based methods (replay, distillation) and the lighter shade for parameter-based ones (regularization, variational). For the parameter-based methods, both c^{cons} and ϵ improve only with the fidelity of the curvature/posterior approximation; analogously, for coresets, consolidation cost rises and ϵ falls with the fidelity of the selection. B: buffer size; M: inducing points; t: history length.

Method	**Memory** s_t	**Surrogate** g_{t-1}	c^{step}	c^{cons}	**excess hist. risk** ϵ
Experience replay	buffer $\mathcal{B}$	$\ell(\cdot, \mathcal{B})$	$\mathbf{O(B)}$	$\mathbf{O(1)}$	$\uparrow$ as t/B
Generative replay	generator $\mathcal{G}$	$\mathbb{E}_{\mathcal{G}}[\ell]$	$\mathbf{O(B)}$	**train** $\mathcal{G}$	$\uparrow$ $\mathcal{G}$ forgetting
Coreset	coreset S	$\ell(\cdot, S)$	$\mathbf{O(\lvert S\rvert)}$	$\uparrow$ selection fidelity	$\downarrow$ selection fidelity
Distillation	teacher (and buffer)	ℓ vs. teacher	$\mathbf{O(B)}$	$\mathbf{O(1)}$	as t/B, better consts.
Quadratic reg. (EWC)	anchor + metric Q	$\lVert\theta-\theta_{t-1}\rVert_Q^2$	$\mathbf{O(1)}$	$\uparrow$ curvature fidelity	$\downarrow$ curvature fidelity
Variational (VCL)	posterior	KL to prior	$\mathbf{O(1)}$	$\uparrow$ posterior fidelity	$\downarrow$ posterior fidelity
Functional / GP	inducing points	fn-space penalty	$\mathbf{O(M)}$	$\mathbf{O(M^2)}$	$\downarrow$ with M

Memory (GEM) (Lopez-Paz & Ranzato, 2017), explicitly constrains updates to prevent an increase in the loss of any past observation, up to a first-order approximation. For a candidate update vector δ, the trust region is defined as:

$$\mathrm{TR}_{t-1}(\theta) = \{\theta + \delta \in \mathbb{R}^d \,|\, \delta^\top \nabla_\theta \ell(\theta, o) \leq 0,\ \forall o \in \mathcal{B}_{t-1}\},$$

where $\mathcal{B}_{t-1}$ is again a buffer, i.e., a subset of the past observations $\mathcal{B}_{t-1} \subset o_{1:t-1}$. While the consolidation cost is low (restricted to buffer maintenance), the step cost is prohibitive: projecting onto $\mathrm{TR}_{t-1}(\theta)$ requires solving a quadratic program per iteration, with the number of constraints scaling with the buffer size. Due to this cost, it is unclear whether GEM would be preferable to a naive experience replay method if the same amount of computation were allocated to both. Chaudhry et al. (2019) proposed A-GEM, which enforces a single constraint on the average replay loss, compared to $|\mathcal{B}_{t-1}|$ constraints in GEM, achieving a more favorable trade-off between stability and efficiency.

Another representative example is *Orthogonal Gradient Descent* (OGD) (Farajtabar et al., 2020). Rather than keeping the past *loss* invariant, OGD keeps the network *function* invariant on past inputs: it constrains each update to be orthogonal to the span of the past output-gradients $\{\nabla_\theta f_k(\theta_{t-1}, x) : x \in \mathcal{B}_{t-1},\ k \in [C]\}$, where C is the number of output units and $\mathcal{B}_{t-1}$ is a buffer. This approach is effective when the network function is linear around θ_{t-1}, and therefore it preserves the function only *locally*, near θ_{t-1}; consistently, Lanzillotta et al. (2024) find that OGD behaves much like quadratic regularizers such as EWC, with similar performance and hyperparameter sensitivity.

Gradient Projection Memory (GPM) (Saha et al., 2021) removes this locality limitation by acting in *representation* space. Consider a linear layer $z_l = W_l\, p_{l-1}(x)$ with input activations $p_{l-1}(x)$. If the weight update Δ_l is orthogonal to the span of the past inputs, so that $\Delta_l\, p_{l-1}(x) = 0$ for every past x, then $(W_l + \Delta_l)\, p_{l-1}(x) = W_l\, p_{l-1}(x)$: the pre-activations, and by extension the entire forward pass on past inputs, are left unchanged *regardless of the step size*. The invariance is exact—not merely first-order as in OGD—because each layer is linear in its weights; this is why GPM outperforms OGD, as reported by Saha et al. (2021), and is more memory-efficient, the protected subspace living in the low-dimensional representation space. Its ϵ is a design choice: protecting the whole input subspace gives $\epsilon = 0$ but uses up capacity fastest (in proportion to the rank of the past features), while protecting only the top directions trades a little forgetting for more remaining plasticity (Barinka et al., 2026). Either way, the representation space is consumed quickly, so GPM saturates capacity and fits fewer tasks over a long horizon than OGD.

The GPM orthogonality principle has been repeatedly re-instantiated in the adapter-based fine-tuning of large pretrained models. The idea is unchanged—enforce an orthogonality constraint that, up to a tolerance ϵ, keeps the function invariant on the past—but the modern parameterization makes it far more effective. LoRA (Hu et al., 2022) adds to each frozen weight W_0 a low-rank update $\Delta W = AB$, so the trainable directions are by construction low-dimensional. Imposing the GPM condition on these directions keeps the function near-constant on the past while making efficient use of the model's diminishing learning capacity. O-LoRA (Wang et al., 2023b) and InfLoRA (Liang & Li, 2024) are direct instances: they confine each task's low-rank update to a subspace orthogonal to those of previous tasks, achieving strong continual-learning performance. This is a clean illustration of a recurring pattern: *an old idea—orthogonal trust regions in the style of GPM—is substantially improved by pairing it with a modern architecture and parameterization, not by a new principle*; low-rank adaptation is simply a much better-conditioned setting for GPM-like methods than full fine-tuning.

More broadly, LoRA-based continual learning methods should be separated into two cases. Some methods define an explicit trust region through orthogonality, subspace design, or low-loss connectivity (Liang & Li, 2024; Wang et al., 2023b; Chen et al., 2025). Other methods do not impose a formal constraint, but exploit the optimization bias induced by low-rank adaptation: since the pretrained backbone is frozen and only a small low-dimensional update is trained, the model is simply altered less—fewer directions of change mean less overlap with the directions that support past behavior, hence less interference overall—so more of the reference model is retained than under full fine-tuning, although usually at the cost of learning less from the new data (Biderman et al., 2024). This makes LoRA naturally attractive for continual learning, but it should not be mistaken for a guarantee of non-interference. Indeed, recent sparse-memory results show that LoRA can forget much less than full fine-tuning while still forgetting substantially more than more selective sparse update mechanisms (Lin et al., 2025). Existing CL methods therefore add orthogonality, decoupling, interpolation, replay, or regularization on top of LoRA to make this implicit retention bias more reliable (Wu et al., 2025; Biswas et al., 2026).

4.2.2 Masking, network expansion and routing

The methods in this section build function invariance *by construction* rather than by an explicit constraint. They partition the network parameters among different inputs—a strategy often referred to as *conditional computation* (Bengio et al., 2013; 2016; Shazeer et al., 2017)—so that updating the parameters for one input cannot alter the function on inputs routed elsewhere. The most generic formulation of conditional computation is obtained by defining a *routing mechanism* $r(x, l)$, which specifies which parameters are active for a given input x and task identifier or label l. [3] The routing mechanism can be seen as designing an axis-aligned trust region for each input or task, freezing the values of all inactive parameters:

$$\mathrm{TR}_{t-1}(\theta_{t-1}) = \{\theta \in \mathbb{R}^d \,|\, \theta_j = (\theta_{t-1})_j \quad \forall j \notin r(x_t, l_t)\}.$$

Note that, by Definition 2, the trust region constrains only the *update*: it is enforced in the backward pass, where the parameters outside $r(x_t, l_t)$ receive no gradient, and it leaves the forward pass untouched. The forward computation may therefore activate any parameters—including previously learned ones—to leverage existing knowledge (transfer); only in fully-routed architectures, such as Mixtures of Experts, does the forward pass itself change with the routing decision. For pathways whose parameters lie outside every later mask, *the network function remains constant on the past*. Since this invariance is built into the architecture, the *consolidation cost* is often minimal: the primary overhead is the cost of designing and executing the router. However, zero-cost implementations (like rigid partitioning) often use model capacity inefficiently, fitting fewer tasks per parameter compared to methods that allow controlled sharing.

Different algorithms use different routing mechanisms; below we review the main examples. Figure 3 sketches the three: a hard mask over a shared network, width-growth of the hidden layer with a frozen old subset, and a learned gate over experts—each a different shape of router for the same input/output.

Dynamic masking. One efficient way to design the router is as a binary mask over neurons or weights for each task, which can be learned through gradient descent (Serrà et al., 2018; Wortsman et al., 2020; Kang

[3] This defines a hard routing mechanism; soft routing can also be used, though it may re-introduce interference.

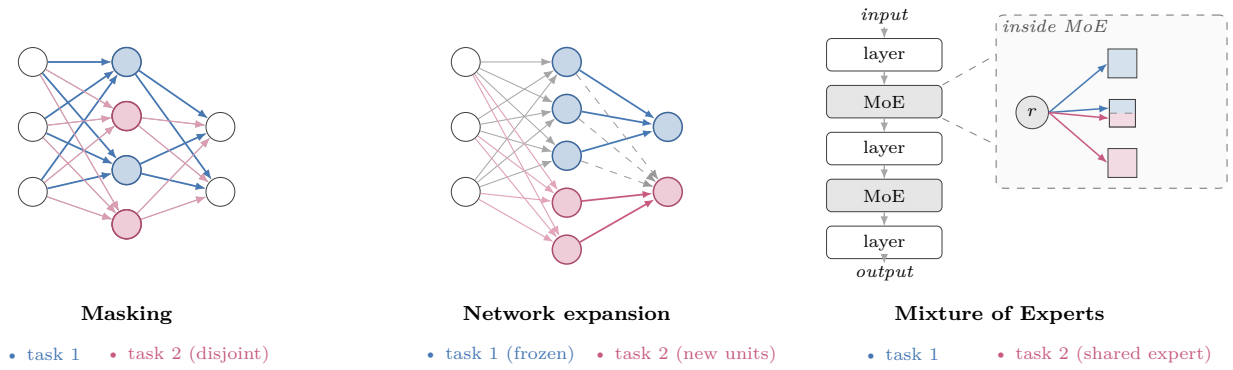

Figure 3: **Three shapes of router for the same input/output.** *Masking* (left): a hard, task-indexed mask selects a disjoint subset of hidden units for each task; the blue path is task 1, the pink units are reserved for task 2. *Network expansion* (middle): the hidden layer grows in width—task 1's units (blue) are frozen, new units (pink) are added in the same layer for task 2, with the frozen units still feeding the task-2 output (dashed) for transfer. *Mixture of Experts* (right): a learned router r dispatches each input to one or more experts inside an architecture that interleaves MoE layers with normal layers; the zoom-in shows the gate plus three experts, with the middle expert shared across both tasks (split blue/pink fill), illustrating soft routing.

et al., 2022). Similarly, the mask can be identified post-training through pruning (Mallya & Lazebnik, 2018; Fernando et al., 2017; Rajasegaran et al., 2019; Veniat et al., 2021), which allows more flexibility to the network. In modern pretrained models, the same principle often appears as sparse or modular updates: only a small set of parameters, memory slots, representation interventions, or adapter components is modified while the rest of the model is frozen (Lin et al., 2025; Wu et al., 2024). In both paradigms, the forward pass typically utilizes a superset of parameters (including those trained on prior tasks) to maximize performance. Conversely, the backward pass restricts updates strictly to the current mask, freezing any parameters shared with previous masks. Allowing *parameter-reuse* across tasks by distinguishing between shared and task-specific parameters, allows a more flexible use of the model capacity. However, the network function is no longer guaranteed to be constant on the past, which may lead to forgetting (Ebrahimi et al., 2020; Hurtado et al., 2021).

Network expansion. A dual approach to dynamic masking is to *expand* the network architecture at each task to allow for more capacity, while simultaneously freezing the backbone. This is effectively equivalent to an implicit static masking applied to a larger network—one in which the forward pass, too, is masked: earlier tasks see only the capacity that existed when they were learned. Classic examples of this approach are the so-called *expansion methods*, prominent among them *Progressive Neural Networks* (PNN) (Rusu et al., 2016), which allocate a new column of parameters per task, and *Compacting, Picking, Growing* (CPG) (Hung et al., 2019), which mitigates the growth of the network by pruning. Recent work on prompt- and prefix-tuning methods, including Learning to Prompt (L2P) (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), CODA-Prompt (Smith et al., 2023), and HiDe-Prompt (Wang et al., 2023a), similarly introduce small task-specific learnable tokens or prefixes, which can be viewed as structured adapters (He et al., 2022). Importantly, *prompt-based continual learning can be read not as a separate principle, but as a parameter-isolation and routing method in which the protected object is the frozen pretrained backbone and the adaptive state is moved into a small prompt memory*: the pool of prompts grows like the modules of an expansion method, and the query–key matching that selects a prompt for each input is a learned router. In the limiting case, the backbone is frozen entirely and only the classifier head is adapted or statistically calibrated, as in continual learning methods built directly on pretrained models such as SLCA (Zhang et al., 2023) and RanPAC (McDonnell et al., 2023). The central scaling question is therefore the same as for classical expansion methods: whether the prompt, adapter, or module pool grows with the number of tasks, or whether the router can select reusable components to improve the learning performance without reintroducing interference.

Mixture of Experts. *Mixture of Experts* (MoE) architectures define a parametric router function, which is typically *learned* end-to-end with the rest of the network. Experts are subnetworks or parameter subsets of specific layers, and when an input is routed to an expert, only that expert is trained while the rest of the model remains frozen; importantly, and unlike in masking, here the forward pass itself changes with the routing decision. MoE architectures were first adopted in CL by Aljundi et al. (2017); Collier et al. (2020); Ramesh & Chaudhari (2022). Recently, MoE architectures have gained renewed prominence in large-scale language modeling (Fedus et al., 2022; Du et al., 2022; Zhang et al., 2025), offering a mechanism to scale model capacity massively without a proportional increase in inference compute. Modern continual learning MoE approaches (Gururangan et al., 2022; Chen et al., 2023; Zhang et al., 2025; Dou et al., 2024) leverage this modularity to improve forward transfer. Some add new experts for new domains or languages, some route old inputs back to protected experts, and others combine MoE routing with LoRA-style adapters.

Across masking, expansion, and routing methods, the recurring tension is between *capacity reuse* and *forgetting*, and it is concentrated in the router. Hard, task-indexed routing resolves the tension by fiat: parameters are isolated, so nothing is forgotten, but nothing is shared—backward transfer is forbidden and capacity is spent linearly in the number of tasks (Rusu et al., 2016; Hung et al., 2019). Learned routers and MoE layers recover reuse and transfer, but every shared component reintroduces interference, and what is shared is overwritten unless replay, regularization, or explicit routing constraints protect it (Aljundi et al., 2017; Collier et al., 2020; Chen et al., 2023; Zhang et al., 2025; Dou et al., 2024). Modularity alone is therefore not a continual-learning solution: it lowers the cost of conditional computation, but the stability–plasticity trade-off reappears, undiminished, in the decision of when two inputs should share parameters.

Table 4: Trust-region methods (section 4.2). Here we report only what distinguishes the methods within the family—their cost, and the **plasticity** they retain. By construction the excess risk ϵ stays near zero until capacity saturates, so the characteristic cost of the family is paid not in forgetting but in lost plasticity, the remaining ability to fit new tasks. Costs report the leading data/time dependence, omit the shared $O(P)$ factor, and are coloured cheap / moderate / expensive; each entry is justified in the corresponding paragraph of the text; r is the adapter rank. GEM's step cost includes a per-iteration quadratic program (QP) with one no-loss-increase constraint per buffered sample, and its plasticity declines as these constraints accumulate with every task. Expansion methods keep plasticity by growing the network, but the larger model raises the per-step (and inference) cost, so their cost is not constant. Prompt pools keep plasticity by growing the pool, but a larger pool increases selection interference unless a task identifier is available. The **task info** column records what task information the method needs: task *boundaries* during training, a task *ID* at test time, or neither (✗); methods that need task information extend to the task-free setting at a cost in performance. Row shading follows table 2 (pink = trust-region family), with the darker shade for orthogonality-based methods and the lighter shade for architecture-based ones.

Method	task info	c^{step}	c^{cons}	plasticity
GEM	✗	**O(B)+QP**	**O(1)**	↓ (constraints pile up)
OGD	boundaries	**O(1)**	**SVD / task**	↓ with t
GPM	boundaries	**O(1)**	**SVD / task**	↓ as subspace fills
Orth. LoRA	boundaries	**O(r)**	**subspace constr.**	↓ as subspaces fill
Masking (PackNet)	boundaries + ID	**O(1)**	**O(1)**	↓ capacity carved up
Expansion (PNN)	boundaries + ID	**↑ model size**	**O(1)**	kept (model grows)
Prompt pool (L2P)	✗	**O(pool) match**	**key upd. (↑ t)**	kept (pool grows; interf.)
MoE	✗	**routing**	**router update**	kept (experts grow)

Summary. Table 4 summarizes the trust-region family. Across it a common signature emerges, and it is the mirror image of the surrogate-objective family: trust-region methods are designed to retain the past (almost)

exactly, so their characteristic failure mode is not forgetting but *loss of plasticity*—as protected or frozen components accumulate, the capacity to fit new tasks shrinks, unless the model is grown at a cost in size. Surrogate-objective methods, by contrast, keep plasticity but must continually balance it against stability through λ. Whether to pay for retention in plasticity or in a dynamic balance is, at bottom, the central design choice between the two families.

4.3 Boundary cases: memory and adaptation beyond a single parameter trajectory

We close the review with the adaptation mechanisms that our taxonomy does not fully capture. In practice, many algorithms mix the primitives identified above: a method may use replay together with a LoRA subspace adaptation rule or regularization, or routing together with a small memory buffer. The taxonomy is therefore best read in terms of *primitives of memory*—what the algorithm's state keeps of the past, and how that content is used—and some systems, discussed below, are built from primitives that lie outside a single parameter trajectory altogether.

Retrieval and external memory. Retrieval-augmented and agentic-memory systems retain information in an external database or context window rather than in the model parameters (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022), and adapt behavior by changing the input context, with zero gradient steps. These methods are orthogonal to the methods above and can be combined with any of them. We regard such *hybrid* systems—some knowledge kept in an external, easily-edited memory, the rest consolidated into the weights—as a particularly promising direction, recently argued for at length from a memory-systems perspective (Dorovatas et al., 2026). The reasoning is twofold: it is often unclear in advance whether a piece of information is worth keeping long-term, and information written into the parameters is notoriously hard to remove (Chang & Khanna, 2026). Storing uncertain or short-lived knowledge externally sidesteps both problems. The need for algorithms that can reliably consolidate knowledge into the parameters is still present: once a piece of knowledge is judged worthy of permanent, low-latency, parametric storage, consolidating it cheaply and without forgetting is again the historical-risk approximation problem of this review. The two questions are complementary, and both remain open—how to consolidate, and how to divide labor between the two memories: what to retrieve, what to internalize, and when to migrate knowledge from one to the other.

Model editing. Model editing (Meng et al., 2022; 2023; Hartvigsen et al., 2023; Wang et al., 2024) is closer to the present taxonomy than retrieval but differs in objective. Mechanistically, the localized editors (Meng et al., 2022; 2023) are trust-region-like, where the protected region is defined by locality and side-effect tests rather than by an explicit loss on $o_{1:t}$. The lifelong, sequential editors (Hartvigsen et al., 2023; Wang et al., 2024) add parameter isolation and retrieval on top. What sets editing apart is not a new primitive but a different goal: there is no emphasis on retention of a past *learning experience*. Evaluation targets local edit success and side-effect control, not minimization of historical risk over previous edits, and edits are applied to a fixed request set rather than as a default update after every observation.

Model merging and task arithmetic. Model merging and task arithmetic are different again: they combine independently trained parameter vectors or task vectors after training (Matena & Raffel, 2022; Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2023). These are legitimate continual learning methods, but we do not treat them in depth here because they are not optimization-based in the sense of algorithm 1, and we refer the reader to dedicated surveys. Conceptually they connect to posterior combination, low-loss connectivity, and interference resolution, and the principle underlying their success is still only partially understood.

4.4 Continual learning is also an optimization problem: the substrate matters

The review so far has treated the STEP operator as the locus of continual learning, with the base optimizer, architecture, and initialization as a neutral backdrop. This is misleading. Continual learning is, in large part, an optimization problem, and the *substrate* on which optimization takes place—the inductive bias of the optimizer, the architecture, the starting point—strongly affects how much a model forgets, can reduce

forgetting on its own, and is often decisive for whether a given STEP operator works at all. It cannot be abstracted away.

Loss of plasticity. The clearest evidence that the substrate matters is *loss of plasticity*: after many updates, a network can progressively lose the ability to fit *new* data at all (Dohare et al., 2024; Lyle et al., 2023). This is a property of the optimization substrate, not of any STEP operator: it is mitigated by substrate-level interventions such as continual-backpropagation-style resets and regenerative regularization (Dohare et al., 2021; Kumar et al., 2024). Its severity is strongly regime-dependent, most visible over long update streams, in reinforcement learning, and with aggressive optimizer settings, and far less so in short task-incremental supervised benchmarks, with pre-trained backbones, or with moderate learning rates.

Optimizer bias, laziness, and pre-training. The optimization level also affects forgetting. Different optimizers carry different inductive biases that determine the minima reached. Mirzadeh et al. (2020) observed that higher learning rates, smaller batch sizes, and dropout implicitly reduce forgetting, and Graldi et al. (2025) show that models in a *lazy* regime—small parameter movement relative to initialization—produce more stable representations. This helps reconcile otherwise contradictory observations about scale: larger models often forget less not because of capacity alone, but because scale pushes optimization into a more stable, lazy regime. The starting point is equally decisive: initializing from self-supervised or pre-trained representations markedly improves stability, suggesting that such features are intrinsically more robust to task-specific interference (Madaan et al., 2022; Mehta et al., 2023).

Depth of forgetting. Forgetting is not uniform across the depth of the network. Using probes to read task-relevant information out of intermediate layers, Ramasesh et al. (2021) and later Hess et al. (2024) found that earlier layers are markedly more stable than later ones; the residual instability of intermediate features is termed *feature* or *deep* forgetting, usually measured at the last layer before the head. Because identifying the optimal read-out requires full data access, this is a diagnostic rather than a practical tool, and it has motivated methods that approximate the optimal head with little or no past data (Hou et al., 2019; Wu et al., 2019; Zhao et al., 2020). The distinction has practical consequences: Lanzillotta et al. (2025a) show that experience replay with very small buffers can curb deep forgetting while failing to reduce head-level forgetting, indicating that different mechanisms act at different depths.

In short, continual learning methods are not applied in a vacuum. As the LoRA discussion of section 4.2.1 illustrated, the same mechanism can fail under full fine-tuning and succeed on a better-conditioned substrate. A finer understanding of how the substrate—optimizer, architecture, initialization—interacts with the choice of algorithm may bring considerable gains and widen the range of applicability of existing algorithms. Characterizing this interaction is another promising direction for theoretical work.

5 Discussion and open problems

Summary. We have made four connected claims in this review, mirroring the four announced in section 1. *First*, that the bulk of optimization-based continual learning algorithms can be read through a single lens: each approximates the minimizer of the historical risk under a computational budget, either by adding a surrogate term g_{t-1} to the objective or by restricting updates to a trust region TR_{t-1}. *Second*, that this reading sorts the field into two mechanistic families whose costs and failure modes mirror each other—surrogate methods trade stability against plasticity through λ and fail by drift; trust-region methods retain by construction and fail by loss of plasticity. *Third*, that comparing methods honestly requires a cost model exposing how computation scales with the history: our step/consolidation decomposition makes this explicit and reveals a Pareto frontier between excess risk ϵ and cost C_t which, we argue, is the real frontier of progress in the field. *Fourth*, that on this frontier no existing method reaches the regime that matters—low ϵ at sublinear cost—so progress is more limited than the volume of published methods suggests; the recent foundation-model wave, which reinstantiates classical mechanisms under new names, has not so far moved this frontier.

We now use this lens to take stock: we make the case for a theoretical foundation (section 5.1), list the open problems the review surfaces (section 5.2), and conclude with some remarks on the limitations of this work (section 5.3).

5.1 The case for a theoretical foundation

Continual learning has been organized in several ways: by the type of incremental setting (task-, domain-, and class-incremental) (van de Ven et al., 2022), by broad method family (replay, regularization, architectural) (De Lange et al., 2022), and through a unifying Bayesian recursion (Farquhar & Gal, 2019). These taxonomies organize the *methods*, but they say little about the *goal* of continual learning at an essential level, or about its fundamental limits. Our *taxonomy* is complementary and mechanism-based, and its aim is less to divide the field than to *unify* it: by exposing the principles the usual families share, it makes visible which methods are genuinely distinct and which are re-instantiations. The Bayesian view of Farquhar & Gal (2019) is essentially its surrogate-objective branch, with the prior term acting as an ϵ-accurate surrogate of the past.

We see this re-framing as part of a broader need. The field's progress is largely measured by accuracy gains, but the standards under which they are reported vary from paper to paper—different benchmarks, compute budgets, buffer sizes, and pretraining—so methods are rarely directly comparable: the field lacks both an equalized evaluation framework and a shared statement of what the core problem is. This has produced a large catalogue of methods without a clear account of whether the core problem is being solved. A method that tops a 20-task leaderboard may simply have enough spare capacity to store 20 masks, and the benchmark cannot reveal whether it would survive 20,000 tasks, or at what cost. What is missing is not more methods but more *essential thinking*: benchmark-independent statements of what a continual learner should achieve (for instance, sublinear adaptation at bounded ϵ); a theoretical characterization of the trade-offs and roadblocks on the way (the ϵ–cost frontier of fig. 2, capacity exhaustion, loss of plasticity); and, ideally, lower bounds that separate what is hard-but-possible from what is impossible. The framework of this review is a step in that direction, but only a first one.

There is also a practical argument for theory beyond the intellectual one. Continual learning systems are meant to keep changing after deployment, and a system that changes after deployment can only be trusted with guarantees on *how* it changes: what it retains, what it optimizes, and when it fails. Without such guarantees—that is, without a deeper understanding of what these methods actually do—the large-scale deployment of continually learning systems is difficult to imagine. Theory, here, is a precondition for practice.

A theory of continual learning need not be built from nothing. Neighbouring fields—online learning, optimization, meta-learning—already rest on solid theoretical foundations, and continual learning, read as constrained minimization of the historical risk, makes close contact with all of them. Building these connections and importing the formalisms of these more mature fields is, we believe, one of the most promising and important directions for continual learning.

5.2 Open problems

To make the stated call concrete rather than rhetorical, we close the paper with the open problems it surfaces.

No method is yet sublinear without loss. As summarized in tables 3 and 4, the methods we surveyed either keep a bounded consolidation cost only by letting performance degrade over a long horizon, or maintain a low excess risk only by paying a cost in compute, memory, or model size that grows with the history. We are not aware of any method that achieves sublinear adaptation $C_t = o(t)$ at bounded ϵ. The claim should also be put to the test: the frontier of fig. 2 ought to be traced empirically—running representative mechanisms of each family at matched compute budgets over long horizons, and recording the excess risk each sustains as the history grows. To our knowledge, no such systematic measurement exists, and it is among the most immediately actionable items on this list. The real question, however, is sharper: can any algorithm reach the intrinsic floor $m_\epsilon(t)$ set by the compressibility of the stream (section 3.3), or does a fundamental limit keep every method strictly above it? Whether the gap we observe is such a limit or merely an unsolved problem is among the most important open questions of the field.

What to remember. We introduced the view that *what* to remember and *how* to remember are orthogonal questions, and the algorithms reviewed here answer only the second, so they transfer directly to any choice of target. To retain a chosen subset rather than everything, replay stores only that subset, EWC anchors on those skills alone, GPM protects only their feature subspace, and expansion or mixture-of-experts methods simply allocate no new module for tasks outside the target: the mechanism is unchanged, only what it is asked to preserve shrinks. The default target—"everything observed"—is not always right: under the Big World Hypothesis (Javed & Sutton, 2024) and the computational view of Kumar et al. (2025), a finite model cannot retain everything, so some selection is unavoidable. Making the selection an explicit, learned decision is an open direction; SEAL (Zweiger et al., 2025), for instance, learns through an outer reinforcement-learning loop which self-generated data to consolidate, yet still forgets and falls back on a trust-region strategy—once more, choosing *what* does not remove the need for an efficient *how*.

The two timescales. A continual learner runs on two clocks: the environment produces observations at its own rate, while the learner chooses when to update. Our cost model exposes the two knobs that set this schedule—the consolidation frequency ν and the per-observation effort K_t (section 3.3)—but is silent on how to choose them. The choice is a genuine open problem: updating too rarely leaves the model stale; updating too often wastes computation on fluctuations that the next observation may wash out. The problem sharpens when the time spent updating is itself part of the agent's response—when the model must act before it has finished learning, as in reinforcement learning or interactive deployment. Our accounting assumes it need not; relaxing this assumption couples learning speed to response latency, and what an optimal schedule looks like under that coupling is largely unexplored.

Equivalences between the two formulations. Surrogate objectives and trust regions are the penalty and constrained forms of the same program (section 3.4), but the correspondence is understood only loosely. A precise account of the connections between the two forms—which surrogate g_{t-1} matches which trust region TR_{t-1}, when one can be converted into the other, and at what cost in excess risk ϵ and computation—would let results proved for one family transfer to the other, and would help separate a method's mechanism from its formulation.

5.3 Limitations

This review has clear limits. It is *conceptual*: we run no experiments, and the figures and tables are illustrative rather than measured. Our cost accounting is deliberately coarse—leading-order data/time scaling with model-size factors held fixed, under the standard implementations—so the per-method costs should be read as scaling signatures, not benchmark numbers. The lens, while broad, is not universal: it is built for optimization-based methods that follow a single parameter trajectory, and it treats merging, editing, and retrieval as boundary cases rather than first-class citizens. Finally, our coverage of the vast continual-learning literature is necessarily selective; we have aimed for representative mechanisms rather than exhaustive citation, and our reading of "progress" is an argued position, not a measured fact.

Broader Impact Statement

This is a conceptual review: it introduces no new models, datasets, or deployable systems, so its direct risks are limited. Two indirect points are nonetheless worth noting. First, the central goal we articulate—integrating new knowledge into a model's weights at sublinear cost—would, if achieved, make it substantially cheaper to keep large models continuously up to date. Like any efficiency gain this is dual-use: it lowers the cost of beneficial adaptation but equally that of undesirable continual fine-tuning. Second, we emphasize (section 4.3) that knowledge written into weights is hard to remove; for applications involving personal or sensitive data, this is an argument for keeping such information in external, editable memory rather than consolidating it into parameters—a consideration relevant to privacy and the "right to be forgotten." We foresee no other negative societal impacts specific to this work.

References

Istabrak Abbes, Gopeshh Subbaraj, Matthew Riemer, Nizar Islah, Benjamin Therien, Tsuguchika Tabaru, Hiroaki Kingetsu, Sarath Chandar, and Irina Rish. Revisiting Replay and Gradient Alignment for Continual Pre-Training of Large Language Models. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2025.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based Continual Learning with Adaptive Regularization. In *Advances in Neural Information Processing Systems*, 2019.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert Gate: Lifelong Learning with a Network of Experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154. Springer, 2018.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019b.

Louis Barinka, Giulia Lanzillotta, Enis Simsar, Tanguy Dieudonné, and Thomas Hofmann. Provable Forgetting Bounds Drive Capacity Savings: Spectral Thresholding in Continual LoRA. In *Continual Adaptation at Scale: Towards Sustainable AI Workshop*, 2026. URL https://openreview.net/forum?id=559x6AYm1p.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional Computation in Neural Networks for faster models. arXiv preprint arXiv:1511.06297, January 2016.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv preprint arXiv:1308.3432, August 2013.

Frederik Benzing. Unifying Importance Based Regularisation Methods for Continual Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA Learns Less and Forgets Less. *Transactions on Machine Learning Research*, 2024.

Shristi Das Biswas, Yue Zhang, Anwesan Pal, Radhika Bhargava, and Kaushik Roy. ELLA: Efficient Lifelong Learning for Adapters in Large Language Models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2026.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving Language Models by Retrieving from Trillions of Tokens. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 2206–2240, 2022.

Zalán Borsos, Mojmír Mutný, and Andreas Krause. Coresets via Bilevel Optimization for Continual Learning and Streaming. In *Advances in Neural Information Processing Systems*, 2020.

Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-Incremental Continual Learning into the eXtended DER-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5497–5512, 2023. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2022.3206549.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, 2020.

Thang D. Bui, Cuong V. Nguyen, and Richard E. Turner. Streaming Sparse Gaussian Process Approximations. In *Advances in Neural Information Processing Systems*, 2017.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: A Strong, Simple Baseline. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930. Curran Associates, Inc., 2020.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking Experience Replay: A Bag of Tricks for Continual Learning. In *International Conference on Pattern Recognition (ICPR)*, 2021.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New Insights on Reducing Abrupt Representation Change in Online Continual Learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Paul E. Chang, Prakhar Verma, S. T. John, Arno Solin, and Mohammad Emtiyaz Khan. Memory-Based Dual Gaussian Processes for Sequential Learning. In *International Conference on Machine Learning*, 2023.

Wei-Kai Chang and Rajiv Khanna. Why Some Models Resist Unlearning: A Linear Stability Perspective, 2026.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations*, 2019.

Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6993–7001, 2021.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by Doing: The Role of On-Policy Data in Mitigating Forgetting. In *International Conference on Machine Learning*, 2026.

Huancheng Chen, Jingtao Li, Weiming Zhuang, Chen Chen, and Lingjuan Lyu. Replay-Free Continual Low-Rank Adaptation with Dynamic Memory. arXiv preprint arXiv:2411.00623, September 2025.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cu. Lifelong Language Pretraining with Distribution-Specialized Experts. In *International Conference on Machine Learning*, 2023.

Mark Collier, Efi Kokiopoulou, Andrea Gesmundo, and Jesse Berent. Routing Networks with Co-training for Continual Learning. arXiv preprint arXiv:2009.04381, September 2020.

Erik Daxberger, Siddharth Swaroop, Kazuki Osawa, Rio Yokota, Richard E. Turner, José Miguel Hernández-Lobato, and Mohammad Emtiyaz Khan. Improving Continual Learning by Accurate Gradient Reconstructions of the Past. *Transactions on Machine Learning Research*, July 2023. ISSN 2835-8856.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, July 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3057446.

Shibhansh Dohare, Richard S Sutton, and A Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, August 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07711-7.

Vaggelis Dorovatas, Malte Schwerin, Andrew D. Bagdanov, Lucas Caccia, Antonio Carta, Laurent Charlin, Barbara Hammer, Tyler L. Hayes, Timm Hess, Christopher Kanan, Dhireesha Kudithipudi, Xialei Liu, Vincenzo Lomonaco, Jorge Mendez-Mendez, Darshan Patil, Ameya Prabhu, Elisa Ricci, Tinne Tuytelaars, Gido M. van de Ven, Liyuan Wang, Joost van de Weijer, Jonghyun Choi, Martin Mundt, and Rahaf Aljundi. Position: Modular Memory Is the Key to Continual Learning Agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviate World Knowledge Forgetting in Large Language Models via MoE-Style Plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In *International Conference on Machine Learning*, 2022.

Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial Continual Learning. In *European Conference on Computer Vision (ECCV)*, 2020.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal Gradient Descent for Continual Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773. PMLR, June 2020.

Sebastian Farquhar and Yarin Gal. A Unifying Bayesian View of Continual Learning. arXiv preprint arXiv:1902.06494, February 2019.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*, 2022.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734, January 2017.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4): 128–135, 1999. ISSN 13646613. doi: 10.1016/S1364-6613(99)01294-2.

Rui Gao and Weiwei Liu. DDGR: Continual Learning with Deep Diffusion-based Generative Replay. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10744–10763. PMLR, July 2023.

Jacopo Graldi, Alessandro Breccia, Giulia Lanzillotta, Thomas Hofmann, and Lorenzo Noci. The Importance of Being Lazy: Scaling Limits of Continual Learning. In *International Conference on Machine Learning*, 2025.

Haiyang Guo, Fanhu Zeng, Fei Zhu, Jiayi Wang, Xukai Wang, Jingang Zhou, Hongbo Zhao, Wenzhuo Liu, Shijie Ma, Da-Han Wang, Xu-Yao Zhang, and Cheng-Lin Liu. Continual Learning for Generative AI: From LLMs to MLLMs and Beyond. arXiv preprint arXiv:2506.13045, June 2025.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. DEMix Layers: Disentangling Domains for Modular Language Modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2022.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 3929–3938, 2020.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong Model Editing with Discrete Key-Value Adaptors. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tyler L. Hayes, Giri P. Krishnan, Maxim Bazhenov, Hava T. Siegelmann, Terrence J. Sejnowski, and Christopher Kanan. Replay in Deep Learning: Current Approaches and Missing Biological Elements. *Neural computation*, 33(11):2908–2950, October 2021. ISSN 0899-7667. doi: 10.1162/neco_a_01433.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*, 2022.

Timm Hess, Eli Verwimp, Gido M van de Ven, and Tinne Tuytelaars. Knowledge accumulation in continually learned representations and the issue of feature forgetting. *Transactions on Machine Learning Research*, 2024.

Truman Hickok. Scalable Strategies for Continual Learning with Replay. arXiv preprint arXiv:2505.12512, May 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

Steven C. Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, Picking and Growing for Unforgetting Continual Learning. In *Advances in Neural Information Processing Systems*, 2019.

Julio Hurtado, Alain Raymond-Saez, and Alvaro Soto. Optimizing Reusable Knowledge for Continual Learning via Metalearning. In *Advances in Neural Information Processing Systems*, 2021.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and Scalable Strategies to Continually Pre-train Large Language Models. *Transactions on Machine Learning Research*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing Models with Task Arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-Efficient Incremental Learning Through Feature Adaptation. In *European Conference on Computer Vision (ECCV)*, 2020.

Khurram Javed and Richard S Sutton. The Big World Hypothesis and its Ramifications for Artificial Intelligence. In *Finding the Frame Workshop at the Reinforcement Learning Conference (RLC)*, 2024.

Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D. Yoo. Forget-free Continual Learning with Winning Subnetworks. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 10734–10750. PMLR, June 2022.

Sanyam Kapoor, Theofanis Karaletsos, and Thang D Bui. Variational Auto-Regressive Gaussian Processes for Continual Learning. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Mohammad Emtiyaz Khan. Knowledge Adaptation as Posterior Correction. arXiv preprint arXiv:2506.14262, June 2025.

Mohammad Emtiyaz Khan and Håvard Rue. The Bayesian Learning Rule. *Journal of Machine Learning Research*, 2023.

Mohammad Emtiyaz Khan and Siddharth Swaroop. Knowledge-Adaptation Priors. In *Advances in Neural Information Processing Systems*, 2021.

Junsu Kim, Yunhoe Ku, Dongyoon Han, and Seungryul Baek. Beyond Synthetic Replays: Turning Diffusion Features into Few-Shot Class-Incremental Learning Knowledge. arXiv preprint arXiv:2503.23402, March 2025.

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.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining Plasticity in Continual Learning via Regenerative Regularization. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2024.

Saurabh Kumar, Henrik Marklund, Ashish Rao, Yifan Zhu, Hong Jun Jeon, Yueyang Liu, and Benjamin Van Roy. Continual learning as computationally constrained reinforcement learning. *Foundations and Trends in Machine Learning*, 2025.

Giulia Lanzillotta, Sidak Pal Singh, Benjamin F. Grewe, and Thomas Hofmann. Local vs Global continual learning. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2024.

Giulia Lanzillotta, Damiano Meier, and Thomas Hofmann. Asymptotic analysis of shallow and deep forgetting in replay with Neural Collapse. https://arxiv.org/abs/2512.07400v1, December 2025a.

Giulia Lanzillotta, Felix Sarnthein, Gil Kur, Thomas Hofmann, and Bobby He. Revisiting Knowledge Distillation: The Hidden Role of Dataset Size. arXiv preprint arXiv:2510.15516, October 2025b.

Shane Legg and Marcus Hutter. A Collection of Definitions of Intelligence. In *Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms*, 2007.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Yan-Shuo Liang and Wu-Jun Li. InfLoRA: Interference-Free Low-Rank Adaptation for Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. Continual Learning via Sparse Memory Finetuning. arXiv preprint arXiv:2510.15103, October 2025.

Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D. Bagdanov, Shangling Jui, and Joost van de Weijer. Generative Feature Replay For Class-Incremental Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

Noel Loo, Siddharth Swaroop, and Richard E. Turner. Generalized Variational Continual Learning. In *International Conference on Learning Representations*, 2021.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, 2023.

Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Representational Continuity for Unsupervised Continual Learning. In *International Conference on Learning Representations*, 2022.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

James Martens. New Insights and Perspectives on the Natural Gradient Method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Michael S. Matena and Colin A. Raffel. Merging Models with Fisher-Weighted Averaging. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17703–17716, 2022.

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.

Mark D. McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. RanPAC: Random Projections and Pre-trained Models for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An Empirical Investigation of the Role of Pre-training in Lifelong Learning. *Journal of Machine Learning Research*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-Editing Memory in a Transformer. In *International Conference on Learning Representations (ICLR)*, 2023.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the Role of Training Regimes in Continual Learning. In *Advances in Neural Information Processing Systems*, 2020.

Daniel Eugênio Neves, Lucila Ishitani, and Zenilton Kleber Gonçalves do Patrocínio Júnior. Advances and challenges in learning from experience replay. *Artificial Intelligence Review*, 58(2):54, December 2024. ISSN 1573-7462. doi: 10.1007/s10462-024-11062-0.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational Continual Learning. In *International Conference on Learning Representations*, 2018.

Gary W. Oehlert. A Note on the Delta Method. *The American Statistician*, 46(1):27–29, 1992. ISSN 0003-1305. doi: 10.2307/2684406.

Manfred Opper and Ole Winther. A Bayesian approach to on-line learning. In David Saad (ed.), *On-Line Learning in Neural Networks*, Publications of the Newton Institute, pp. 363–378. Cambridge University Press, Cambridge, January 1999. ISBN 978-0-262-19416-7. doi: 10.2277/0521652634.

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E. Turner, and Mohammad Emtiyaz Khan. Continual Deep Learning by Functional Regularisation of Memorable Past. In *Advances in Neural Information Processing Systems*, 2020.

Guiomar Pescador-Barrios, Sarah Lucie Filippi, and Mark van der Wilk. Adjusting Model Size in Continual Gaussian Processes: How Big is Big Enough? In *Forty-Second International Conference on Machine Learning*, June 2025.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, volume 12347, pp. 524–540, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58535-8 978-3-030-58536-5. doi: 10.1007/978-3-030-58536-5_31.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet Dokania, Philip H. S. Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally Budgeted Continual Learning: What Does Matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*, 2021.

Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random Path Selection for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *International Conference on Learning Representations*, 2021.

Rahul Ramesh and Pratik Chaudhari. Model Zoo: A Growing "Brain" That Learns Continually. In *International Conference on Learning Representations*, 2022.

Roger Ratcliff. Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychological Review*, 97(2):285–308, 1990.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

A. Robins. Catastrophic forgetting in neural networks: The role of rehearsal mechanisms. In *Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pp. 65–68, November 1993. doi: 10.1109/ANNES.1993.323080.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv:1606.04671*, 2016.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient Projection Memory for Continual Learning. In *International Conference on Learning Representations*, 2021.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. PRIORITIZED EXPERIENCE REPLAY. In *International Conference on Learning Representations*, 2016.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 2018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*, 2017.

H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, Long Beach, CA, 2017.

Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On Learning the Geodesic Path for Incremental Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. CODA-Prompt: COntinual Decomposed Attention-based Prompting for Rehearsal-Free Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by Distilling Context. arXiv preprint arXiv:2209.15189, 2022.

Peiquan Sun, Wengang Zhou, and Houqiang Li. Attentive Experience Replay. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5900–5907, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i04.6049.

Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. Exploring Example Influence in Continual Learning. In *Advances in Neural Information Processing Systems*, 2022.

Siddharth Swaroop, Cuong V. Nguyen, Thang D. Bui, and Richard E. Turner. Improving and Understanding Variational Continual Learning. arXiv preprint arXiv:1905.02099, May 2019.

Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional Regularisation for Continual Learning with Gaussian Processes. In *International Conference on Learning Representations*, 2020.

Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. GCR: Gradient Coreset Based Replay Buffer Selection For Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Hanna Tseran, Mohammad Emtiyaz Khan, Tatsuya Harada, and Thang D. Bui. Natural variational continual learning. In *Continual Learning Workshop@ NeurIPS*, volume 2, 2018.

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

Tom Veniat, Ludovic Denoyer, and Marc'Aurelio Ranzato. Efficient Continual Learning with Modular Networks and Task-Driven Priors. In *International Conference on Learning Representations*, 2021.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical Decomposition of Prompt-Based Continual Learning: Rethinking Obscured Sub-optimality. In *Advances in Neural Information Processing Systems*, 2023a.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: Rethinking the Knowledge Memory for Lifelong Model Editing of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal Subspace Learning for Language Model Continual Learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023b.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary Prompting for Rehearsal-free Continual Learning. In *European Conference on Computer Vision (ECCV)*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to Prompt for Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022b.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in Superposition. In *Advances in Neural Information Processing Systems*, 2020.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model Soups: Averaging Weights of Multiple Fine-tuned Models Improves Accuracy without Increasing Inference Time. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 23965–23998, 2022.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-LoRA: Scalable Decoupled Low-Rank Adaptation for Class Incremental Learning. In *International Conference on Learning Representations*, 2025.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large Scale Incremental Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation Finetuning for Language Models. In *Advances in Neural Information Processing Systems*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving Interference When Merging Models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Dong Yin, Mehrdad Farajtabar, Ang Li, Nir Levine, and Alex Mott. Optimization and generalization of regularization-based continual learning: A loss approximation viewpoint. *arXiv preprint arXiv:2006.10974*, 2020.

Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations*, 2022.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pp. 3987–3995. PMLR, 2017.

Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. Experience Replay Optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4243–4249, 2019.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. SLCA: Slow Learner with Classifier Alignment for Continual Learning on a Pre-trained Model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19148–19158, 2023.

Xiao Zhang, Dejing Dou, and Ji Wu. Feature forgetting in continual representation learning. *arXiv preprint arXiv:2205.13359*, 2022a.

Xue Zhang, Yunlong Liang, Fandong Meng, Songming Zhang, Yufeng Chen, Jinan Xu, and Jie Zhou. Less, but Better: Efficient Multilingual Expansion for LLMs via Layer-wise Mixture-of-Experts. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17948–17963, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.878.

Yaqian Zhang, Bernhard Pfahringer, Eibe Frank, Albert Bifet, Nick Jin Sean Lim, and Yunzhe Jia. A simple but strong baseline for online continual learning: Repeated Augmented Rehearsal. In *Advances in Neural Information Processing Systems*, 2022b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shutao Xia. Maintaining Discrimination and Fairness in Class Incremental Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-Adapting Language Models. In *Advances in Neural Information Processing Systems*, 2025.