Continual Learning in Transition

Zhiyan Hou[1,6*], Dan Zhang[2*†], Tao Feng[3*†], Liyuan Wang[3], Wei Li[3], Xiangzhao Hao[4], Hongyan An[1], Junfeng Fang[2], Haokai Ma[2], Zhaohui Xu[5], Haiyun Guo[1,6†], Jinqiao Wang[1,6,7], Tat-Seng Chua[2]

[1]Institute of Automation, Chinese Academy of Sciences, [2]National University of Singapore
[3]Tsinghua University, [4]Shanghai Jiao Tong University, [5]Alibaba Group
[6]School of Artificial Intelligence, University of Chinese Academy of Sciences
[7]Wuhan Artificial Intelligence Research Institute

Abstract

Classical continual learning (CL) has primarily focused on enabling models to update and retain knowledge through parameter-centric mechanisms, e.g., training strategies, architectural designs, and weight adaptation. However, emerging paradigms are reshaping the scope of CL beyond this traditional model adaptation view. For instance, on-policy learning broadens the space of update mechanisms; test-time training extends CL from the training phase to inference; and external harness components such as memory, skill libraries, and interaction protocols extend the evolutionary boundaries of model capabilities far beyond the static parameter space. Collectively, these developments indicate a transition from parameter-centric learning toward system-level adaptation. To characterize this transition, we examine the evolution of continual learning through three dimensions: When, How, and Where learning occurs. The How dimension encompasses off-policy, on-policy, and beyond-gradient optimization mechanics. The When dimension captures evolution across pre-training, post-training, and inference-time stages. The Where dimension delineates updates occurring within internal parameters versus external structural constraints. Anchored by this tri-axial framework, we systematically survey representative methods, trace the ongoing transition of continual learning, and discuss the key challenges, broader implications, and future directions arising from this paradigm shift.

1 Introduction

Recent advances in large language models (LLMs) and Agentic AI [1–7], including substantial progress on complex reasoning [8, 9], long-horizon task execution [5, 7], and code generation [10, 11], are widely regarded as a substantive step toward artificial general intelligence (AGI). A genuinely general intelligence, however, must operate in open environments where the state of the world continuously evolves and task demands are constantly renewed. This requires a system to update knowledge as the environment changes [12], accumulate and recombine skills through interaction [13], adjust behavioral policies in light of feedback [14], and consolidate long-term memory across sessions and tasks [15, 16]. Therefore, the path toward AGI lies not merely in training stronger static models, but in constructing agent systems that can continually improve after deployment [17]. *How to remain adaptive under distribution shift, persistently accumulate experience, and resist forgetting* is precisely what continual learning seeks to address [18–20].

Classical continual learning (CL) has long centered on catastrophic forgetting and the stability–plasticity trade-off, with mainstream methods broadly grouped into four categories: *replay-based* methods, *gradient-based* methods, *parameter-isolation or architecture-expansion* methods, and *regularization-based* methods [20, 21] (Figure 1). With the rapid development of LLMs and agentic systems, however, the methodological

*Equal contribution. †Corresponding authors.

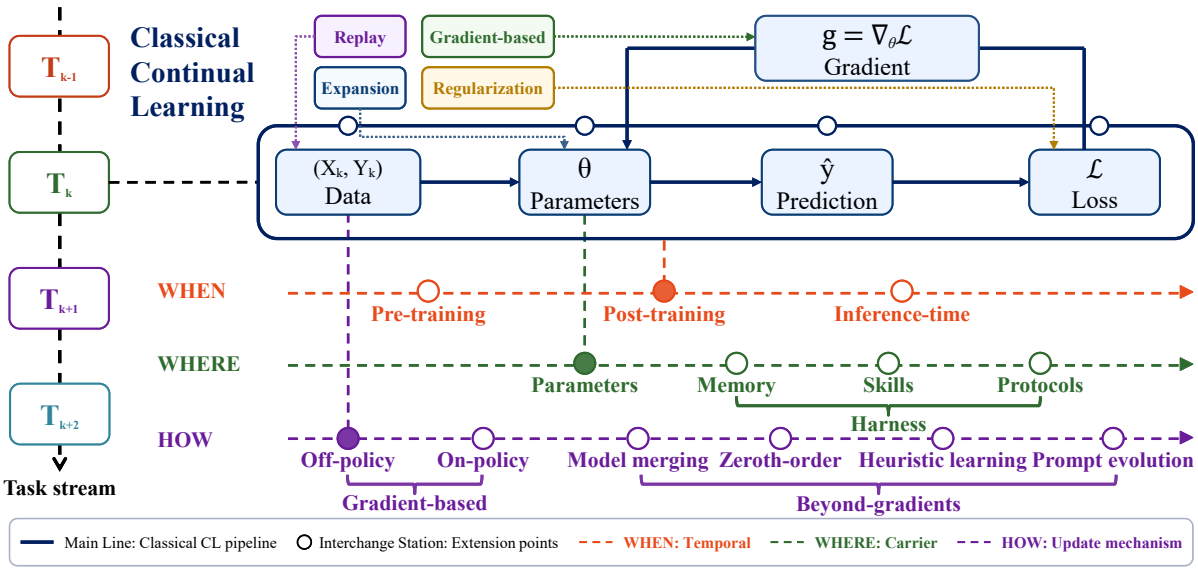

Figure 1 Overview of the three-dimensional view of continual learning developed in this survey. The main line (solid) depicts the classical continual learning pipeline, in which a stream of tasks is fitted by back-propagated gradient updates to model parameters, and circles mark the points at which classical method families intervene. The three dashed lines unfold the extensions examined in this survey: the *When* line spans pre-training, post-training, and inference time; the *Where* line spans parameters and the harness components (memory, skills, and protocols); and the *How* line spans gradient-based updates (off-policy, on-policy) and beyond-gradient updates. Filled circles mark the position of the classical setting on each line.

landscape and application scenarios of CL are expanding rapidly, and the classical taxonomy can no longer fully capture these emerging developments.

The research landscape of CL is undergoing simultaneous changes along three interrelated dimensions (Figure 1). With respect to the *learning mechanism* (the *How* line in Figure 1), the change unfolds at two levels. Within *gradient-based learning*, classical off-policy updates tend to cause the model to deviate substantially from its initial behavior distribution and thereby induce pronounced catastrophic forgetting, whereas on-policy paradigms, including reinforcement learning from human feedback (RLHF) [22] and reinforcement learning with verifiable rewards (RLVR) [23], exhibit notable advantages in mitigating forgetting [24]. This observation has subsequently inspired a line of on-policy post-training methods exemplified by On-Policy Distillation (OPD) [25] and, more recently, on-policy self-distillation (OPSD) for continual learning [26]. *Beyond gradients*, approaches that do not rely on standard backpropagation, such as model merging [27], zeroth-order optimization [28], heuristic learning [29], and prompt evolution [30], have further broadened the design space of update mechanisms. With respect to the *learning timing* (the *When* line in Figure 1), continual adaptation now spans the full model lifecycle, from continual pre-training and multi-stage post-training to the post-deployment inference loop. Test-Time Adaptation (TTA) [31] and Test-Time Training (TTT) [32] introduce writable state or parameter updates at inference time, enabling models to perform on-the-fly adjustments when confronted with novel samples and distribution shifts. With respect to the *locus of capability* (the *Where* line in Figure 1), the research focus has migrated from inside the model to the external harness layer that surrounds it [33]: stateless retrieval-augmented generation (RAG) has evolved into long-term memory systems with read–write semantics that persist across sessions and tasks [15]; fixed tool sets have evolved into self-generated, reusable, and composable skill libraries [13]; and static interaction protocols and behavioral rules have evolved into adaptive protocols that are continually refined through reflection and feedback [14]. Capability accumulation and updates have thereby extended from the parameter space to multiple carriers situated outside the model.

These developments collectively indicate that the research landscape of continual learning is undergoing concurrent changes along three dimensions: learning mechanism, learning timing, and locus of capability. The

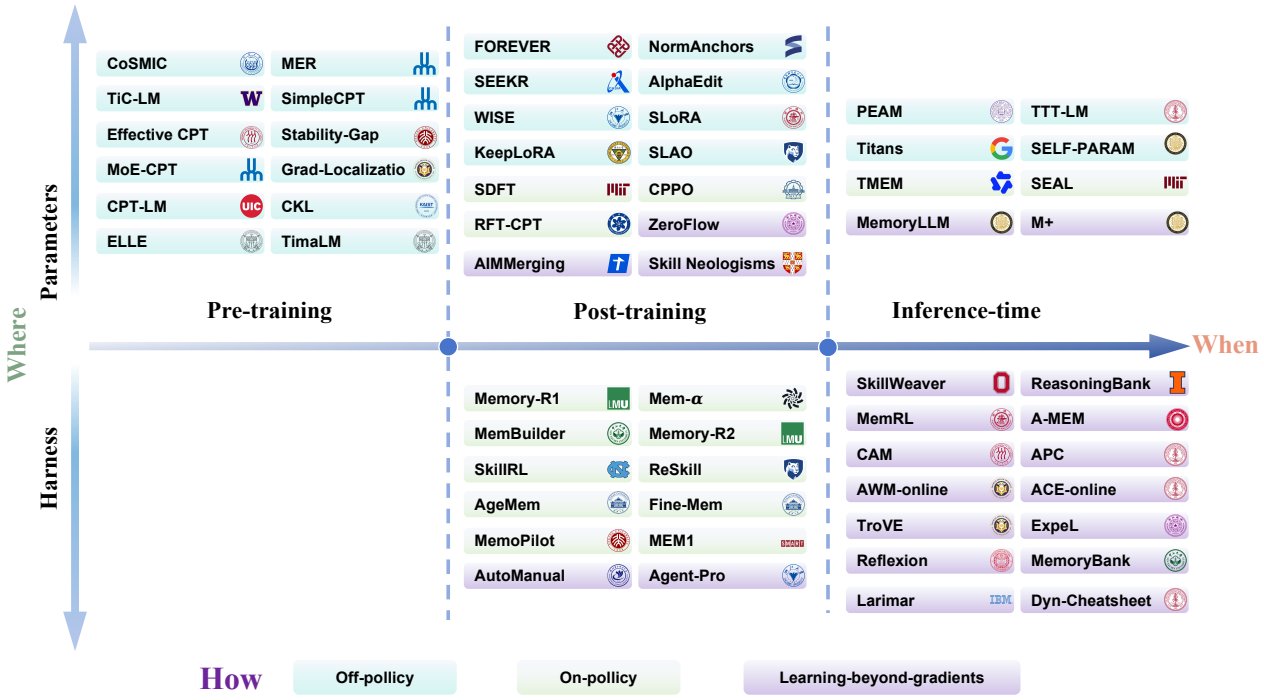

Figure 2 The three dimensions of continual learning with representative methods. Methods are placed by When capability evolution occurs, Where capability is accumulated, and How the update is driven.

classical taxonomy, primarily organized around the canonical setting of training-time, parameter-level, off-policy gradient updates, can no longer naturally accommodate these emerging directions. Motivated by this observation, we reformulate continual learning in the era of large language models and agentic AI as *continual capability evolution* across mechanism, timing, and locus, and propose a three-axis taxonomy that uniformly characterizes this process (Figure 2). The *How* axis characterizes the extension of update mechanisms from off-policy to on-policy and further to learning-beyond-gradients; the *When* axis characterizes the extension of capability evolution across the lifecycle from pre-training and post-training to inference-time; and the *Where* axis characterizes the extension of capability locus from parameters to the harness.

The main contributions of this survey are threefold.

- We recast continual learning in the era of large language models and agentic AI as *continual capability evolution*, viewed through three complementary questions: when capability evolves, where it is carried, and how it is updated. In contrast to existing LLM continual-learning surveys, which largely extend the classical parameter-centric categorization [20, 21], this view brings the agent harness (memory, skills, and protocols) together with inference-time and gradient-free mechanisms into the same frame as parameter-level learning.
- Guided by this view, we review representative methods that are seldom examined together, spanning continual pre- and post-training, test-time training, reinforcement-learning-based alignment, model merging, memory systems, skill libraries, and prompt evolution, and locate each in a common (when, where, how) space that makes their otherwise implicit relationships explicit.
- Reading this space as a whole, we characterize where methods concentrate and why, and identify the sparse and empty regions that mark concrete challenges and opportunities for continual learning in self-evolving agent systems.

2 Revisiting Continual Learning

Section 1 motivated the need to revisit the scope of continual learning in light of recent developments beyond the classical setting. Before the three dimensions can be developed, however, it is necessary to make explicit what the classical formulation held fixed. This section therefore revisits classical continual learning, including its formulation, objectives, and major method families (Section 2.1), and shows that they share three implicit assumptions. Section 2.2 then turns these assumptions into the three questions of *when*, *where*, and *how*, under which classical continual learning can be viewed as a particular point in a broader space.

2.1 Classical Continual Learning and Its Three Implicit Assumptions

Classical continual learning studies a model that learns under a non-stationary data distribution. Training data arrive as a sequence of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_K$, each associated with its own distribution $\mathcal{D}_k := p(X_k, Y_k)$ and loss $\mathcal{L}_k$ [21]. At step k, the model θ is updated while historical data $\{\mathcal{D}_i\}_{i<k}$ are inaccessible or available only in a restricted form. The ideal objective is to fit the current task while keeping the aggregate expected loss $\sum_{i\leq k} \mathbb{E}_{(X,Y)\sim\mathcal{D}_i}[\mathcal{L}_i]$ over all observed tasks low, despite the fact that earlier data are no longer fully available. The central obstacle is *catastrophic forgetting* [18, 34]: updates that improve performance on $\mathcal{T}_k$ may perturb the representations or decision boundaries that support earlier tasks, sharply degrading their performance.

Classical methods are commonly evaluated against two coupled desiderata. The first is the *plasticity and stability* trade-off: the model must remain plastic enough to acquire new tasks while maintaining stable knowledge of previous ones. From a Bayesian perspective, continual learning can be viewed as sequential posterior updating, where the posterior induced by previous tasks becomes the prior for the next task. This view motivates regularization-based formulations that approximate the preservation of previous knowledge by penalizing changes to parameters important for earlier tasks [20]. The second desideratum is *intra-task and inter-task generalizability*: the learner should remain robust to the train/test gap within each task while adapting to distribution shifts across tasks, a requirement that continual reinforcement learning frames as retaining and transferring behavior across non-stationary environments [19].

Around these desiderata, the literature has consolidated several major families of methods. *Replay-based* methods mitigate forgetting by approximating access to past experience, either by retaining examples in a memory buffer with designed sample-selection rules [35, 36], generating pseudo-samples through generative replay, or storing intermediate representations through feature replay. *Gradient-based* methods modify the optimization process itself [37], projecting updates onto directions that reduce interference with earlier tasks (*e.g.,* Gradient Episodic Memory (GEM) [38], Orthogonal Gradient Descent (OGD) [39]), and C-Flat series [40, 41]. *Architecture-based* methods [42, 43] reduce interference by isolating or expanding capacity, allocating task-specific subspaces or modules through pruning, hard attention masks, or dynamic growth (*e.g.,* PackNet [44], Hard Attention to the Task (HAT) [45], and progressive networks [46]). *Regularization-based* methods preserve prior knowledge by constraining updates [47], either through parameter-importance penalties based on Fisher information or related sensitivity measures (*e.g.,* Elastic Weight Consolidation (EWC) [48], Synaptic Intelligence (SI) [49], and Memory Aware Synapses (MAS) [50]) or through functional regularization that distills the behavior of previous models into the current one (*e.g.,* Learning without Forgetting (LwF) [51]).

Despite their distinct technical mechanisms, these method families largely operate within a common formulation that is often left implicit. Three assumptions are particularly important: (i) capability acquisition and retention primarily occur during a dedicated *training* stage before deployment; (ii) accumulated capability is primarily carried by model *parameters*; and (iii) updates are typically implemented through *gradient*-based optimization over externally supplied or previously collected data. These assumptions are not intrinsic to the broader goal of continual learning; rather, they reflect the design choices of the setting in which the field originally developed. Together, they delineate the classical formulation and reveal the directions along which recent work has begun to extend it.

2.2 From Three Assumptions to Three Questions

Each of the three assumptions is, in effect, a fixed answer to a question that any continual learning system must implicitly settle. Assumption (i) fixes the answer to *when* learning happens: capability evolution is confined to a dedicated training stage. Assumption (ii) fixes the answer to *where* capability resides: it is carried by model parameters. Assumption (iii) fixes the answer to *how* the update is driven: by gradient-based optimization over externally supplied data. In the classical setting these answers were so uniform that the questions themselves were rarely asked. The developments reviewed in Section 1 make each answer a genuine variable: learning now happens at several stages of the model lifecycle, capability accumulates on carriers beyond parameters, and updates are driven by mechanisms beyond off-policy gradients. We therefore recast continual learning from sequential parameter learning to continual capability evolution, characterized through the When, Where, and How questions. The three dimensions are used as complementary analytical perspectives rather than as mutually exclusive categories. A method may be associated with more than one applicable label, and its characterization may change over time when capability moves between lifecycle stages or carriers. We therefore describe a method using a When–Where–How profile rather than treating it as occupying one immutable point in a strict Cartesian grid.

Within the taxonomy adopted in this survey, classical continual learning remains the reference setting of post-training in When, model parameters in Where, and off-policy gradient updates in How. The developments of the LLM era extend this reference setting: When spans the full model lifecycle, including continual pre-training and inference-time adaptation; Where extends from parameters to the harness; and How extends from off-policy gradient learning to on-policy learning and learning beyond gradients. Section 3 examines these three dimensions in turn.

3 The Three Dimensions of Continual Learning

3.1 Overview

We examine every continual learning method by asking three questions: when capability evolution takes place, where the acquired capability is accumulated, and how the update is driven.

The three axes are intended as complementary perspectives rather than mutually exclusive categories. A continual learning method can be examined from all three at the same time, since its learning stage, capability carrier, and update mechanism jointly determine how continual capability evolution is realized. From this view, classical continual learning corresponds to a canonical setting, namely post-training updates to model parameters through off-policy gradient learning, while recent large-model and agent-based methods depart from this setting along one or more dimensions.

This perspective helps clarify that continual learning is no longer confined to the classical problem of preventing forgetting during sequential parameter training. Instead, it increasingly concerns continual capability evolution across the full lifecycle of large models and agent systems. Sections 3.2 through 3.4 examine the three perspectives in turn (Figure 3), and Section 3.5 places existing methods back into the joint space of the three axes and analyzes how they combine and populate it.

3.2 When: Capability Evolution across the Model Lifecycle

The *When* axis asks at which stage of the model lifecycle continual capability evolution is realized. In the era of large models, this question becomes especially important because learning is no longer confined to a single sequential training process (Figure 4). Instead, capability evolution may occur during continued pre-training over evolving corpora, during post-training through instruction tuning, preference optimization, or reinforcement learning (RL), and after deployment through inference-time adaptation to incoming inputs and environmental feedback.

This temporal shift changes not only the location of continual learning, but also the form of the problem itself. Different lifecycle stages expose different update targets, feedback signals, and stability requirements. Continued pre-training emphasizes knowledge renewal under changing data distributions. Post-training emphasizes capability alignment, task adaptation, and forgetting control across successive optimization stages.

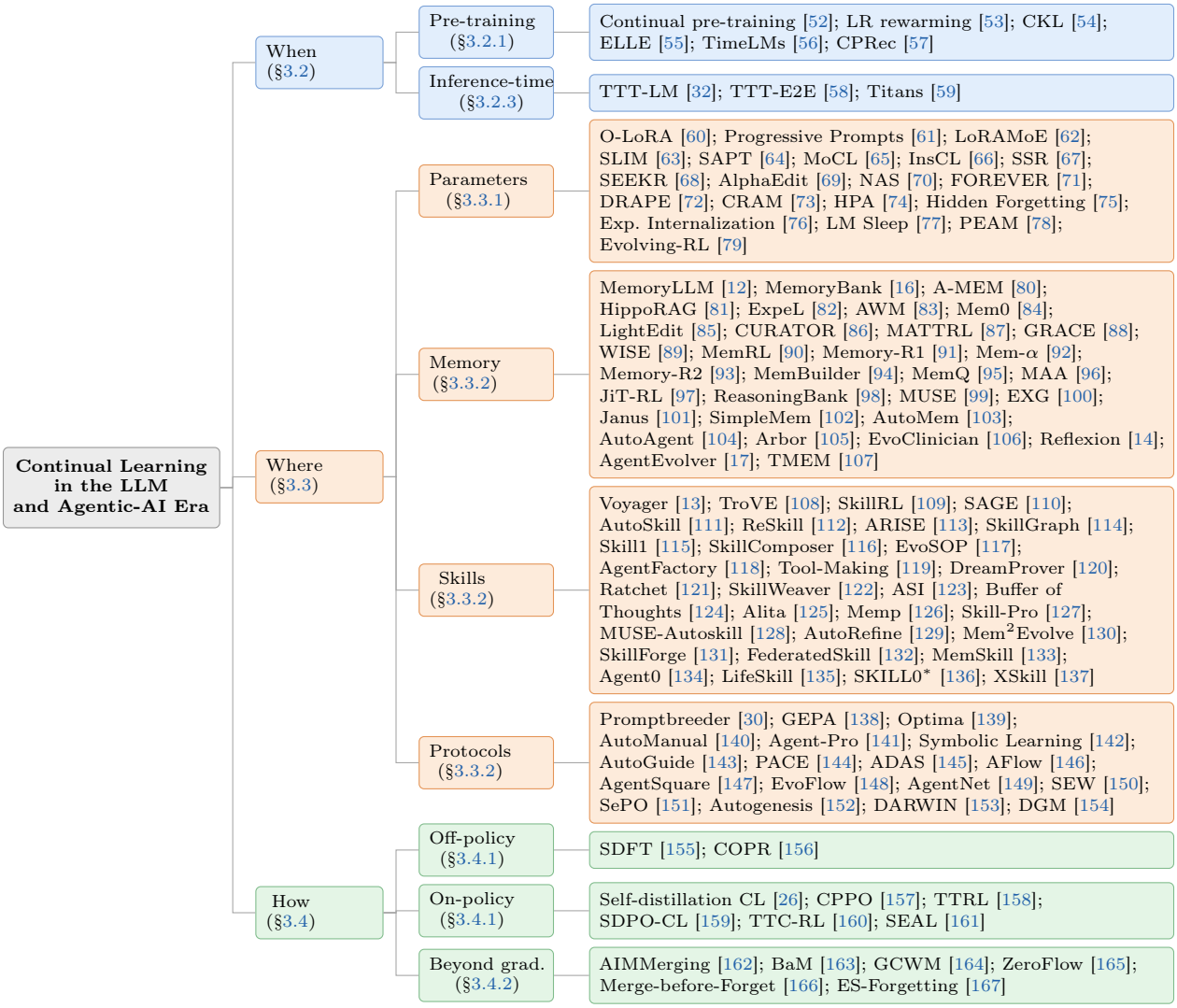

Figure 3 Representative LLM-era and agentic-AI-era continual-learning methods organized along the three dimensions. Methods that span several dimensions are placed once, under their most salient carrier or mechanism.

Inference-time adaptation emphasizes rapid adjustment under test-time distribution shift, often with limited writable state or lightweight parameter updates. The *When* axis therefore repositions continual learning as a lifecycle-level problem of how large models and agent systems keep evolving over time.

3.2.1 Pre-Training

Continual pretraining turns pretraining itself from a one-shot procedure into a multi-stage process over evolving corpora, so that a model already trained on a base distribution can be further adapted to new domains, languages, or temporal slices without restarting from scratch. The direction has roots in domain- and task-adaptive pretraining, which showed that a further phase of pretraining on domain or task corpora consistently improves downstream performance [168], and in continual knowledge learning, which formulates keeping a language model's world knowledge current as renewing outdated facts while retaining time-invariant ones [54]. Ibrahim *et al.* [52] establish a simple yet scalable recipe that combines learning-rate rewarming with a small replay buffer of base-distribution tokens, allowing 405M and 10B models to absorb new corpora while keeping degradation on the original distribution close to negligible. Yıldız *et al.* [169] complement this with a systematic study of how the magnitude and ordering of domain shifts modulate forgetting and forward

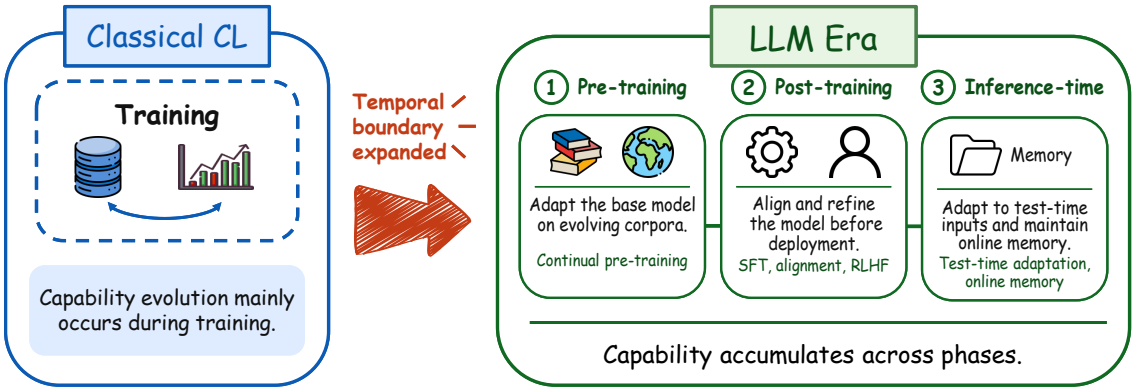

Figure 4 The *When* dimension. Classical continual learning confines capability evolution to a single training stage (left). In the LLM-era, it spans pre-training, post-training, and inference time, so that capability accumulates across the phases of the model lifecycle (right).

transfer at LLM scale, building on earlier empirical evidence on warm-up schedules from [53]. Relative to the classical setting in which task boundaries are explicit and supervised losses are reused, continual pretraining operates over unsupervised next-token objectives at scales where individual task labels are absent, shifting the unit of incremental adaptation from a labeled task to a corpus stream.

3.2.2 Post-Training

After pre-training, post-training has evolved from a single instruction-tuning step into a longitudinal pipeline: supervised fine-tuning is typically followed by one or more preference- or reward-alignment stages (*e.g.,* RLHF, RLVR), and deployed models continue to be revised in successive versions. Each round of updating risks eroding what earlier rounds established, which makes this stage a continual-learning problem in its own right.

Recent work characterizes this problem empirically. Luo *et al.* [170] quantify catastrophic forgetting during continual fine-tuning of LLMs, showing that domain knowledge, reasoning, and reading comprehension all degrade as instruction tuning proceeds, with the effect intensifying with model scale in the 1B–7B range. TRACE [171] consolidates this observation into a benchmark for aligned LLMs, demonstrating that sequential training on new tasks erodes not only general ability but also instruction following and safety alignment. Zheng *et al.* [172] refine the diagnosis by showing that part of the measured degradation is *spurious* forgetting, a recoverable loss of task alignment rather than a loss of underlying knowledge, and that freezing bottom layers largely prevents it. The alignment stages contribute their own failure mode: preference optimization imposes an alignment tax on upstream capabilities, which InstructGPT already mitigated by mixing pre-training gradients into RLHF [22], and the choice of update mechanism strongly modulates how much a given stage forgets [24]. Together, these findings establish the post-training pipeline itself, rather than any single fine-tuning step, as the unit at which forgetting must be measured and controlled [173].

3.2.3 Inference-Time

The *inference-time* position on the *When* axis refers to capability evolution that occurs after deployment. The key distinction is persistence. General inference-time computation, such as search, sampling, self-consistency, or other test-time scaling strategies, can improve the answer to a single query, but it usually leaves the system unchanged once the query is completed. Inference-time continual learning, in contrast, writes information obtained during inference back into the system, through writable states or parameter updates, so that earlier inputs, feedback signals, or self-generated supervision can influence later behavior.

This setting is especially relevant for large models and agent systems. During deployment, a model may face long input streams, shifting user requirements, evolving task distributions, and feedback signals that are unavailable during training. Treating inference as a purely read-only process limits the system to static deployment. Inference-time continual learning instead turns deployment into an adaptive process, where the

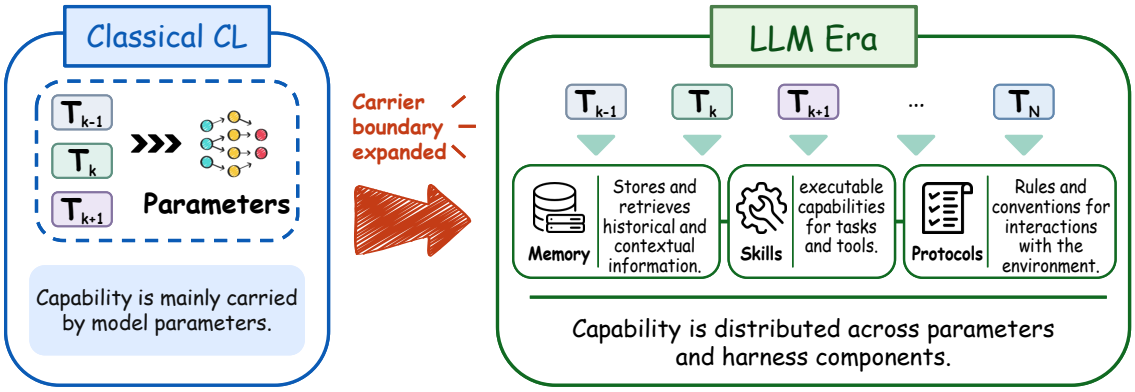

Figure 5 The *Where* dimension. Classical continual learning carries capability almost exclusively in model parameters (left); in the LLM era, capability is distributed across parameters and the harness components, namely memory, skills, and protocols (right).

model or its surrounding system can accumulate information from the test stream and adjust its behavior over time.

Recent methods instantiate this idea in different ways. TTT-LM [32] updates trainable hidden-state modules within a test sequence, allowing contextual information from earlier tokens to be compressed into fast weights. TTT-E2E [58] extends this direction to long-context language modeling by updating model components through next-token prediction at inference time, thereby using parameter updates to absorb information from the growing context. TTRL [158] further introduces reward-driven test-time learning, deriving supervision from the model's own sampled outputs on unlabeled test inputs and using it to improve performance on the test distribution.

From the perspective of the *When* axis, these methods mark a shift from static deployment to post-deployment capability evolution. They expand continual learning beyond pre-deployment optimization and show that adaptation can also occur during the actual use of a model, as long as inference-time signals are persistently accumulated and affect subsequent predictions.

3.3 Where: Capability Carriers from Parameters to the Harness

The *Where* axis asks where acquired knowledge and capabilities are stored and updated. Model parameters remain the canonical carrier of continual learning, although modern systems may update them through full fine-tuning, parameter-efficient modules, or other structured forms. The more fundamental shift in the LLM era, however, is that capability accumulation increasingly extends beyond the parameter space (Figure 5). Agent systems externalize part of their evolving capability into the harness layer surrounding the core model, including memory, skills, and protocols. These external carriers are explicit, editable, retrievable, and callable, allowing continual learning to operate not only through weight updates, but also through persistent non-parametric objects that can be accumulated and revised over time.

3.3.1 Parametric Carrier

Parameters remain the canonical carrier of continual learning: new capabilities are consolidated by modifying trainable weights, and forgetting is often studied as interference within the shared parameter space. In the LLM era, however, the main change lies in the granularity of parametric updates. Full-parameter fine-tuning is often costly and prone to interference at scale, so many continual adaptation methods restrict updates to parameter-efficient or task-specific modules. This follows the motivation of reducing cross-task interference under LLM-scale constraints, while avoiding holistic modification of the entire model.

The low-rank adaptation (LoRA) [174] family exemplifies this direction. O-LoRA [60] constrains the LoRA directions of different tasks to be mutually orthogonal during continual fine-tuning, thereby reducing over-writing between task-specific subspaces. Similar ideas also appear across the broader parameter-efficient

fine-tuning (PEFT) family. Progressive Prompts [61] learns a separate soft prompt for each task and concatenates it with previously learned prompts, confining adaptation to separable prompt parameters. Expert-based approaches such as LoRAMoE [62] route different inputs or tasks to different LoRA experts, while SLIM [63] combines LoRA experts with an identity pathway to support new-task learning with reduced forgetting.

These methods refine the parametric carrier from a monolithic parameter space into a set of modular, locally updated components. Nevertheless, the acquired capability is still stored in trainable parameters inside or attached to the model. The more substantial expansion of the *Where* axis begins when capability accumulation moves beyond parametric objects to the external harness layer.

3.3.2 The Harness Layer

With the rapid development of LLM agent systems, the carrier of knowledge and capability is no longer confined to model parameters but extends further to the harness layer surrounding the core model [175], namely a collection of objects external to the parameters that are explicitly constructed and edited by the system designer. Following [33], the harness divides, by the nature of what it carries, into three classes: *memory* is information storage that is readable and writable across inference calls (such as dialogue history, long-term memory banks, and retrieved document fragments); *skills* are executable units that the model actively invokes (such as external tools, function modules, and callable sub-agents); and *protocols* are rules and formats that govern how the model interacts with its environment (such as system prompts, rule files, and message-format schemas). All three classes can be read, extended, or rearranged without updating the backbone weights, supporting continual learning outside the parameter tensor. In the LLM era, each has given rise to a relatively mature methodological lineage, covering hierarchical designs of memory systems, externalized extension of tool repertoires and skill libraries, and the evolution of prompt-, rule-, and message-format protocols, respectively.

Memory. As the harness component that carries historical and contextual information, memory provides LLM systems with a pathway for capability accumulation that does not depend on gradient updates: the model writes facts, experience, and intermediate states into external storage and retrieves them in subsequent inference, thereby continually expanding the body of knowledge it can access and preserving historical context without modifying backbone weights. Recent work pushes memory as a continual-learning carrier along several distinct design dimensions. *MemoryLLM* [12] equips the model with a fixed-size learnable memory pool into which the model itself integrates new knowledge during inference, turning memory into a self-rewritable resource that the model actively manages and enabling continual knowledge update without backbone updates. *MemoryBank* [16] consolidates long-term memory for dialogue agents through an Ebbinghaus-style forgetting curve that strengthens important information across interactions while gradually fading redundant content, making the stability–plasticity trade-off of continual learning explicit at the memory-consolidation layer. *A-MEM* [80] takes atomic propositions as the smallest unit of reading, writing, and forgetting, reducing the granularity of memory updates to the fact level and avoiding the collateral forgetting that paragraph-level overwrites induce. *MemRL* [90] extends the memory-update mechanism from rule-based retrieval to reward-driven RL self-evolution. Collectively these methods turn memory from a passive context container into a writable, structurable, self-managed capability carrier in which capability evolution proceeds outside the parameter tensor and accumulates across tasks.

Skills. As the harness component that carries procedural capability, skills encapsulate the model's reusable execution flows as externally callable units. By adding, refining, and composing skill units, an agent extends its action repertoire without updating backbone weights, and continual learning proceeds as skill accumulation at the execution layer. Representative methods advance the skill library as a continual-learning carrier along different design dimensions. *Voyager* [13] lets an LLM agent write code as reusable skills while interacting with its environment and store them in a persistent skill library, so that the agent continually masters new tasks through progressive expansion of the library while the discreteness of skills keeps new and old tasks from interfering. *SkillRL* [109] jointly optimizes the skill library and the policy via reinforcement learning, extending skill updates from rule-based curation to reward-driven iterative refinement so that the growth of the library is also guided by task utility. *SAGE* [110] couples the skill library with reinforcement learning to build a self-improving agent that acquires, stores, and reuses skills under reward feedback, so that the

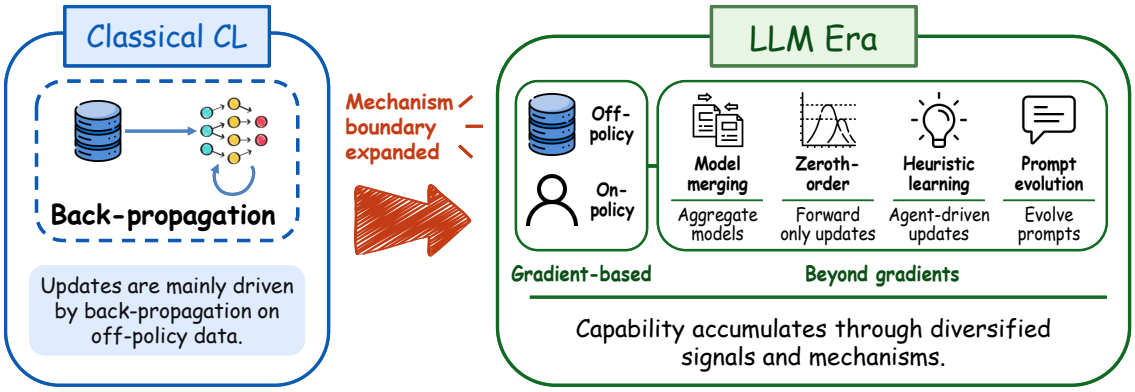

Figure 6 The *How* dimension. Classical continual learning relies on back-propagation over off-policy data (left); the LLM era adds on-policy learning within the gradient-based regime, and beyond-gradient mechanisms such as model merging, zeroth-order optimization, heuristic learning, and prompt evolution (right).

library grows with the agent's own experience. Beyond these methods that take the skill library as the update target, *SKILL0* [136] further illustrates a reverse migration between skills and the parametric carrier by first accumulating skills on the harness via RL and then internalizing them into model parameters via curriculum (discussed in Section 3.5). Together these methods turn the skill library from a static toolkit into an extensible, optimizable carrier that can also migrate to parameters, enabling the agent's capability repertoire to accumulate continuously across tasks.

Protocols. As the harness component that carries interaction structure, protocols externalize the behavioral rules and communication conventions between the model and its environment as editable artifacts. By revising and self-evolving these rules, an agent continually adjusts its behavioral framework without updating backbone weights, allowing continual learning to proceed in the form of protocol evolution at the interaction layer. *Promptbreeder* [30] uses an LLM to evolve prompts themselves through self-referential mutation, making the system prompt a carrier that task feedback continually rewrites; it is the most explicit representative of continual learning via protocol evolution. Rule files follow the same externalization approach: engineering practices such as Cline, Aider, and Cursor externalize an agent's behavioral rules as files that developers iteratively revise as project requirements evolve, treating the interaction structure as a continually editable carrier. Compared with the denser lineages already formed for memory and skills, the research landscape of protocols as a continual-learning carrier remains at an early stage, but it is a necessary conceptual anchor among the three classes of externalization that the harness comprises.

3.4 How: Update Mechanisms from Off-Policy Gradients to Gradient-Free Learning

The LLM era extends the *How* axis in two directions (Figure 6). First, within gradient-based learning, it broadens the data source from off-policy data decoupled from the current policy (historical replay, static corpora, distillation targets) to on-policy rollouts generated by the policy that is currently being updated under reward signals (Section 3.4.1). Second, it moves beyond backpropagated gradients altogether, accumulating capability through mechanisms that never compute an analytic gradient of a loss (Section 3.4.2). What unifies the continual-learning relevance of these regimes is how each controls drift away from prior capability: the central question along the *How* axis is not merely how an update is computed, but how strongly it perturbs what the model already knows.

3.4.1 Update Regime: Off-Policy vs. On-Policy

The off-policy methods in classical continual learning are dominated by replay over historical samples and distillation against the outputs of an older model, both relying on externally supplied signal sources. The continual-learning question for off-policy updates is how to fine-tune on new data without drifting away from previously acquired behavior. Self-distillation offers a direct LLM-era answer: SDFT [155] prompts the model to rewrite the task data into responses that match its own distribution and then fine-tunes on this

self-generated, distribution-aligned data, thereby bridging the distribution gap between the task data and the base model and empirically mitigating catastrophic forgetting of general capabilities and safety alignment. Off-policy learning thus controls drift by keeping the training distribution close to the model's own.

The genuinely new mechanism that the LLM era brings to the *How* axis is on-policy learning, where training data is generated by the very policy being updated, conditioned on reward signals; the reward-driven alignment paradigms of post-training (*e.g.,* RLHF, RLVR) are its principal instances. Its significance for continual learning is structural rather than incidental: Shenfeld *et al.* [24] show that the degree of forgetting depends on the Kullback–Leibler (KL) divergence between the fine-tuned and the reference policy on the new task, and that on-policy RL implicitly prefers, among the many solutions to a new task, the one with smallest KL, so its drift away from previously acquired capabilities stays small—whereas off-policy supervised fine-tuning can converge to distributions arbitrarily far from the reference model. On-policy learning is therefore not merely an additional mechanism but one structurally aligned with the central concern of continual learning, avoiding forgetting; at inference time the same principle is carried into deployment by TTRL [158], which drives test-time updates with an on-policy reward derived from the model's own majority vote.

3.4.2 Learning Beyond Gradients

Beyond the gradient-based regimes lies a family of mechanisms that accumulate capability without ever backpropagating an analytic gradient of a loss. The most established member is model merging, which composes capability directly in weight space among already-trained tensors. The pioneering Task Arithmetic [176] defines task vectors as the differences between task-specific fine-tuned weights and the base weights and obtains a multi-task model by arithmetic combination of these vectors; TIES-Merging [177] prunes small-magnitude changes and aligns sign-consistent directions to reduce conflicts, Evolutionary Model Merge [178] searches the parameter and data-flow spaces for an optimal recipe, and further methods such as DARE [179] and Twin-Merging [180] refine coefficient control, aggregation granularity, and conflict mitigation. From a continual-learning standpoint, model merging accumulates multi-task capability with zero access to historical training data, directly addressing the no-historical-data constraint of classical continual learning; recent work such as AIMMerging [162] connects it explicitly to the continual-learning problem for language models, and a dedicated survey [27] reviews the area, which this survey reads as the canonical gradient-free mechanism along the *How* axis.

A second, near-gradient member is forward-only or zeroth-order optimization, which updates weights by estimating a descent direction from forward passes alone, without backpropagation. MeZO [28] established that a memory-efficient zeroth-order estimator can fine-tune large models with only forward evaluations, and ZeroFlow [165] showed that such forward-pass optimization is an effective continual learner: across forgetting benchmarks the implicit smoothing of zeroth-order updates matches or exceeds first-order fine-tuning at mitigating catastrophic forgetting—an intrinsic stability property paralleling the small-KL behavior of on-policy RL. Beyond weight space, the same gradient-free principle operates on the harness, where memory writes, skill accumulation, and prompt evolution update capability without backpropagation—a paradigm recently articulated as *heuristic learning*, in which an agent revises an external system from feedback rather than gradient signals [29]. Because for those methods the defining departure is the externalized carrier rather than the mechanism, they are detailed under the *Where* axis (Section 3.3); here it suffices to note that learning beyond gradients spans a spectrum, from zeroth-order optimization that still estimates a gradient, through gradient-free weight composition, to harness edits that leave the parameter space entirely.

3.5 Cross-Dimensional Method Profiles

Sections 3.2–3.4 examine the literature one dimension at a time. Figure 2 presents representative methods under the When, Where, and How perspectives, but an occurrence in one panel does not by itself encode a method's complete profile. Methods may appear in several panels when several dimensions are relevant. Appendix C records the corresponding cross-dimensional profiles in Table C.1.

Several methods illustrate why the three dimensions should be read together. Reflexion [14] combines inference-time evolution, a harness-based memory carrier, and learning beyond gradients through reflective text generation and direct memory writing. TTRL [158] combines inference-time parameter adaptation with

on-policy learning. AgentEvolver [17] spans parameters and the harness: on-policy reinforcement learning updates the policy parameters, while selected trajectories are accumulated in experience memory. These methods are therefore described by combinations of labels rather than by their position in only one branch of Figure 3.

A method's profile may also change over time. SKILL0 [136] first accumulates reusable capability in a harness-based skill library and subsequently internalizes selected skills into model parameters through a curriculum. It is therefore represented as a Harness (Skills)-to-Parameters trajectory along the Where dimension rather than as a method with one fixed carrier.

Figures 2 contains selected representative methods and do not result from an exhaustive literature-enumeration protocol. Consequently, the number of methods displayed in a region should not be interpreted as a statistical estimate of research density, nor should an unoccupied region be treated as evidence that no relevant work exists. The figures instead illustrate recurring combinations and motivate comparatively underexplored questions for future research.

4 Future Outlook and Discussion

This survey has reviewed how continual learning in the era of large models has spread well beyond its classical setting: capability is now updated at different points in the model lifecycle, carried by parameters as well as by external memory, skills, and protocols, and driven by a widening range of update mechanisms. Building on that overall picture, rather than on any single method, we step back and discuss several open trends together with the questions they raise for future work. Our central observation is that the field is shifting from an isolated algorithmic problem, namely the suppression of catastrophic forgetting in a single model, toward a broader question of how capability is organized across a whole system: where it should live, when it should be updated, and how it should be combined, retained, and released. We organize the discussion around five themes: the structural ceilings that limit how far capability can be pushed into ever longer context (Section 4.1); whether the hand-tuned engineering that currently manages these carriers already amounts to a learning mechanism (Section 4.2); how a model and its harness might instead evolve in a coordinated way (Section 4.3); why long-horizon agents, rather than static benchmarks, form the real testbed for such directions (Section 4.4); and what all of this suggests for the path toward artificial general intelligence (AGI) and for research priorities (Section 4.5).

4.1 From Longer Context to Persistent Capability

A defining change of the LLM era is that the carrier of capability is no longer confined to the parameter tensor but migrates outward, from parameters to external memory and then to the context window. Understanding the capacity, compressibility, and decay of each layer is the basis for the discussions that follow.

Three layers of carriers. The innermost layer is the *parameters*, whose capacity scales with model size and whose update cost is the highest, since it requires backpropagation and alignment data. Once written, however, parameters offer the most stable recall and represent capability that is genuinely internalized. The middle layer is *external memory*, including vector stores, key-value caches, atomic memory items [80], and self-managed memory pools. This layer accumulates facts and experience without parameter updates, at low write cost and with high interpretability, but every use must pass through retrieval and its capacity is bounded by storage and retrieval bandwidth. The outermost layer is the *context window*, which holds the system prompt, in-context examples, and tool-call history. It is the fastest channel accessible within a single inference pass, yet also the most volatile, because the window has a hard length limit and overly long contexts dilute attention. Across the three layers a clear trend emerges: as capability moves outward, write cost falls and interpretability rises, while recall stability and capacity ceilings fall in step.

From ICL to TTT: the structural ceiling of context. In-context learning, which operates entirely through context with parameters fixed, and test-time training [32], which updates part of the parameters at inference time, represent two ways of acquiring capability during inference. The migration from the former toward the latter is not incidental: it exposes the structural limit of a context-only route, in that information is discarded once it exceeds the window and any compression is necessarily lossy. This limit takes the form of two ceilings:

a hard ceiling on length, since even windows extended to a million tokens or more remain bounded, and a soft ceiling on effective attention, since the well-documented needle-in-a-haystack phenomenon shows that usable attention over long contexts is unevenly distributed by position and effective information density does not grow linearly with window length. A practical observation reinforces this point: distilling rich tacit experience into a textual summary is brittle outside text-centric domains such as software engineering, where a compaction step can silently reverse a hard-won optimization because the rationale behind it never entered the summary [181]. Consequently, as task complexity grows and reasoning chains lengthen, context alone cannot carry full capability accumulation, and the update signal is forced back toward memory and parameters. The three layers are therefore not substitutes for one another but a hierarchy whose division of labor follows their timescales and capacity ceilings. A purely engineering route that tries to replace continual learning with ever longer context will meet a bottleneck on long-horizon tasks.

Retrieval decay and the need for active forgetting. Retrieval-augmented generation has long been read as a way to address continual learning through external memory, but as dialogue and task sequences lengthen it also decays: retrieval becomes dominated by stale documents, low-quality entries accumulate, relevance scores drift, and cross-session consistency degrades. This is the same motivation that led MemoryBank to introduce an Ebbinghaus-style forgetting curve [16]: external memory likewise requires an active forgetting mechanism. Recent surveys of agent memory frame its development as an evolution from storage to reflection to experience and identify long-range consistency and continual learning as its core drivers [182]. The implication is that continual learning at the memory layer cannot focus only on how to write and retrieve, but must also study how to selectively reduce content, since otherwise the memory system degrades from a capability carrier into an accumulator of noise. The trade-off between stability and plasticity, classically posed at the parameter layer, thus shifts to the memory layer and calls for its own metrics.

Taken together, this section converges on a question that still lacks a systematic solution: which capabilities should be internalized into parameters over the long run, which should be retained in memory, and which should be used and discarded within context. Before asking how such scheduling might be organized, however, we first confront a prior objection: whether the harness engineering already used to manage these carriers by hand is in itself enough, which we take up next (Section 4.2).

4.2 Why Harness-Level Accumulation Is Not Enough

Since no single carrier suffices, capability has to be spread across several at once, and something has to manage how they are combined. Today that management is largely done by hand, which prompts a question worth asking: whether the engineering progress now carried on the harness already amounts, in part, to continual learning. The question matters because many widely deployed means of extending capability, such as rule files in coding assistants, the evolution of system prompts in agent frameworks, and the steady growth of tool libraries, do not come from learning algorithms in the academic sense but from iterative engineering practice.

Our position is that the engineering progress on the harness already attains part of the goal of continual learning at the functional level, but does not replace it at the mechanistic level, for three reasons. *First*, the harness offers externally editable capability rather than self-accumulated capability: a rule file or an agent tool list can be revised repeatedly by humans so that system behavior keeps improving, but the agent of that revision is a person, and it does not constitute a closed loop driven by the system's own experience. By continual learning we mean that the subject acquires and updates capability autonomously through interaction with the environment. *Second*, the few methods that do close the loop, such as Promptbreeder [30], Voyager [13], and AgentEvolver [17], show that the harness layer can indeed carry continual learning in the genuine sense, yet their update mechanisms are either evolutionary search or reinforcement learning, and they remain far from mature in evaluation, stability, and reproducibility, as discussed in Section 4.4. *Third*, capability on the harness is strongly coupled to context and transfers less readily than capability in parameters: a rule file written for one project or a skill library accumulated for one framework is bound to a specific context and is hard to transfer without loss when that context changes, whereas a parameter tensor, though costly to modify, carries capability that is comparatively transferable once internalized. A common conflation can now be clarified: the intuition that engineering progress is approaching AGI rests on the visible accumulation

of capability on the harness, but a substantial part of that capability resides in the scaffolding around the model rather than in the model itself.

This connects to an open debate worth presenting evenhandedly. One position holds that continual learning remains essential even in the foundation-model era, because a deployed model is a snapshot of the world at training time and must contend with both task-shift and time-shift forgetting [183]. A second position argues that the lack of continual learning is the primary bottleneck on the path from current systems to AGI, since prompt engineering and long rolling contexts are brittle patches that do not generalize beyond text-centric domains [181]. A third position counters that continual learning is a systems problem rather than a learning problem, contending that more context and more computation will let a system composed of several models, memory, and retrieval behave indistinguishably from one that learns continually, so that one need not make the model resemble a human too closely [184]. These positions map onto the central tension of this survey, between engineering compensation and mechanistic internalization. Our own reading is that engineering progress on the harness functionally substitutes for, but does not yet mechanistically constitute, continual learning, and that the role future continual-learning research should take on is to gradually convert the capability accumulation now carried by engineering practice into a process carried by the system's own mechanisms. This leaves an open question that motivates the rest of this section: if hand-tuned engineering is not in itself a self-driven mechanism, what might such a mechanism look like? We turn to that next Section (Section 4.3).

4.3 Coordinating Memory, Skills, Protocols, and Parameters

Having argued that hand-tuned engineering is functionally useful but not in itself a self-driven mechanism, we now ask what a genuine mechanism might look like. Neither model-only nor harness-only updates fully capture how capability is organized in LLM systems, since capability is now distributed across parameters, external memory, skills, and protocols at once. A natural direction is therefore to ask whether, and how, the model and the harness should be updated in a coordinated manner. Rather than treating the coordinated evolution of the two as a settled solution, we present it as a direction suggested by this multi-carrier organization of capability, and we frame it through three couplings that have not yet been studied systematically.

Bidirectional transfer between parameters and harness. The route taken by SKILL0, which explores skills on the harness before internalizing them into parameters [136], illustrates one direction. The reverse direction may matter just as much: making latent capability that is hard to invoke from parameters explicit as a visible skill or memory item on the harness, so as to improve invocation efficiency and interpretability. A complete bidirectional mechanism would have to answer when to consolidate capability into parameters, for items that are frequently invoked and broadly shared across tasks, and when to return capability to the harness, for items that are context-specific or not yet stable. This has a structural analogy with cache and memory hierarchy management in computer systems, but no corresponding formal framework yet exists.

Scheduling capability across carriers. Seen this way, coordination is in large part a scheduling problem: what should stay in context, what should be written to memory, what should be consolidated into skills, and what should ultimately be internalized into parameters. Any such schedule would also have to account for the differing update cadences of the carriers, since memory is written most frequently, at the level of individual interactions, skills expand at an intermediate rate, at the level of tasks, and protocols and parameters are revised most slowly, at the level of projects or long-term constraints. Coordinating writing and reclamation across carriers that move at different rates is a key open problem for turning this direction from intuition into mechanism, and it too lacks a unified solution.

Forgetting becomes multi-faceted and hard to localize. In classical continual learning, forgetting is one-dimensional, in that parameters are overwritten. Under composite carriers it has at least four faces: catastrophic forgetting at the parameter layer, retrieval decay and entry aging at the memory layer, window overflow and attention dilution at the context layer, and capability mismatch at the skill and protocol layer caused by context drift. These differ in timescale, observability, and reversibility, are hard to describe with a single metric, and make it difficult to localize where forgetting actually occurs. Given this, it may be more useful to recast forgetting, from a catastrophe at the parameter layer into a process of active compression and release across composite carriers, which would in turn call for unified criteria for which layer to act on,

how fast, and what to retain or release. Developing such criteria is, on this view, the central obstacle to moving coordinated evolution from intuition to theory.

4.4 Long-Horizon Agents as the Real Testbed

If coordinated updating along these lines is to be more than an intuition, it has to be testable. Future continual learning should be judged not by short tasks or static benchmarks, but by whether an agent can hold its goal, track state, correct errors, and accumulate experience across long, multi-step interactions in open environments. This is precisely the setting in which the carrier ceilings of Section 4.1 and the coordinated updating discussed in Section 4.3 interact and where problems surface most readily.

Why training on a fixed set is insufficient. A static benchmark measures the capability a model ships with, whereas a long-horizon task measures the net gain of capability during operation. A model that scores highly on a fixed test set need not keep its goal from drifting or its state from being lost over tens or hundreds of interaction steps. This pushes the evaluation criterion from single-point accuracy toward trajectory-level measures, and it explains why inference-time update mechanisms such as test-time training become necessary: capability must be replenished during inference and deployment rather than frozen at training time. Recent long-horizon memory benchmarks and surveys respond to exactly this gap, casting agent memory as an evolution from storage to reflection to experience and treating long-range consistency as a primary driver [182].

Memory compression and architecture are drawn in together. Over long runs, memory cannot grow without bound and must be continually compressed, consolidated, and forgotten, echoing the selective reduction discussed in Section 4.1. When both context and external memory reach their ceilings, the burden of carrying long-horizon capability flows back to the model architecture itself. Long-horizon operation is therefore not merely a matter of longer tasks; it forces memory mechanisms, update timing, and architectural design to change in concert, which is also why long-horizon capability is rarely obtained through a single-point improvement.

Error accumulation is the core difficulty. When a task spans tens to hundreds of steps, small per-step errors accumulate and amplify along the call chain: an incorrect retrieval at step k can lead the agent to invoke the wrong skill at step $k{+}1$ and shift later decisions as a whole. Such compounding error is studied in the sequential-learning setting of classical continual learning, but it takes on new features in LLM agent systems. First, errors arise not only at the parameter layer but across a composite state made up of memory, skills, protocols, and parameters. Second, most harness operations are non-differentiable, so conventional gradient-based sensitivity analysis no longer applies. Third, an agent lacks a clear boundary between training and evaluation, so an erroneous state may be written back into memory through immediate feedback and form a positive feedback loop. Reflexion corrects through verbal self-reflection [14] and TTRL corrects through deployment-time updates [158], showing that drift-resistant routes are feasible, yet both remain single-mechanism and lack a unified framework. Establishing a verifiable model of error accumulation and drift-resistant mechanisms for long-horizon agents is thus one of the most challenging and valuable directions for the coming years.

4.5 Continual Learning as a Priority on the Path to AGI

The most open-ended question is how far the memory-and-harness route still is from genuine AGI. Because any forecast of AGI risks excess optimism or pessimism, this subsection offers no definite answer. Instead it draws together several signals visible in the developments reviewed above and then states a clear judgment about research priorities.

Signal one: the unit of study extends from the model to the system. Classical continual learning studies a model, whereas the real unit of work in the LLM era is increasingly a system composed of the model, memory, skills, protocols, tools, and environment. A paradigm worth pursuing is an end-to-end learnable agent framework, in which the components of the harness are no longer hand-built scaffolding but learnable modules trained and evolved jointly with the model. AgentEvolver offers an early demonstration [17], but a complete theory of system-level continual learning does not yet exist.

Signal two: verifiable rewards as a bridge. Reinforcement learning with verifiable rewards [23], training pipelines in the style of DeepSeek-R1 [4], and the use of verifiable signals at deployment time by TTRL [158] point

together to one shift: when a task outcome can be verified automatically, through code execution, the correctness of a mathematical solution, or passing unit tests, an agent can keep improving itself without human labels. Verifiable rewards are thus a key bridge from engineering progress to a continual-learning mechanism. The limitation is equally clear, since many real tasks, such as writing, consulting, and long-horizon planning, carry no explicit verifiable signal, so future work needs frameworks for semi-verifiable rewards or composite feedback.

Signal three: forgetting reinterpreted as capability management rather than a defect. As recast in Section 4.3, selective forgetting is better seen as an active means of managing capability than as a passive decline; memory systems such as MemoryBank [16] and atomic memory [80], the pruning of tool libraries, and the finiteness of context all point this way, and the foundation-model perspective likewise treats selective forgetting as an important direction, distinguishing task-shift from time-shift forgetting [183].

A judgment about AGI priorities. We are inclined to locate the role of continual learning on the path to AGI as follows: continual learning for the frontier model is the first priority, whereas domain-specific continual learning is not. The value of future continual learning lies less in having a narrow model repeatedly absorb the knowledge of one domain, and more in letting the frontier model keep growing across a broad range of knowledge and skills. Achieving this requires coordinated improvement across these dimensions, spanning both the parameters and the harness, rather than relying on domain-specific knowledge alone or on single-point changes that touch only the model or only the harness, even though, at the present stage, changing only the model or only the harness can still yield sizable gains. This judgment is consistent with both poles of the debate above: it accepts the diagnosis that continual learning is a primary bottleneck toward AGI, and it also accepts the optimism that a systematized route can come close, while placing the emphasis on the systematic organization of capability at the level of the frontier model. It has further been argued that, once online learning is truly solved, a model could pool what it learns across all of its copies, so that a single system effectively learns every job, which helps explain why continual learning for the frontier model carries such an overriding priority [181].

An honest estimate of the distance. Taking the signals together, we are inclined to believe that the memory-and-harness route is rapidly approaching the capability ceiling of a general assistant at the engineering level, but that genuine AGI, a system capable of autonomous capability expansion, cross-domain transfer, and robust long-horizon decision making, remains a considerable distance away, and that this distance will not be closed simply by longer context, larger memory, or more skills. Other conditions remain uncertain, yet continual learning is an unavoidable part of the path; recasting it from an isolated algorithmic problem into a systematic question of how capability is organized is the direction that the perspective advocated in this survey is meant to keep pointing toward.

5 Conclusion

This survey reframes continual learning in the LLM era as a boundary extension along three independent dimensions: *When*, *Where*, and *How*. Classical continual learning is situated as a single specific point in the three-axis coordinate space, while LLM-era methods extend the boundary independently along each axis (Sections 3.2, 3.3, and 3.4) and increasingly depart from the classical coordinate on multiple axes at once, moving into the interior of the coordinate space (Section 3.5). The framework not only organizes the distribution of existing methods but, through the contrast between dense and empty regions, identifies unfilled coordinates that offer concrete directions for future work. We expect the taxonomy to serve as a unified reference for describing and advancing methods as LLM agent systems continue to mature toward engineering practice.

We acknowledge several limitations of this work. First, the three-axis taxonomy admits crossings at its boundaries; the classification of some cross-axis methods follows the primary-axis convention adopted in Section 3.1, and alternative classifications are equally reasonable. Second, LLM continual learning is a rapidly iterating area; our coverage is limited to work publicly available before the submission deadline and may lag behind the most recent arXiv preprints. Third, evaluation for harness-layer and cross-axis methods is itself in rapid flux, so the evaluation gaps discussed in Section 4.4 are necessarily a coarse summary, leaving the design of concrete protocols for future work.

References

[1] OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

[2] GLM Team. ChatGLM: A family of large language models from GLM-130B to GLM-4 all tools, 2024. URL https://arxiv.org/abs/2406.12793.

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

[4] Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

[5] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi K2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

[6] Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models, 2025. URL https://arxiv.org/abs/2508.06471.

[7] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chengxing Xie, Cunxiang Wang, et al. GLM-5: From vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

[8] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process-reward-guided tree search. In *NeurIPS*, 2024.

[9] Dan Zhang, Min Cai, Jonathan Light, Ziniu Hu, Yisong Yue, and Jie Tang. TDRM: Smooth reward models with temporal difference for LLM RL and inference, 2025. URL https://arxiv.org/abs/2509.15110.

[10] Sining Zhoubian, Dan Zhang, and Jie Tang. ReST-RL: Achieving accurate code reasoning of LLMs with optimized self-training and decoding, 2025. URL https://arxiv.org/abs/2508.19576.

[11] Xiao Xia, Dan Zhang, Zibo Liao, Zhenyu Hou, Tianrui Sun, Jing Li, Ling Fu, and Yuxiao Dong. Scenegenagent: Precise industrial scene generation with coding agent. *arXiv preprint arXiv:2410.21909*, 2024.

[12] Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MemoryLLM: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

[13] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

[14] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

[15] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[16] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2305.10250.

[17] Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. AgentEvolver: Towards efficient self-evolving agent system, 2025. URL https://arxiv.org/abs/2511.10395.

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

[19] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

[20] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024. URL https://arxiv.org/abs/2302.00487.

[21] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.

[22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[24] Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

[25] 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. URL https://arxiv.org/abs/2306.13649.

[26] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897. Method label in Figure 2: SDFT.

[27] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in LLMs, MLLMs, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

[28] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.17333.

[29] Jiayi Weng. Learning beyond gradients. Blog post, 2026. URL https://trinkle23897.github.io/learning-beyond-gradients/. Contextual overview rather than a primary technical source.

[30] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023. URL https://arxiv.org/abs/2309.16797.

[31] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.10726.

[32] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 57503–57522. PMLR, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

[33] Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in LLM agents: A unified review of memory, skills, protocols and harness engineering, 2026. URL https://arxiv.org/abs/2604.08224.

[34] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[36] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

[37] Tao Feng, Wei Li, Hangjie Yuan, Liyuan Wang, Yuxiao Dong, and Minlie Huang. Infty engine: An optimization toolkit to support continual ai. *GitHub repository*, 2026. URL https://github.com/THUDM/INFTY.

[38] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

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

[40] Ang Bian, Wei Li, Hangjie Yuan, Chengrong Yu, Mang Wang, Zixiang Zhao, Aojun Lu, Pengliang Ji, and Tao Feng. Make continual learning stronger via c-flat. *Advances in Neural Information Processing Systems*, 37: 7608–7630, 2024.

[41] Wei Li, Hangjie Yuan, Zixiang Zhao, Borui Kang, Ziwei Liu, and Tao Feng. A faster path to continual learning. *CVPR*, 2026.

[42] Aojun Lu, Hangjie Yuan, Tao Feng, and Yanan Sun. Rethinking the stability-plasticity trade-off in continual learning from an architectural perspective. *ICML*, 2025.

[43] Aojun Lu, Tao Feng, Hangjie Yuan, Xiaotian Song, and Yanan Sun. Revisiting neural networks for continual learning: An architectural perspective. *IJCAI*, 2024.

[44] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[45] 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.

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

[47] Tao Feng, Mang Wang, and Hangjie Yuan. Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *CVPR*, pages 9417–9426. IEEE, 2022.

[48] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

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

[50] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget, 2018. URL https://arxiv.org/abs/1711.09601.

[51] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[52] 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, 2024. URL https://arxiv.org/abs/2403.08763. Method label in Figure 2: SimpleCPT.

[53] Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re)warm your model?, 2023. URL https://arxiv.org/abs/2308.04014.

[54] Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[55] Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. ELLE: Efficient lifelong pre-training for emerging data. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

[56] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. TimeLMs: Diachronic language models from twitter. In *Proceedings of ACL 2022: System Demonstrations*, 2022.

[57] Haokai Ma, Yunshan Ma, Ruobing Xie, Lei Meng, Jialie Shen, Xingwu Sun, Zhanhui Kang, and Tat-Seng Chua. Large language model empowered recommendation meets all-domain continual pre-training. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2026. doi: 10.1109/TKDE.2026.3717059.

[58] Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

[59] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2025.

[60] Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671, 2023.

[61] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[62] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, et al. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[63] Jiayi Han, Liang Du, Hongwei Du, Xiangguo Zhou, Yiwen Wu, Yuanfang Zhang, Weibo Zheng, and Donghong Han. SLIM: Let LLMs learn more and forget less with soft LoRA and identity mixture. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4792–4804, 2025. Soft-LoRA component shown as SLoRA in Figure 2.

[64] Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. SAPT: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[65] Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. Rehearsal-free modular and compositional continual learning for language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

[66] Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. InsCL: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

[67] Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[68] Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. *arXiv preprint arXiv:2411.06171*, 2024.

[69] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations (ICLR)*, 2025.

[70] Mingda Liu, Zhenghan Zhu, Ze'an Miao, and Katsuki Fujisawa. Norm anchors make model edits last, 2026.

[71] Yujie Feng, Hao Wang, Jian Li, Xu Chu, Zhaolu Kang, Yiran Liu, Yasha Wang, Philip S. Yu, and Xiao-Ming Wu. FOREVER: Forgetting curve-inspired memory replay for language model continual learning, 2026.

[72] Tao Hu and Da-Wei Zhou. Dynamic cross-modal prompt generation for multimodal continual instruction tuning, 2026.

[73] Jun-Tao Tang, Zhen-Hao Xie, Yu-Cheng Shi, and Da-Wei Zhou. CRAM: Centroid-routing and adaptive MoE for multimodal continual instruction tuning, 2026.

[74] Ziqi Wang, Chang Che, Qi Wang, Hui Ma, Zenglin Shi, Cees G. M. Snoek, and Meng Wang. Harmonious parameter adaptation in continual visual instruction tuning for safety-aligned MLLMs, 2025.

[75] Qianyu Chen, Canran Xiao, and Runxuan Tang. Hidden forgetting in continual multimodal learning: When accuracy survives but grounding fails, 2026.

[76] Jingwen Chen, Wenkai Yang, Shengda Fan, Wenbo Nie, Chenxing Sun, Shaodong Zheng, Yangen Hu, Lu Pan, Ke Zeng, and Yankai Lin. Rethinking continual experience internalization for self-evolving LLM agents, 2026.

[77] Ali Behrouz, Farnoosh Hashemi, and Vahab Mirrokni. Language models need sleep: Learning to self-modify and consolidate memories, 2026.

[78] Yuchen Guo, Junli Gong, Weicheng Wang, Hongmin Cai, Yiu ming Cheung, and Weifeng Su. Peam: Parametric embodied agent memory through contrastive internalization of experience in minecraft, 2026.

[79] Zhiyuan Fan, Wenwei Jin, Feng Zhang, Bin Li, Yihong Dong, Yao Hu, and Jiawei Li. Evolving-rl: End-to-end optimization of experience-driven self-evolving capability within agents, 2026.

[80] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents, 2025. URL https://arxiv.org/abs/2502.12110.

[81] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[82] 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*, 2024.

[83] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning (ICML)*, 2025. Online setting shown as AWM-online in Figure 2.

[84] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025.

[85] Dahyun Jung, Jaewook Lee, and Heuiseok Lim. Towards scalable lifelong knowledge editing with selective knowledge suppression, 2026.

[86] Beining Wu, Zihao Ding, Jun Huang, and Yanxiao Zhao. Forget to improve: On-device LLM-agent continual learning via budget-curated memory, 2026.

[87] Zhiyuan Hu, Yunhai Hu, Juncheng Liu, Shuyue Stella Li, Yucheng Wang, Zhen Xu, Xinxing Xu, See-Kiong Ng, Anh Tuan Luu, Bryan Hooi, Cynthia Breazeal, and Hae Won Park. Collaborative multi-agent test-time reinforcement learning for reasoning, 2026.

[88] Tom 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 (NeurIPS)*, 2023.

[89] 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 (NeurIPS)*, 2024.

[90] Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory, 2026. URL https://arxiv.org/abs/2601.03192.

[91] Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2025.

[92] Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning, 2025.

[93] Sikuan Yan, Ahmed Bahloul, Ercong Nie, Susanna Schwarzmann, Riccardo Trivisonno, Volker Tresp, and Yunpu Ma. Memory-R2: Fair credit assignment for long-horizon memory-augmented LLM agents, 2026.

[94] Zhiyu Shen, Ziming Wu, Fuming Lai, Shaobing Lian, and Yanghui Rao. MemBuilder: Reinforcing LLMs for long-term memory construction via attributed dense rewards, 2026.

[95] Junwei Liao, Haoting Shi, Ruiwen Zhou, Jiaqian Wang, Shengtao Zhang, Wei Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Bo Tang, Weinan Zhang, and Muning Wen. Memq: Integrating q-learning into self-evolving memory agents over provenance dags, 2026.

[96] Mingyu Yang, Keye Zheng, Congchao Cheng, Yujie Liu, Xingkang Lu, Fan Jiang, and Yefei Zheng. Marginal advantage accumulation for memory-driven agent self-evolution, 2026.

[97] Yibo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yufei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in LLM agents without gradient updates, 2026.

[98] Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2025.

[99] Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, Yu Qiao, and Haifeng Li. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks, 2025.

[100] Yuxin Jin, Siyuan Zhang, Hanchen Wang, Lu Qin, Ying Zhang, and Wenjie Zhang. Exg: Self-evolving agents with experience graphs, 2026.

[101] Zihan Chen, Songwei Dong, Chengshuai Shi, Peng Wang, Song Wang, Cong Shen, and Jundong Li. The past is prologue: A plug-in controller for selective updates in sequentially evolving LLM memory, 2026.

[102] Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simple-Mem: Efficient lifelong memory for LLM agents, 2026.

[103] Shengguang Wu, Hao Zhu, Yuhui Zhang, Xiaohan Wang, and Serena Yeung-Levy. Automem: Automated learning of memory as a cognitive skill, 2026.

[104] Xiaoxing Wang, Ning Liao, Shikun Wei, Chen Tang, and Feiyu Xiong. Autoagent: Evolving cognition and elastic memory orchestration for adaptive agents, 2026.

[105] Jiajie Jin, Yuyang Hu, Kai Qiu, Qi Dai, Chong Luo, Guanting Dong, Xiaoxi Li, Tong Zhao, Xiaolong Ma, Gongrui Zhang, Zhirong Wu, Bei Liu, Zhengyuan Yang, Linjie Li, Lijuan Wang, Hongjin Qian, Yutao Zhu, and Zhicheng Dou. Toward generalist autonomous research via hypothesis-tree refinement, 2026.

[106] Yufei He, Juncheng Liu, Zhiyuan Hu, Yulin Chen, Yue Liu, Yuan Sui, Yibo Li, Nuo Chen, Jun Hu, Bryan Hooi, Xinxing Xu, and Jiang Bian. Evoclinician: A self-evolving agent for multi-turn medical diagnosis via test-time evolutionary learning, 2026.

[107] Tao Ren, Weiyao Luo, Hui Yang, Rongzhi Zhu, Xiang Huang, Yuchuan Wu, Bingxue Chou, Jieping Ye, Jiafeng Liang, Yongbin Li, and Yijie Peng. Scaling self-evolving agents via parametric memory, 2026. URL https://arxiv.org/abs/2606.04536. Method label in Figure 2: TMEM.

[108] Zhiruo Wang, Daniel Fried, and Graham Neubig. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *International Conference on Machine Learning (ICML)*, 2024.

[109] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

[110] Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, and Panpan Xu. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025.

[111] Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. AutoSkill: Experience-driven lifelong learning via skill self-evolution, 2026.

[112] Zelin He, Haotian Lin, Boran Han, Wei Zhu, Haoyang Fang, Bernie Wang, Xuan Zhu, Runze Li, and Matthew Reimherr. Reskill: Reconciling skill creation with policy optimization in agentic rl, 2026.

[113] Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning, 2026.

[114] Xiaoyuan Li, Moxin Li, Keqin Bao, Yubo Ma, Wenjie Wang, Dayiheng Liu, and Fuli Feng. Skillgraph: Skill-augmented reinforcement learning for agents via evolving skill graphs, 2026.

[115] Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning, 2026.

[116] Qi Zhang, Zhaopeng Feng, Xiaonan Shi, Xiaomeng Hu, Chu Liu, Pengjun Xie, Xiaobin Wang, Jieping Ye, Bryan Hooi, Haobo Wang, and Junbo Zhao. Skillcomposer: Learning to evolve agent skills for specification and generalization, 2026.

[117] Haipeng Ding, Yuexiang Xie, Zhewei Wei, Yaliang Li, and Bolin Ding. From atomic actions to standard operating procedures: Iterative tool optimization for self-evolving LLM agents, 2026.

[118] Zhang Zhang, Shuqi Lu, Hongjin Qian, Di He, and Zheng Liu. Agentfactory: A self-evolving framework through executable subagent accumulation and reuse, 2026.

[119] Kalle Kujanpää, Ning Liu, Shahnawaz Alam, Yeshwanth Reddy Sura, Tianyu Yang, Kristina Klinkner, and Shervin Malmasi. Tool-making and self-evolving LLM agents in low-latency systems, 2026.

[120] Youyuan Zhang, Jialiang Sun, Hangrui Bi, Chuqin Geng, Wenjie Ma, Zhaoyu Li, and Xujie Si. Dreamprover: Evolving transferable lemma libraries via a wake-sleep theorem-proving agent, 2026.

[121] Xing Zhang, Yanwei Cui, Guanghui Wang, Ziyuan Li, Wei Qiu, Bing Zhu, and Peiyang He. Ratchet: A minimal hygiene recipe for self-evolving LLM agents, 2026.

[122] Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. SkillWeaver: Web agents can self-improve by discovering and honing skills, 2025. URL https://arxiv.org/abs/2504.07079.

[123] Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks, 2025.

[124] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models, 2024.

[125] Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, Xing Zhou, Dongrui Liu, Ling Yang, Yue Wu, Kaixuan Huang, Shilong Liu, Hongru Wang, and Mengdi Wang. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution, 2025.

[126] Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory, 2025.

[127] Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-Pro: Learning reusable skills from experience via non-parametric PPO for LLM agents, 2026.

[128] Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. Muse-autoskill: Self-evolving agents via skill creation, memory, management, and evaluation, 2026.

[129] Libin Qiu, Zhirong Gao, Junfu Chen, Yuhang Ye, Weizhi Huang, Xiaobo Xue, Wenkai Qiu, and Shuo Tang. AutoRefine: From trajectories to reusable expertise for continual LLM agent refinement, 2026.

[130] Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. Mem^{2}evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation, 2026.

[131] Xingyan Liu, Xiyue Luo, Linyu Li, Ganghong Huang, Jianfeng Liu, and Honglin Qiao. Skillforge: Forging domain-specific, self-evolving agent skills in cloud technical support, 2026.

[132] Jingbo Yang, Guanyu Yao, Yang Zhang, Ramana Rao Kompella, Gaowen Liu, and Shiyu Chang. Federatedskill: Federated learning for agentic skill evolution, 2026.

[133] Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026.

[134] Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning, 2025. URL https://arxiv.org/abs/2511.16043.

[135] Bo Mao, Jie Zhou, Yutao Yang, Xin Li, Xian Wei, Qin Chen, Xingjiao Wu, and Liang He. Learning while acting: A skill-enhanced test-time co-evolution framework for online lifelong learning agents, 2026.

[136] Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. SKILL0: In-context agentic reinforcement learning for skill internalization, 2026. URL https://arxiv.org/abs/2604.02268.

[137] Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents, 2026.

[138] Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning, 2025. URL https://arxiv.org/abs/2507.19457.

[139] Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent systems, 2024.

[140] Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning, 2024.

[141] Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization, 2024.

[142] Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents, 2024.

[143] Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents, 2024.

[144] Chen Ling, Pei Chen, Albert Guan, Jiaming Qu, Shayan Ali Akbar, Madhu Gopinathan, and Erwin Cornejo. Pace: Two-timescale self-evolution for small language model agents, 2026.

[145] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems, 2024.

[146] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2024.

[147] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space, 2024.

[148] Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. Evoflow: Evolving diverse agentic workflows on the fly, 2025.

[149] Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. AgentNet: Decentralized evolutionary coordination for LLM-based multi-agent systems, 2025.

[150] Siwei Liu, Jinyuan Fang, Han Zhou, Yingxu Wang, and Zaiqiao Meng. Sew: Self-evolving agentic workflows for automated code generation, 2025.

[151] Wangcheng Tao, Han Wu, and Weng-Fai Wong. Sepo: Self-evolving prompt agent for system prompt optimization, 2026.

[152] Wentao Zhang et al. Autogenesis: A self-evolving agent protocol, 2026.

[153] Henry Jiang. Darwin: Dynamic agentically rewriting self-improving network, 2026.

[154] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents, 2025.

[155] Zhaorui Yang, Qian Liu, Tianyu Pang, Han Wang, Haozhe Feng, Minfeng Zhu, and Wei Chen. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2402.13669.

[156] Han Zhang, Lin Gui, Yu Lei, Yuanzhao Zhai, Yehong Zhang, Zhuo Zhang, Yulan He, Hui Wang, Yue Yu, Kam-Fai Wong, Bin Liang, and Ruifeng Xu. COPR: Continual human preference learning via optimal policy regularization. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

[157] Han Zhang, Yu Lei, Lin Gui, Min Yang, Yulan He, Hui Wang, and Ruifeng Xu. CPPO: Continual learning for reinforcement learning with human feedback. In *International Conference on Learning Representations (ICLR)*, 2024.

[158] Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

[159] Meng Wang, Haohan Zhao, Wenzhuo Liu, Lu Yang, Geng Liu, Haiyang Guo, Guo-Sen Xie, Gaofeng Meng, Hongbin Liu, and Fei Zhu. Denser $\neq$ better: Limits of on-policy self-distillation for continual post-training, 2026.

[160] Jonas Hübotter, Leander Diaz-Bone, Ido Hakimi, Andreas Krause, and Moritz Hardt. Learning on the job: Test-time curricula for targeted reinforcement learning. *arXiv preprint arXiv:2510.04786*, 2025.

[161] Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[162] Yujie Feng, Jian Li, Xiaoyu Dong, Pengfei Xu, Xiaohui Zhou, Yujia Zhang, Zexin Lu, Yasha Wang, Alan Zhao, Xu Chu, and Xiao-Ming Wu. AIMMerging: Adaptive iterative model merging using training trajectories for language model continual learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13420–13437, Suzhou, China, 2025.

[163] Anton Alexandrov, Veselin Raychev, Mark Niklas Müller, Ce Zhang, Martin Vechev, and Kristina Toutanova. Mitigating catastrophic forgetting in language transfer via model merging. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

[164] Yuanyi Wang, Yifan Yang, Su Lu, Yanggan Gu, Pengkai Wang, Wenjun Wang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, Jialun Cao, Shing-Chi Cheung, and Hongxia Yang. Geometry conflict: Explaining and controlling forgetting in LLM continual post-training, 2026.

[165] Tao Feng, Wei Li, Didi Zhu, Hangjie Yuan, Wendi Zheng, Dan Zhang, and Jie Tang. ZeroFlow: Overcoming catastrophic forgetting is easier than you think. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 16808–16824. PMLR, 2025. URL https://proceedings.mlr.press/v267/feng25j.html.

[166] Fuli Qiao and Mehrdad Mahdavi. Merge before forget: A single LoRA continual learning via continual merging, 2025. Method label in Figure 2: SLAO.

[167] Immanuel Abdi, Akshat Gupta, Micah Mok, Alexander Lu, Nicholas Lee, and Gopala Anumanchipalli. Evolutionary strategies lead to catastrophic forgetting in LLMs, 2026.

[168] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[169] Çağatay Yıldız, Nishaanth Kanna Ravichandran, Prishruit Punia, Matthias Bethge, and Beyza Ermis. Investigating continual pretraining in large language models: Insights and implications, 2024. URL https://arxiv.org/abs/2402.17400.

[170] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yuechen Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *ArXiv*, abs/2308.08747, 2023. URL https://api.semanticscholar.org/CorpusID:261031244.

[171] Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models, 2023. URL https://arxiv.org/abs/2310.06762.

[172] Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models, 2025. URL https://arxiv.org/abs/2501.13453.

[173] Ziwei Liu, Borui Kang, Hangjie Yuan, Zixiang Zhao, Wei Li, Yifan Zhu, and Tao Feng. Continual gui agents. *ICML*, 2026.

[174] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[175] Ruhan Wang, Yucheng Shi, Zongxia Li, Zhongzhi Li, Yue Yu, Junyao Yang, Kishan Panaganti, Haitao Mi, Dongruo Zhou, et al. Harness handbook: Making evolving agent harnesses readable, navigable, and editable. *arXiv preprint arXiv:2607.13285*, 2026.

[176] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[177] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

[178] Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *arXiv preprint arXiv:2403.13187*, 2024.

[179] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2311.03099.

[180] Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.15479.

[181] Dwarkesh Patel. Why I do not think AGI is right around the corner. https://www.dwarkesh.com/p/timelines-june-2025, 2025. URL https://www.dwarkesh.com/p/timelines-june-2025. Essay, Dwarkesh Podcast; contextual commentary rather than a primary technical source.

[182] Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of LLM agent memory mechanisms, 2026. Findings of ACL 2026.

[183] Jack Bell, Luigi Quarantiello, Eric Nuertey Coleman, Lanpei Li, Malio Li, Mauro Madeddu, Elia Piccoli, and Vincenzo Lomonaco. The future of continual learning in the era of foundation models: Three key directions, 2025. arXiv:2506.03320; accepted at the TCAI workshop, 2025.

[184] Nathan Lambert. Contra dwarkesh on continual learning. https://www.interconnects.ai/p/contra-dwarkesh-on-continual-learning, 2025. URL https://www.interconnects.ai/p/contra-dwarkesh-on-continual-learning. Essay, Interconnects; contextual commentary rather than a primary technical source.

[185] Cheng Chen, Junchen Zhu, Xu Luo, Heng Tao Shen, Jingkuan Song, and Lianli Gao. CoIN: A benchmark of continual instruction tuning for multimodel large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 57817–57840, 2024. doi: 10.52202/079017-1844. URL https://doi.org/10.52202/079017-1844.

[186] Haiyun Guo, Zhiyan Hou, Yandu Sun, Jinghan He, Yu Chen, Yuzhe Zhou, Yuheng Jia, Jinqiao Wang, and Tat-Seng Chua. MLLM-CTBench: A benchmark for continual instruction tuning with reasoning process diagnosis, 2025. URL https://arxiv.org/abs/2508.08275.

[187] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2203.13591.

A Glossary of Abbreviations

For readability, Table A.1 summarizes the abbreviations used throughout this survey.

Table A.1 Abbreviations Used in This Survey.

Abbrev.	Expansion
CL	Continual Learning
LLM	Large Language Model
MLLM	Multimodal Large Language Model
CPT	Continual Pre-Training
ICL	In-Context Learning
PEFT	Parameter-Efficient Fine-Tuning
LoRA	Low-Rank Adaptation
MoE	Mixture of Experts
SFT	Supervised Fine-Tuning
SDFT	Self-Distillation Fine-Tuning
RLHF	Reinforcement Learning from Human Feedback
RLVR	Reinforcement Learning from Verifiable Rewards
OPD	On-Policy Distillation
OPSD	On-Policy Self-Distillation
TTA	Test-Time Adaptation
TTT	Test-Time Training
TTRL	Test-Time Reinforcement Learning
EWC	Elastic Weight Consolidation
SI	Synaptic Intelligence
MAS	Memory Aware Synapses
LwF	Learning without Forgetting
GEM	Gradient Episodic Memory
OGD	Orthogonal Gradient Descent
RAG	Retrieval-Augmented Generation
KL	Kullback–Leibler Divergence

B Benchmark Catalogue

Table B.1 summarizes representative benchmarks and evaluation protocols used by the literature reviewed in this survey. The *When*, *Where*, and *How* columns indicate the update regimes directly instantiated or evaluated by each benchmark; they are profile labels rather than mutually exclusive benchmark categories. The catalogue is representative rather than exhaustive.

C Representative Cross-Dimensional Method Profiles

Table C.1 complements Fig. 3 by recording representative methods across the three perspectives. The entries are non-exclusive profile labels, not strict single-valued coordinates: a method may carry several labels within a perspective, and an arrow denotes a carrier trajectory over time. The table is not an exhaustive literature census, and its row counts should not be interpreted as estimates of research density.

Table C.1 Representative cross-dimensional method profiles. Semicolons indicate multiple applicable labels; arrows indicate a temporal carrier trajectory.

Method	When	Where	How	Profile note
CKL [54]	Pre-training	Parameters	Off-policy	Sequential knowledge updating over evolving corpora

Continued on next page

Table C.1, continued

Method	When	Where	How	Profile note
ELLE [55]	Pre-training	Parameters	Off-policy	Expansion and function-preserving initialization during lifelong pre-training
TimeLMs [56]	Pre-training	Parameters	Off-policy	Diachronic language-model updates over temporal data slices
O-LoRA [60]	Post-training	Parameters	Off-policy	Orthogonal adapter subspaces mitigate interference across tasks
LoRAMoE [62]	Post-training	Parameters	Off-policy	Modular LoRA experts preserve and route task-specific capability
SDFT [155]	Post-training	Parameters	Off-policy	Self-distillation uses a fixed teacher distribution during fine-tuning
COPR [156]	Post-training	Parameters	Off-policy	Preference learning regularizes successive policy updates
CPPO [157]	Post-training	Parameters	On-policy	Continual reinforcement learning from human feedback
Self-distillation CL [26]	Post-training	Parameters	On-policy	The current policy supplies responses used for continual self-distillation
AIMMerging [162]	Post-training	Parameters	Learning beyond gradients	Capability is combined through model merging
ZeroFlow [165]	Post-training	Parameters	Learning beyond gradients	Zeroth-order updates avoid ordinary back-propagated gradients
TTT-LM [32]	Inference-time	Parameters	Off-policy	Self-supervised updates are performed on the observed test sequence
TTRL [158]	Inference-time	Parameters	On-policy	Test-time reinforcement learning uses current-policy generations
A-MEM [80]	Inference-time	Harness (Memory)	Learning beyond gradients	Agent memory is reorganized and written directly during use
Reflexion [14]	Inference-time	Harness (Memory)	Learning beyond gradients	Reflective text is generated and written to verbal memory
Voyager [13]	Inference-time	Harness (Skills)	Learning beyond gradients	Executable skills are accumulated in an external library
SkillRL [109]	Post-training	Parameters; Harness (Skills)	On-policy	Reinforcement learning updates the policy while a skill library co-evolves
SkillWeaver [122]	Inference-time	Harness (Skills)	Learning beyond gradients	Web-agent skills are discovered, refined, and stored for reuse
Promptbreeder [30]	Post-training	Harness (Protocols)	Learning beyond gradients	Evolutionary search refines prompts and mutation prompts
GEPA [138]	Post-training	Harness (Protocols)	Learning beyond gradients	Reflective prompt evolution updates the surrounding protocol
AgentEvolver [17]	Post-training	Parameters; Harness (Memory)	On-policy	Reinforcement learning updates policy parameters while selected trajectories accumulate in experience memory
SKILL0 [136]	Post-training	Harness (Skills) → Parameters	On-policy	A curriculum first accumulates reusable skills and then internalizes selected capability

Table B.1 Representative benchmarks and evaluation protocols viewed through the three analytical perspectives.

Benchmark / protocol	When	Where	How	Evaluation setting	Main outputs
TRACE [171]	Post-training	Parameters	Off-policy	Sequential instruction tuning across eight heterogeneous text tasks	Task performance, forgetting, general ability, and instruction following
CoIN [185]	Post-training	Parameters	Off-policy	Multimodal continual instruction tuning over ten datasets from eight task categories	Task performance, forgetting, instruction following, and general-knowledge retention
MLLM-CTBench [186]	Post-training	Parameters	Off-policy; On-policy; Learning beyond gradients	Continual instruction tuning across seven tasks and six domains, comparing supervised fine-tuning, reinforcement fine-tuning, and model fusion	Final-answer accuracy, average performance, backward transfer, forgetting, and process-level reasoning quality
CoTTA protocol [187]	Inference-time	Parameters	Off-policy	Continual test-time adaptation on a non-stationary stream using teacher-based pseudo-labels	Streaming error or accuracy, stability, error accumulation, and retention under domain shifts