Self-Improvements in Modern Agentic Systems: A Survey

Zhe Ren[1], **Yimeng Chen**[2]*, **Dandan Guo**[1,2]*, **Guowei Rong**[1], **Tonghui Li**[1], **R.B. Xiong**[3], **Qingfeng Lan**[4], **Wenyi Wang**[2], **Li Nanbo**[2], **Yibo Yang**[2], **Mingchen Zhuge**[2], **Jürgen Schmidhuber** [2,5]

[1]School of Artificial Intelligence, Jilin University
[2]King Abdullah University of Science and Technology (KAUST)
[3]Independent Researcher [4]University of Alberta
[5]The Swiss AI Lab IDSIA/USI/SUPSI

renzhe25@mails.jlu.edu.cn, yimeng.chen@kaust.edu.sa, guodandan@jlu.edu.cn,
{ronggw25, lith}@mails.jlu.edu.cn, rbxiong1@outlook.com, qlan3@ualberta.ca,
{wenyi.wang, nanbo.li, yibo.yang, mingchen.zhuge, juergen.schmidhuber}@kaust.edu.sa

Self-Improving-Agents Project Page

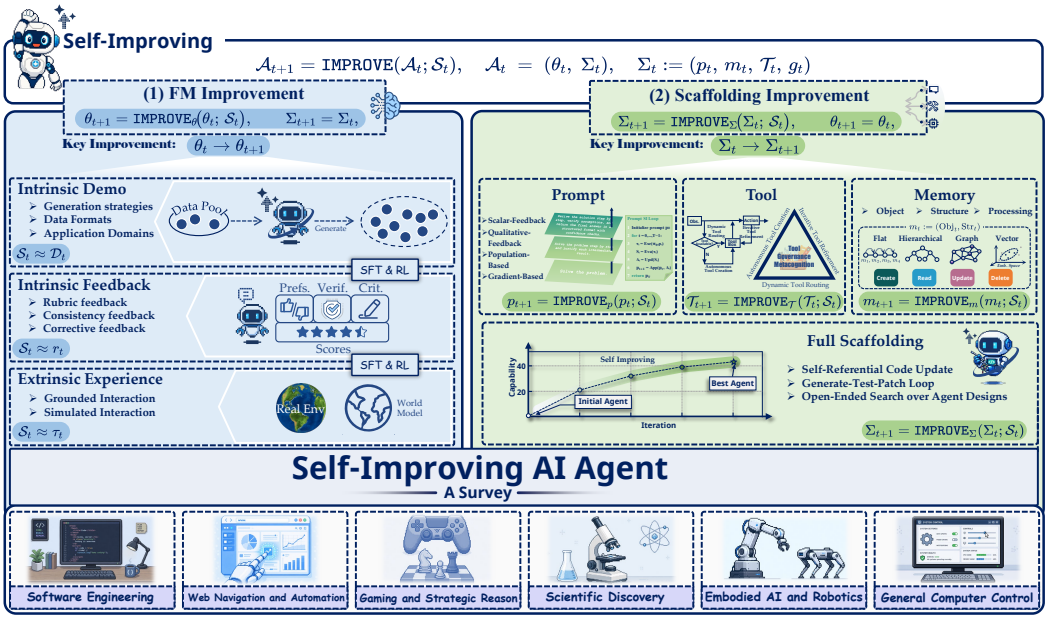


Figure 1: **Overview of self-improvement paradigms for modern AI agents.** We categorize existing methods into two primary pathways according to what is modified. The first pathway is **Foundation Model Improvement**, where the model parameters are updated from θ_t to θ_{t+1} using intrinsic generative demonstrations $\mathcal{D}_t$, intrinsic evaluative feedback e_t, or extrinsic exploratory experience τ_t. The second pathway is **Scaffolding Improvement**, where the operational scaffold is updated from Σ_t to Σ_{t+1} through non-parametric changes. Across scaffold components, a generic update signal $\mathcal{S}_t$ is instantiated to drive improvements in prompts p_t, memory m_t, tools $\mathcal{T}_t$, or the full scaffolding Σ_t.

*Corresponding authors

Abstract

Self-improving autonomous agents are moving from research prototypes to deployed systems. The primary goal is controllable evolution, or adaptation, from experience with minimal or even no human input. This survey frames modern self-improving agents as adaptive systems that convert experience into accumulated capability gains. We offer a system-level framework that represents a modern agent as a configuration coupling a foundation model with an operational scaffold of prompts, memory, tools, and control logic. Within this framework, self-improvement is formalized as a self-induced update operator that obtains and commits updates to model parameters or scaffold components. We organize prior work by update target and by the signals that drive change, then review applications and discuss evaluation, before closing with open problems and future directions. For convenience, we track technical updates on this GitHub page.

Contents

1 Introduction

> *"The first ultraintelligent machine is the last invention that man need ever make."*
> — I. J. Good (1966)

The development of artificial intelligence (AI) technology has driven a paradigm shift in agentic systems (Shoham, 1993; Maes, 1994; Wooldridge & Jennings, 1995; Yao et al., 2022; Wang et al., 2024a), from earlier narrow systems built around task-specific models or hand-engineered modules to modern agentic systems powered by **foundation models** (FMs), including large language models (LLMs) and vision-language models (VLMs), where natural language serves as a shared interface for representation, reasoning, and control. Progress in foundation models has produced a qualitative shift in generalization, yielding striking successes across a wide range of domains, most notably code generation (Chen et al., 2021), language understanding (Hendrycks et al., 2020), and mathematical and formal reasoning (Wei et al., 2022). These advances have brought a long-standing question to the foreground: the prospect of AI systems that improve themselves. Fundamentally, self-improvement is an inherently self-referential process. It defines a system's capacity to autonomously inspect, evaluate, and deliberately modify its own underlying optimization mechanisms and operational logic. Good articulated possible consequences of machine self-improvement (Good, 1966), describing the possibility of an "Intelligence Explosion" once machines acquire the capacity to design more capable successors. Early work on concrete self-improvement algorithms dates back to Schmidhuber's self-referential learning framework (Schmidhuber, 1987), which introduced mechanisms in which a system generates and evaluates modified descendant versions of itself. Establishing the theoretical ceiling of this pursuit, the Gödel Machine (Schmidhuber, 2003b) introduced a fully self-referential algorithm designed to rewrite its own code whenever it can mathematically prove an expected-utility improvement. While a persistent lineage of research successfully demonstrated neural networks (NNs) learning to program other networks via fast weights (Schmidhuber, 1992), advancing to self-referential systems capable of modifying themselves (Schmidhuber, 1993; Irie et al., 2022), and acting as meta-learning systems to discover their own learning algorithms (Hochreiter et al., 2001; Schmidhuber, 2004; Kirsch & Schmidhuber, 2021; 2022; Irie et al., 2025), parallel efforts introduced incremental self-improvement, establishing the ability to enforce long-term reward accelerations through the success-story algorithm for undoing policy self-modifications through backtracking (Schmidhuber, 1994; Schmidhuber et al., 1996; 1997). However, scaling these visionary mechanisms into open-ended agents was historically constrained. Traditionally, systems were forced to search through vast, low-level spaces of assembly-like code or raw synaptic weights. Today, modern FMs alleviate this historical bottleneck by introducing natural language as a unified, highly capable semantic medium for reasoning, policy execution, and self-modification. By drastically reducing the search space for viable modifications, this language-native paradigm has drawn intense recent attention, with growing evidence demonstrating that FM-based self-improving agents can yield substantial empirical gains (Yuan et al., 2024b; Acikgoz et al., 2025; Qi et al., 2025; Zhang et al., 2026d).

To function autonomously in concrete environments, the FM serving as the cognitive core is typically enveloped by an operational scaffold, a structured framework comprising instruction schemes (Yuksekgonul et al., 2024; Guo et al., 2024), memory systems (Chhikara et al., 2025; Zhang et al., 2026e), tool interfaces (Qiu et al., 2025c; Wölflein et al., 2025), and control logic (Hong et al., 2023; Zhuge et al., 2024a; Xiong et al., 2025). Recently, such an operational scaffold is often also referred to as an agent harness (Rajasekaran, 2026; Trivedy, 2026; Zhang et al., 2026b; Zhong & Zhu, 2026), while this survey uses the term scaffold to emphasize the modifiable structures surrounding the foundation model. Operationally, this scaffold acts as a controller that constructs context, selects actions, and enforces constraints. This modern architecture was conceptually foreshadowed by the "learning to think" framework (Schmidhuber, 2015), where a general-purpose controller learned to dynamically query, or "prompt," a predictive world model to generate reasoning sequences akin to a modern "chain of thought" (Wei et al., 2022). Building upon this core-and-scaffold architecture, the integration of these classical control principles has crystallized

into an emergent paradigm, which we refer to as FM-based self-improving agents. Because modern agentic systems comprise both the aforementioned neural core and scaffold, we organize this survey around a unified taxonomy with two primary pathways, as illustrated in Fig. 1 and summarized chronologically in Fig. 2. The first pathway, which we call **foundation model improvement**, aims to achieve a slower but more stable form of long-term consolidation by updating the underlying model, thereby amortizing capability gains across varied tasks (Huang et al., 2023; Zhao et al., 2024b; Wang et al., 2025d; Xiao et al., 2025; Bougie et al., 2026). The second pathway, which we call **scaffold improvement**, is typically faster and more easily reversible; it improves the agent by updating structural components, including prompts, memory, tool interfaces, and end-to-end control logic, to reshape the agent's effective observation and action semantics (Fernando et al., 2024; Chhikara et al., 2025; Wölflein et al., 2025; Zhang et al., 2026d; Borthwick & Ash, 2026). Although these pathways differ in their targets of modification, both are fundamentally driven by learning signals extracted during interaction. Accordingly, our taxonomy further refines these methods by separating *what* is updated from *where* the improvement signals originate, providing a common language for comparing disparate methods on a consistent footing. Fig. 3 provides a unified taxonomy of this survey.

Despite the rapid proliferation of such empirical frameworks, the broader research landscape remains highly fragmented in both terminology and scope. Closely related ideas are often described under different labels (e.g., self-correction, meta-prompting, or self-play), obscuring underlying mechanistic similarities. Recent surveys have provided valuable perspectives by organizing self-evolving agents along dimensions such as what to evolve, when to evolve, and how to evolve (Gao et al., 2026a; Fang et al., 2025a), or by focusing specifically on the autonomous learning capabilities of static LLMs (Tao et al., 2024). However, existing reviews often treat foundation model fine-tuning and agent scaffolding as isolated topics, lacking a unified formal perspective. Furthermore, few trace the conceptual roots of self-improvement back to classical AI. To bridge this gap, our survey provides a rigorous and systematic positioning of these modern agentic systems. We offer a comprehensive taxonomy under a unified formalization, clarifying their historical evolution and providing a clear, forward-looking roadmap for self-improvement. Concretely, our main contributions are as follows:

Dimension	Ours (2026)	Gao et al. (2026a)	Fang et al. (2025a)	Tao et al. (2024)
Agent formulation	✓	✓	✓	○
Definition scope	✓	✓	○	○
Historical roots	✓	–	○	○
Signal lens	✓	○	✓	✓
Update substrate	✓	✓	✓	○
Evaluation lens	✓	✓	✓	✓
Domain coverage	✓	○	○	–
Outlook & issues	✓	○	✓	○

✓ primary ○ secondary – not focus

Table 1: Comparison of organizing emphases across related surveys.

- **Historical Context and Evolution.** We trace the evolutionary roots of self-improving systems from classical AI to modern FM-based agents, establishing a clear trajectory for how foundational learning mechanisms have adapted to the agentic era.
- **A Unified Formalization and Systematic Taxonomy.** Under our proposed formulation, we systematically categorize mechanisms into two distinct pathways: foundation model improvement and scaffold improvement (covering prompt optimization, memory evolution, tool governance, and full-scaffold redesign).
- **Empirical Landscape, Evaluation Paradigms, and Frontiers.** We integrate the classification system with practical applications (such as software engineering, web navigation, and scientific discovery) and critically analyze current evaluation protocols. By comparing mechanism-level benchmarking and end-to-end evaluation, we highlight the security factors and challenges that need to be considered in achieving reliable, continuous self-improvement.

The remainder of this survey is organized as follows. Section 2 traces the historical context and theoretical foundations of self-improving systems. Section 3 introduces the systems view of foundation-model-based agents and formalizes the problem setting. Section 4 presents our

unified taxonomy, outlining the two primary pathways for improvement. These pathways are subsequently surveyed in depth: Section 5 covers updates to the underlying foundation model, while Section 6 focuses on structural modifications to the surrounding scaffold. Section 7 reviews practical instantiations across representative domains. Section 8 discusses protocols and benchmarks for evaluating these systems. Finally, Section 9 synthesizes open problems and safety considerations, followed by concluding remarks in Section 10.

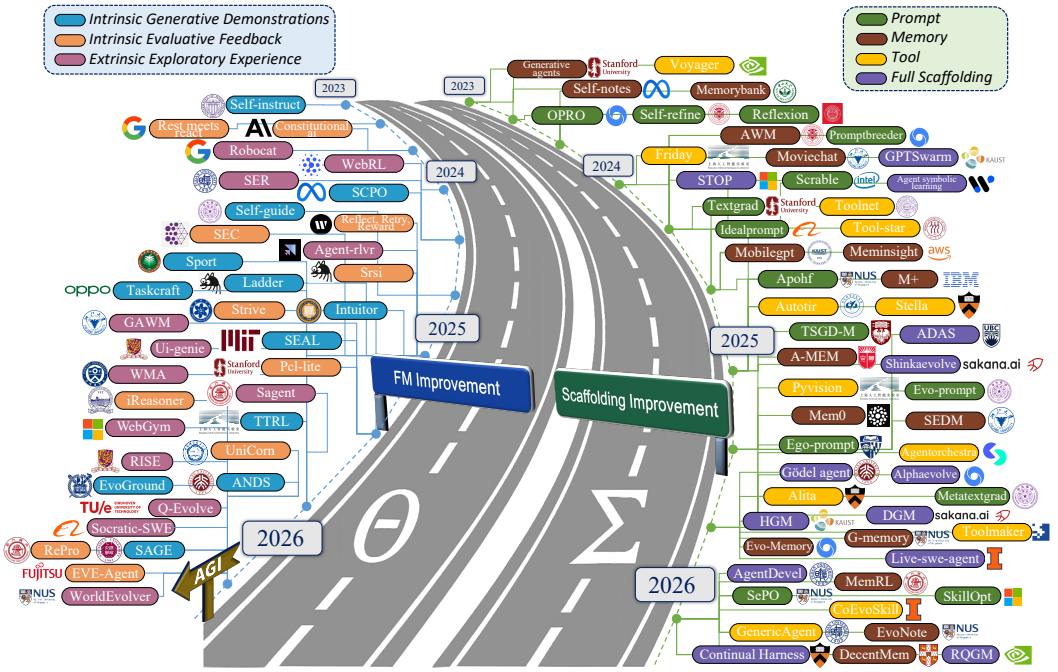


Figure 2: Timeline and taxonomy of self-improvement in foundation-model-based agents (2023–2026). Representative works are positioned by publication year. The left lane denotes foundation-model improvement (θ), while the right lane denotes scaffolding improvement (Σ). The AGI signpost highlights the field's long-term aspiration toward increasingly general agentic intelligence.

2 Historical Context and Theoretical Foundations

Self-improvement is not unique to modern foundation models; it is a foundational objective of artificial intelligence. Whereas standard machine learning optimizes parameters within a fixed architecture, true self-improvement demands that a system explicitly inspect and rewrite its own operational logic, heuristics, or learning algorithms. As illustrated in Figure 4, the mechanisms have been successfully instantiated across several historical paradigms. By tracing this intellectual lineage, we clarify how earlier systems achieved self-improvement within bounded domains, and how modern foundation models now provide a highly expressive, general-purpose substrate to scale these established principles into open-ended and real-world environments.

2.1 Foundational Concepts (1790s-1960s)

The mathematical roots of error-driven adaptation extend back to early optimization frameworks. The method of least squares (Legendre, 1805; Gauss, 1809)[1] demonstrated how

[1]The first famous example of pattern recognition through an NN dates back over 200 years: the rediscovery of the dwarf planet Ceres in 1801 through Gauss, who collected noisy data points from previous astronomical observations, then used them to adjust the parameters of a predictor,

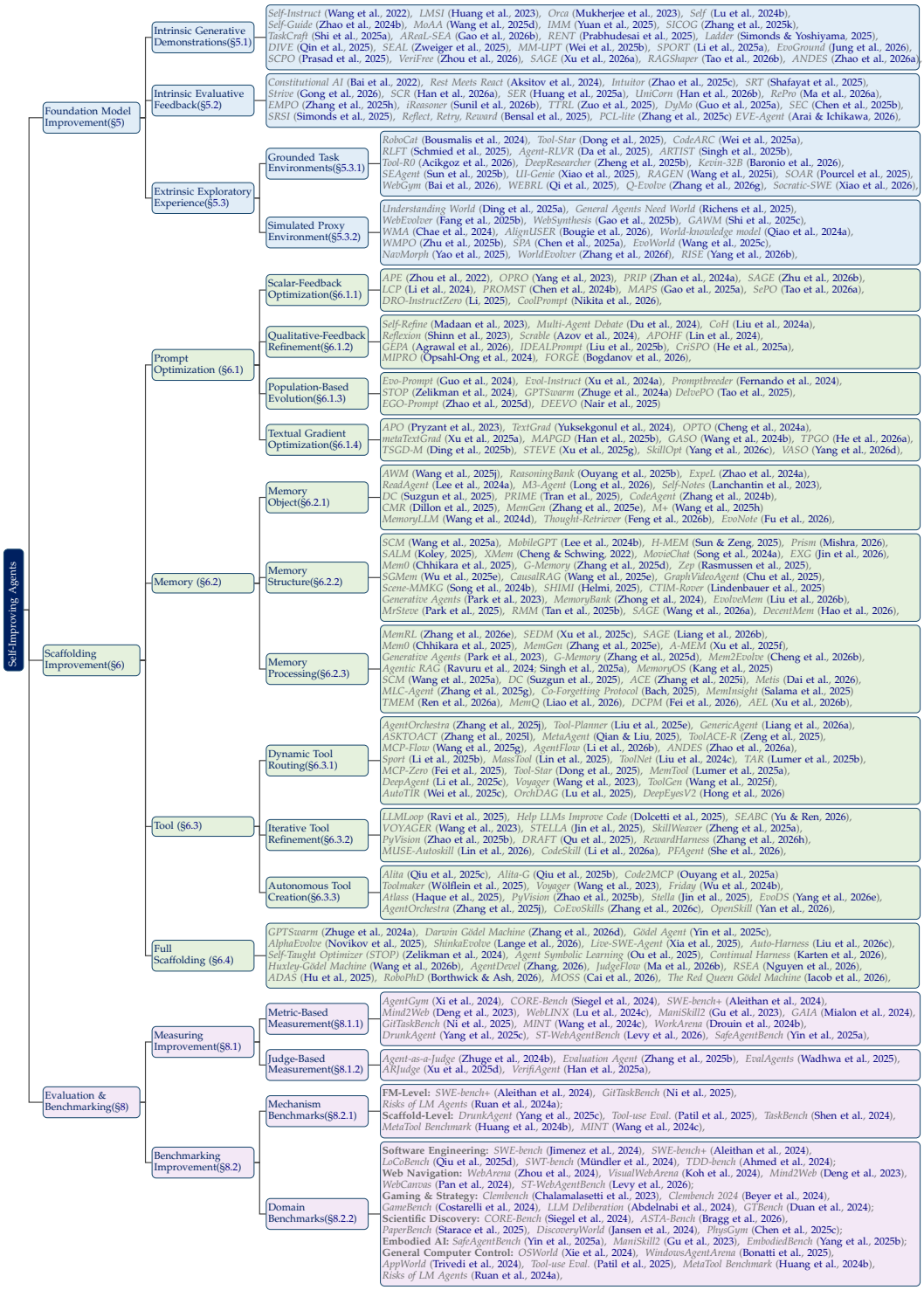


Figure 3: A unified taxonomy of self-improving agents spanning foundation-model updates, scaffold updates, and evaluation benchmarks.

parameters of what's now known as "linear neural networks" can be systematically adjusted to minimize errors. As highlighted in historical retrospectives, this established a

which essentially learned to generalize from the training data to correctly predict the new location of Ceres (Schmidhuber, 2022).

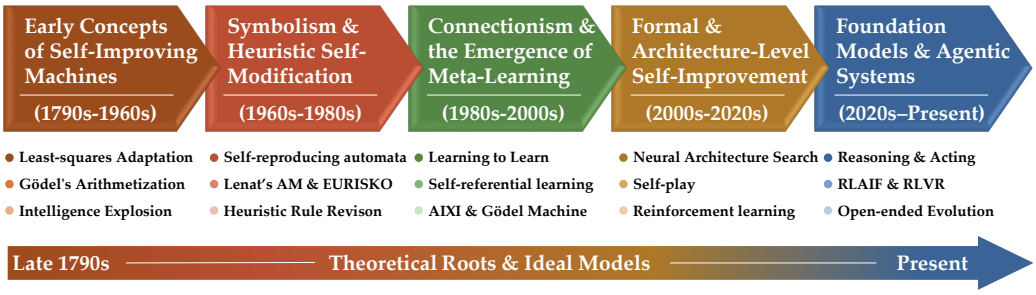


Figure 4: A timeline of theoretical roots and idealized models for self-improving agents, from the late 1790s to the present.

mechanism that is still heavily used today and serves as the very foundation of all neural networks, including deeper ones (Schmidhuber, 2022).

Throughout the mid-20th century, these concepts of adaptation continued to evolve, acting as conceptual precursors. Cybernetics popularized the feedback view of adaptive behavior in closed-loop systems (Wiener, 1948), while Ashby's work on homeostasis emphasized internal state adjustment to maintain viability under perturbations (Ashby, 1952). Alan Turing's "child machine" proposed a model whose internal configuration could be shaped through training and external feedback, such as reward and punishment (Turing, 1948; 1950). Early systems, from Rosenblatt's perceptron (Rosenblatt, 1958) to Samuel's checkers program (Samuel, 1959), provided concrete demonstrations of how performance feedback could change future behavior. However, while these mid-century milestones demonstrated that behavior could optimize over time, their underlying architectures and learning algorithms themselves remained strictly fixed. Consequently, they are best understood as advanced learning systems, rather than fully self-referential, self-improving systems.

The formal basis for transcending fixed learning rules and enabling true self-reference followed a distinct trajectory. In the early 1930s, Kurt Gödel founded modern theoretical computer science and identified fundamental limits of any type of computation-based AI (Gödel, 1931). He introduced a universal coding language based on integers, using it to represent both data and programs in axiomatic form. By famously constructing formal statements that talk about the computation of other formal statements—especially self-referential statements—Gödel provided the theoretical blueprint for programs capable of manipulating their own code (Schmidhuber, 2022). In the 1960s, Good (Good, 1966) introduced the visionary prospect of an "Intelligence Explosion," hypothesizing that machines might one day acquire the capacity to autonomously design more capable successors. However, while this narrative highlighted self-improvement as an ultimate pursuit in AI, it remained a conceptual idea without a formal mathematical or algorithmic framework. Translating this speculative concept into actual deployable mechanisms would require the subsequent development of explicit structural and algorithmic self-modification frameworks.

2.2 Symbolism and Heuristic Self-Modification (1960s–1980s)

As symbolic AI rose to prominence, self-improvement was reinterpreted as the ability to manipulate explicit representations of knowledge and strategy. A key conceptual precursor was von Neumann's theory of self-reproducing automata. Originating in the late 1940s and later published in 1966, it distinguished a constructive mechanism from a symbolic description that can be copied and varied (Von Neumann et al., 1966). Myhill (1964) abstracted self-reproduction into a rigorous formal framework, emphasizing how systems can use symbolic descriptions to generate copies or variants of themselves. Together, these concepts clarified how a system can generate successive versions of increasing complexity by editing the descriptions it interprets. A similar approach in symbolic artificial intelligence is to treat problem-solving heuristics as first-class objects that can be examined, modified, and recombined.

Lenat's Automated Mathematician (AM) system exemplified this approach by using heuristics to propose, extend, and evaluate candidate concepts, effectively searching over symbolic programs that encode mathematical ideas (Lenat[1], 1984). EURISKO pushed further by representing heuristics themselves as manipulable Lisp objects, allowing the system to generate, modify, and test its own problem-solving strategies (Lenat, 1983; Lenat & Brown, 1984). However, as emphasized by Schmidhuber (1987), the practical success of systems like EURISKO depended heavily on the user serving as an external evaluation signal, interpreting outputs and manually pruning unproductive heuristic drift, rather than on the system possessing a truly autonomous closed-loop credit assignment mechanism. This historical limitation underscores a challenge that persists in modern agentic systems, namely the need for robust internal evaluation signals and verification procedures to sustain autonomous recursive self-improvement.

2.3 Connectionism and the Emergence of Meta-Learning (1980s–2000s)

The resurgence of connectionism shifted the emphasis from hand-coded heuristics to learning dynamics as the substrate of improvement. Self-improvement was consequently reframed as explicitly optimizing the learning process itself, encompassing the optimizer, inductive biases, and adaptation rules. Schmidhuber (1987) introduced self-referential learning frameworks in which evolutionary processes could optimize not only candidate solutions but also the learning procedures that generate them, making the learning system itself the object of search. Along a complementary line, Bengio et al. (1991) demonstrated that synaptic learning rules need not be fixed; by parameterizing these rules, the optimizer itself becomes a learnable object.

Schmidhuber introduced fast-weight programmers (FWPs) as a broad class of networks (Schmidhuber, 1991a; 1992; 1993). A representative early instance—retrospectively identified as the 1991 unnormalized linear Transformer (ULTRA) (Schlag et al., 2021)—features a "slow" neural network that learns by gradient descent to program the "fast weights" of another network through additive outer-product updates. The explicitly self-referential version followed in work on self-referential weight matrices, where a network learns to modify its own fast weights while receiving feedback signals such as errors or rewards as inputs, making the learning dynamics themselves part of what is optimized (Schmidhuber, 1993). This line is important for self-improvement because it moves self-modification into a continuous program space, rather than restricting it to symbolic code rewriting. Recent work has revisited this direction in scalable self-referential neural architectures, including modern self-referential weight matrices that learn to modify themselves (Irie et al., 2022), recurrent fast-weight programmers and their self-referential extensions (Irie et al., 2023; 2021), and self-referential networks that meta-learn continual learning algorithms (Irie et al., 2025). Such mechanisms suggest a possible additional meta-level for self-improving foundation models, where the foundation model itself may become part of the modifiable substrate rather than remaining only a frozen component.

This period also consolidated *learning to learn* as a framing for self-improvement. In reinforcement learning (RL), the 1994 paper introduced an RL machine as an early recursive self-improving agent that alter parts of their own learning strategy to shift inductive bias over a single lifelong interaction stream (Schmidhuber, 1994). The same underlying idea was later described using related terminology, including environment-independent reinforcement acceleration and the success-story algorithm (Wiering & Schmidhuber, 1996; Schmidhuber et al., 1996; Schmidhuber & Zhao, 1996; Schmidhuber et al., 1997). Thrun and Pratt likewise emphasized learning reusable inductive biases across task distributions (Thrun & Pratt, 1998). Related neuroevolution work such as NeuroEvolution of Augmenting Topologies (NEAT) improved neural architectures by expanding network topology under performance-driven selection (Stanley & Miikkulainen, 2002). AIXI formalized an uncomputable upper bound on optimal sequential decision-making in computable environments (Hutter, 2000). Furthermore, Schmidhuber (2003b) introduced the Gödel Machine, a fully self-referential, self-improving machine that is theoretically optimal. A Gödel Machine operates in a single-life reinforcement learning environment (Schmidhuber, 1994) and iteratively searches for candidate self-modifications, adopting a modification only when it

can prove that doing so will improve its expected cumulative future reward. To establish this improvement, the proof must reason about future rewards while accounting for all possible subsequent self-modifications. The machine is fully self-referential in the sense that every component of the Gödel Machine is itself subject to modification, including the self-modification generator and the theorem prover.

2.4 Formal and Architecture-Level Self-Improvement (2000s-2020s)

From the mid-2000s onward, research followed two main tracks: (i) analyzing the theoretical limits of self-improvement, and (ii) building practical mechanisms to automate architecture design. Philosophical analyses began to explore the trajectory of these systems, debating the existential risks of the technological singularity (Subramanian, 2010). In theory, Orseau and Ring pointed out that when agents interact with limited resources and irreversible actions, they face significant risks such as self-deception, reward tampering, and fatal errors (Orseau & Ring, 2011). Related research formalized how agents can safely model their environments and themselves without generating logical paradoxes. For instance, Fallenstein et al. (Fallenstein et al., 2015) introduced reflective oracles to reason about probabilistic programs that call themselves, while logical induction offered a mathematical approach to updating beliefs under self-referential uncertainty (Garrabrant et al., 2020). Complementary work on proof-producing reflection in higher-order logic studied how reflective reasoning can be embedded in formal proof systems, providing a technical route for agents or theorem provers to reason about their own reasoning steps (Fallenstein & Kumar, 2015).

Together, these frameworks clarified the gap between ideal models and practical deployments, demonstrating the necessity of safe and constrained self-modification. Early attempts to bridge this gap computationally, such as the endeavor to implement Gödel machines, sought architectures where agents could mathematically prove the optimality of their own code updates (Steunebrink & Schmidhuber, 2012). At the architecture level, Nivel et al. proposed bounded recursive self-improvement, where reflective mechanisms and value-driven scheduling support self-modification under designer-imposed constraints (Nivel et al., 2013). In engineering, automated design has matured into Neural Architecture Search (NAS), where controller learning proposes architectures that can directly optimize task performance (Zoph & Le, 2017). Meanwhile, scalable algorithms such as self-play in reinforcement learning demonstrated how agents can generate increasingly difficult curricula from their own behavior, thus achieving continuous capability growth without human supervision (Silver et al., 2017b). Ultimately, these theoretical and engineering advances have laid the foundation for modern agent systems centered on foundation models.

2.5 Scalable Foundation Models and Agentic Systems (2020s–Present)

The classical vision of autonomous agents—capable of perceiving, acting, and learning in open-ended environments—found a scalable, modern substrate in foundation models (FMs). Powered by the immense knowledge compressed during large-scale pretraining and the flexibility of a unified natural-language interface, FMs function as the cognitive engines of contemporary agentic systems. Crucially, the emergence of these systems fundamentally reshaped the paradigm of self-improvement, decoupling it into two distinct but complementary mechanisms: rapid, training-free adaptation and persistent parameter optimization.

Within this modern paradigm, the most immediate form of self-improvement occurs without computationally expensive weight updates. Because scalable sequence models facilitate broad generalization, agents can achieve rapid, training-free adaptation by dynamically revising their plans, prompts, and memory representations on the fly. This short-horizon self-improvement relies heavily on in-context learning, which effectively functions as a form of meta-learning. Mechanistically, this adaptation is mediated by the attention mechanism's key-value cache, acting as an associative memory that generates transient, context-dependent weight changes during sequence processing. This functional equivalence directly connects modern in-context learning to the 1991 fast-weight programmers (formally identi-

fied as unnormalized linear Transformers), in which networks similarly learned to induce rapid parameter changes during inference without requiring standard gradient descent (Schmidhuber, 1991a; 1992; 1993; Schlag et al., 2021).

Complementing this fast, in-context adaptation are slow improvement loops designed to permanently internalize new capabilities. Reinforcement learning from human feedback (RLHF) and its variants achieved this by converting interaction traces and preference signals into persistent parameter updates (Ouyang et al., 2022). Parallel methodologies advanced partially self-supervised alignment by enabling models to critique their own outputs and provide AI feedback (Bai et al., 2022). Meanwhile, frameworks like Reasoning and Acting (ReAct) and Reflexion build these loops on top of contextual execution, converting execution errors into natural-language reflections (Yao et al., 2022; Shinn et al., 2023). Recent systems pushed this further by maintaining skill libraries, curricula, and evaluation harnesses, spanning open-ended embodied learning (Wang et al., 2023), software-engineering agents with instrumented interfaces (Yang et al., 2024; Zhang et al., 2026d; Wang et al., 2026b), and explicitly self-improving computer-use agents that iterate data, reward models, and policies over generations (Xiao et al., 2025; Sun et al., 2025b). Overall, the modern era reframes self-improvement as nested loops across parameters and scaffolding, renewing classical questions about control, evaluation, and safety at scale.

3 Definitions

3.1 Formulation of Agentic Systems

We begin by clarifying the formal definition of an agent. The core of an autonomous agent lies in perceiving its environment, updating its internal state, and performing actions to achieve a specific goal. While such systems have driven decades of AI research, this survey focuses exclusively on a contemporary class: *foundation-model-based agents*, which utilize large foundation models as their central cognitive engines. Crucially, they leverage natural language as a unified interface to integrate perception, reasoning, and tool manipulation. Because a foundation model fundamentally operates as a stateless inference engine, achieving autonomy requires coupling this cognitive core with a persistent, interactive scaffold. Therefore, we formally define the configuration of a foundation-model-based agent at time step t as:

$$\mathcal{A}_t \;=\; (\theta_t,\, \Sigma_t), \tag{1}$$

where θ_t encapsulates the neural parameters of the foundation model, and Σ_t denotes the agent's dynamic operational scaffold. The scaffold Σ_t specifies how the foundation model is conditioned, grounded, and connected to the external world. Concretely, it can be decomposed as

$$\Sigma_t := (p_t,\, m_t,\, \mathcal{T}_t,\, g_t), \tag{2}$$

where p_t denotes structured prompts or system instructions, m_t denotes memory mechanisms and their retrieval and update policies, $\mathcal{T}_t$ denotes the set of external tools together with their invocation interfaces, and g_t denotes additional control logic such as routing, scheduling, or safety constraints. The interaction between these components and the model's internal parameters is illustrated in Figure 5, which provides a schematic overview of an FM-based agent within our proposed formalism. Together, θ_t and Σ_t determine how the agent reasons, plans, and acts.

During execution, to bridge this intrinsic configuration with the external environment, the agent maintains an ephemeral execution state X_t (e.g., key-value caches, intermediate plans, or short-term working memory) that evolves as it processes an interaction stream. To connect this structural definition to observable behavior, it is beneficial to clarify how the agent's configuration generates its action selection mechanism. Although the foundation model parameters θ_t implement a general generative distribution, the agent's realized behavior is jointly determined by both θ_t and its operational scaffold Σ_t. Accordingly, we denote the induced policy of the agent based on the foundation model as:

$$\pi_{\theta_t,\Sigma_t}(A_t \mid X_t), \tag{3}$$

where A_t denotes the action produced by the system at time step t. Here, the conditioning on X_t captures the transient context of the ongoing interaction. While X_t may strongly influence immediate behavior, it is inherently ephemeral. It is typically discarded or reset once an immediate goal is reached or an external task boundary is crossed, and therefore does not constitute part of the agent's intrinsic architecture. In contrast, the parameters (θ_t, Σ_t) represent the agent's intrinsic configuration over time. While a standard agent may adapt to novel situations by dynamically updating its transient state X_t (e.g., in-context examples), its underlying capabilities remain bounded by its fixed initial setup.

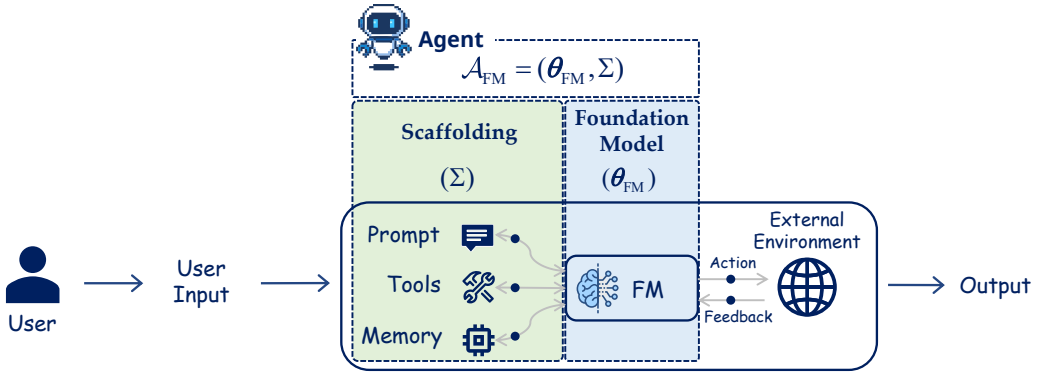


Figure 5: Schematic of an FM-based agent under our formalism.

3.2 Formal Definition of Self-Improvement

Building upon the formal configuration $\mathcal{A}_t = (\theta_t, \Sigma_t)$, we formalize self-improvement in foundation-model-based agents (SI-FMA) as a process of persistent, endogenous adaptation. Rooted in the foundational principles of explicitly self-referential meta-learning (Schmidhuber, 1987; 1993; Schmidhuber et al., 1997), a self-improving agent actively leverages signals induced by its own execution—such as interaction outcomes, critiques, verification results, or proposed edits—to durably modify its underlying computational components.

Self-improvement as a self-induced operator. We conceptualize self-improvement through a *self-induced* operator $\mathcal{U}$ that updates the agent's intrinsic configuration. Let π_{θ_t, Σ_t} denote the agent's induced policy. We formalize this process by factorizing the self-improvement step into an execution phase and an update phase:

$$\mathcal{A}_{t+1} \;=\; \mathcal{U}\Big(\mathcal{A}_{1:t},\; \mathcal{E}\left(\pi_{\theta_t, \Sigma_t}; \Sigma_t, \mathcal{C}_t\right)\Big), \tag{4}$$

where $\mathcal{E}$ denotes an *agent-executed* procedure that produces a learning signal (e.g., interaction trajectories, reflections, critiques, or proposed edits) by running the induced policy against a task context $\mathcal{C}_t$ (e.g., a task distribution, a user interaction stream, or a self-play environment). The scaffold Σ_t is included as an explicit argument to $\mathcal{E}$ to permit direct self-inspection (e.g., critiquing prompt templates or auditing tool configurations).

The system-level update rule $\mathcal{U}$ then applies this self-generated signal to modify the agent's intrinsic components (θ or Σ). Unlike the routine evolution of the transient state X_t (e.g., merely accumulating dialogue history or working memory), the operator $\mathcal{U}$ commits durable changes. Crucially, in line with lifelong meta-learning principles, while these updates are not strictly irreversible and may be undone, they allow the agent to consolidate successful strategies into an increasingly stable policy over its interaction stream (Schmidhuber, 1994; 1995; Wiering & Schmidhuber, 1996; Schmidhuber et al., 1996).

Two modes of self-reference in SI-FMA. SI-FMA may be self-referential in multiple, qualitatively distinct senses. In the first mode, the agent's policy is executed to *indirectly induce* improvement by generating experience or auxiliary artifacts that serve as learning

signals. Self-generated trajectories, evaluations, preferences, or synthetic labels give rise to a learning objective that is subsequently consumed by an update rule, such as an external optimization procedure acting on the foundation model parameters θ_t. This mode realizes self-improvement at the *distributional* level: the agent's behavior shapes the data and supervision that determine its future parameters.

In the second mode, the agent's policy is executed to *directly implement* improvements by modifying components of its own operational definition. Through its actions, the agent may edit prompt templates, manage or reorganize memory, reconfigure tool interfaces, or alter control logic, thereby explicitly updating the scaffolding Σ_t that governs future execution. In this case, self-improvement is realized through direct, action-level self-modification.

These two modes correspond to distinct but complementary paradigms. *Foundation model improvement* realizes self-improvement through self-induced learning in parameter space, whereas *scaffolding improvement* realizes self-improvement through direct self-modification in execution space. Together, they capture the principal mechanisms by which FM-based agents may alter themselves over time.

Foundation model improvement. In foundation model improvement, self-improvement targets the parameters of the underlying foundation model while holding the agent-level scaffolding fixed. Starting from an induced policy π_{θ_t,Σ_t}, the agent is repeatedly executed in the environment, generating self-induced experience through its own interaction trajectories.

We abstract this process by a self-induced update operator that maps the agent's current configuration to an updated set of model parameters:

$$\theta_{t+1} = \mathcal{U}_\theta\big(\theta_{1:t},\ \mathcal{E}(\pi_{\theta_t,\Sigma_t}; \Sigma_t, \mathcal{C}_t)\big), \quad \Sigma_{t+1} = \Sigma_t, \tag{5}$$

where $\mathcal{C}_t$ is the task or deployment context defined above, $\mathcal{E}$ denotes the execution of the agent under its induced policy in $\mathcal{C}_t$, producing learning signals such as rewards, preference comparisons, synthetic labels, critiques, or verification outcomes, and $\mathcal{U}_\theta$ denotes a parameter-learning procedure that updates the foundation model parameters based on these signals.

Importantly, while the construction of these learning signals may involve rule-based evaluators, learned reward models, or auxiliary model invocations, the underlying data distribution is induced by the agent's own policy. The operator $\mathcal{U}_\theta$ may instantiate policy-gradient methods, offline or online reinforcement learning, preference optimization, or other parameter-learning algorithms. Foundation model improvement typically operates on longer time scales, incurs substantial computational cost, and leads to stable, global changes in the agent's internal representations, generalization behavior, and capabilities.

Scaffolding improvement. In *scaffolding improvement*, self-improvement targets the agent-level structures surrounding the foundation model while keeping the model parameters θ fixed. The scaffolding Σ_t governs how histories are converted into the conditioning context for the foundation model, how tool calls are specified and executed, how memory is stored and retrieved, and more generally how token sequences are constrained such that they correspond to valid, executable actions. Through these mechanisms, Σ_t shapes the agent's effective observation and action semantics, as well as the construction and evolution of the agent state X_t during execution.

Formally, scaffolding improvement can be expressed as a self-induced update in which an *agent-executed* procedure produces a meta-level signal that is then applied as an intrinsic update:

$$\Sigma_{t+1} \ = \ \mathcal{U}_\Sigma\Big(\Sigma_{1:t},\ \mathcal{E}\big(\pi_{\theta_t,\Sigma_t}; \Sigma_t, \mathcal{C}_t\big)\Big), \qquad \theta_{t+1} = \theta_t, \tag{6}$$

where $\mathcal{E}$ denotes the execution of the agent (possibly under a meta-objective) to generate update-relevant artifacts, such as proposed prompt edits, memory reorganizations, tool-interface changes, or new control routines, and $\mathcal{U}_\Sigma$ denotes the system-level mechanism that commits these artifacts as structural scaffolding updates. Unlike the transient evolution of X_t, the update above is intended to endure beyond individual task boundaries.

These updates modify the induced policy $\pi_{\theta,\Sigma}$ by reshaping (i) the conditioning context supplied to the foundation model, (ii) the effective action space via admissibility constraints and tool schemas, and (iii) the execution semantics by which token sequences are parsed and grounded into environment actions. Compared to foundation model improvement, scaffolding improvement is typically faster, more reversible, and more context-dependent, yet it can produce substantial changes in the agent's effective behavior and problem-solving strategies.

Skill as a reusable update. The two modes of self-reference above realize self-improvement either indirectly, through self-generated signals that an update rule consolidates into the parameters θ, or directly, through actions that edit the scaffold Σ; a skill cuts across this distinction. We model a *skill* as a reusable instance of the self-induced update operator $\mathcal{U}$: a named update to the agent's own configuration $\mathcal{A}_t$ that it retains and reuses. Acquiring a skill serializes this update into one of $\mathcal{A}_t$'s substrates: a tool and its calling convention ($\mathcal{T}$), an instruction or workflow (p), a memory entry (m), consolidated weights (θ), or control logic (g). The skill's identity is the update it encodes; the substrate only names where it is stored. This is what makes "skill" orthogonal to the substrate axis of our taxonomy. A skill library is then a structured store of, and retrieval policy over, these serialized updates; the same store-and-retrieve structure recurs across substrates, which is why it surfaces in both tool routing and memory organization.

Reusability takes two forms. A skill may be invoked repeatedly, as a retained routine called many times across tasks, or applied once in the manner of an installer: a single update whose value lies in the persistent change it leaves behind, but which remains a portable artifact, reusable across agents and sessions. Either way, a skill is a first-class, serialized operator, in contrast to an ad hoc one-off action that leaves no retained trace.

We further distinguish two scopes according to what an invoked skill acts on. An *object-level* skill acts on the task or world state: invoking it runs a (typically multi-step) routine through the execution operator $\mathcal{E}$ to carry out a sub-task (e.g., a learned `collect-wood` routine), the agentic analog of a temporally extended option in hierarchical reinforcement learning (Sutton et al., 1999; Bacon et al., 2017). Here the $\mathcal{U}$ step lies in acquisition, which writes the routine into its substrate (e.g., $\mathcal{T}$), not in invocation. A *meta-level* skill instead acts on the agent's own configuration $\mathcal{A}_t$: invoking it edits a component of Σ_t or triggers a parameter update (e.g., writing a new tool, refactoring a prompt, consolidating experience into memory, or patching one's own scaffold). The installer-like skills above are inherently meta-level. For self-improvement, the meta-level scope is the central one. Because a meta-level skill both acts on $\mathcal{A}_t$ and is itself serialized back into $\mathcal{A}_t$, the operator can become part of its own operand. Once such a skill is in turn improved, we recover the self-referential loop in which the improver evolves together with the system it improves (Schmidhuber, 1987; 1993; 2003b).

Later sections describe how skills are serialized in recent works: as tools (Section 6.3), as memory workflows (Section 6.2); the application sections illustrate how acquired skills are invoked and reused across tasks.

3.3 Connections to Related Learning Paradigms

Because the SI-FMA paradigm is deeply rooted in established learning paradigms, examining it through the lens of these foundational theories offers critical insights. We map SI-FMA onto classical frameworks to clarify where self-improving FM-based agents inherit existing assumptions, where they recombine familiar mechanisms, and where agent architecture makes these mechanisms operationally distinct in practice.

Relation to Reinforcement Learning (RL). Classical reinforcement learning provides a natural reference frame for SI-FMA: it formalizes how an agent improves its policy through interaction, and recent work on Agentic RL (Zhang et al., 2026a) extends this view to foundation-model-based agents. Mapping SI-FMA onto this frame clarifies which channels of self-improvement inherit standard RL machinery and which lie outside its formulation. Specifically, updates to θ correspond to standard policy optimization under a fixed decision

process, whereas updates to Σ lie outside the standard RL formulation, reshaping the decision process in which the policy operates.

- **Parameter updates (θ).** When improvement targets the foundation model parameters using interaction trajectories, the process aligns with standard policy optimization. The foundation model acts as a large-scale policy network π_θ, and techniques such as RLHF (Christiano et al., 2017; Ouyang et al., 2022) or self-play optimize π_θ via algorithms like Proximal Policy Optimization (PPO) (Schulman et al., 2017) or Direct Preference Optimization (DPO) (Rafailov et al., 2023).
- **Scaffolding updates (Σ).** When improvement targets the scaffolding, the agent performs a form of structural meta-learning that has no direct counterpart in classical RL. First, whereas classical RL assumes a fixed action space and state representation, updating Σ (e.g., adding tools or modifying memory) dynamically alters the effective state-action space and the observation processing logic, thereby reshaping the underlying Markov decision process (MDP) itself. Second, the optimization target itself differs in kind: Σ comprises discrete, structured artifacts such as prompts, memory entries, and routing rules, which are typically updated through search, generation, or symbolic edits rather than gradient descent, and which remain explicit and inspectable rather than absorbed into network weights.

SI-FMA also departs from classical RL along a second axis: the source of the learning signal. While classical RL relies on external scalar rewards, self-improving agents increasingly utilize self-generated supervision, where the agent acts as its own critic to synthesize feedback signals from its own trajectories, judgments, or verifier modules.

Relation to Online Learning. Online learning (Littlestone, 1988; Hoi et al., 2021) studies sequential decision or prediction under a data stream (Qi et al., 2025), where the learner updates each round and performance is assessed through cumulative loss or regret. SI-FMA intersects with this view when an agent is updated repeatedly during deployment and evaluation tracks an improvement trajectory rather than a single endpoint, though it is not restricted to the online setting—many self-improvement pipelines operate in batched or offline regimes. Where the paradigms meet, the two channels relate to online learning asymmetrically: θ updates recover the classical view of updating a hypothesis under a non-stationary stream, whereas Σ updates enact a system-level analogue that shifts adaptation to explicit, inspectable components.

- **Parameter updates (θ).** These inherit the standard stability challenges of online learning, including distribution shift and catastrophic forgetting. Systems mitigate these risks through controlled update operators such as replay or rehearsal (Robins, 1995), regularization (Kirkpatrick et al., 2017; Ramesh et al., 2026), parameter-efficient tuning (Hu et al., 2022), and explicit versioning with rollback.
- **Scaffolding updates (Σ).** These provide a channel for rapid adaptation through prompting policies, memory read-write rules, tool routing, and orchestration logic. Externalizing adaptation in this way improves transparency and control, but does not eliminate forgetting and introduces its own failure modes such as memory poisoning, drift in tool semantics, and brittle template dependence.

Relation to Active Learning. Active learning studies how a learner selects queries under a labeling budget to maximize information gain, typically by requesting labels from an external oracle (Ren et al., 2021). SI-FMA intersects with this view when an agent actively controls its data-acquisition process—for example, by targeting frequent failure modes, seeking environments that maximize verifier disagreement, or explicitly requesting human feedback. However, because modern agents often rely on self-generated verification or critiques rather than oracle labels, this behavior transcends standard active learning. It connects SI-FMA directly to classical artificial-curiosity frameworks, in which systems are explicitly designed to actively construct experiments that maximize learning progress, Bayesian surprise, or compression progress (Schmidhuber, 1991c; Storck et al., 1995; Schmidhuber, 2006b; 2010; Herrmann & Schmidhuber, 2026). This perspective places self-improvement within a broader family of curiosity-driven exploration mechanisms, emphasizing that the

key design choices are the query policy, the intrinsic objective, and the trustworthiness of the resulting feedback.

4 A Taxonomy of Existing Approaches

Building on the definitions introduced in Section 3, we now introduce a taxonomy to organize existing approaches to self-improvement in FM-based agents. This section serves as a *methodological classification* that provides a common reference frame for comparing a rapidly growing and heterogeneous literature. In the remainder of this section, we use $\texttt{IMPROVE}_{target}(\,\cdot\,; \mathcal{S}_t)$ to denote an abstract self-improvement procedure in the sense of SI-FMA. Here $\mathcal{S}_t$ denotes an update signal produced through the agent's own execution (the operator $\mathcal{E}$ in Section 3), such as interaction trajectories, critiques, preferences, or other self-generated artifacts. This notation organizes existing approaches first by the target of modification, separating foundation-model updates from scaffolding updates. Within each target, approaches are further distinguished by the form of the self-induced learning signal that drives the update. For scaffolding improvement, we additionally group methods by the scaffold component being modified, such as prompts, memory, tools, or full scaffolding.

4.1 Foundation Model Improvement

Foundation model improvement targets the parameters of the underlying foundation model while leaving the agent-level scaffold unchanged:

$$\theta_{t+1} = \texttt{IMPROVE}_{\theta}(\theta_{1:t}; \mathcal{S}_t), \qquad \Sigma_{t+1} = \Sigma_t. \tag{7}$$

In this paradigm, the agent's own execution under its induced policy π_{θ_t,Σ_t} generates learning signals that are subsequently consumed by a parameter-update procedure. By committing the update directly into θ_t, the agent internalizes the learned capabilities. This allows the adaptation cost to be amortized across future interactions, though such parametric updates typically operate on longer time scales and incur higher computational overhead. $\theta_{1:t}$ denotes the parameter history, enabling validation and rollback (e.g., reverting to a prior checkpoint) when a proposed update degrades performance or violates constraints.

Classification by signal form. As detailed in Section 5, we further categorize foundation model improvement according to the form of the self-induced signal $\mathcal{S}_t$:

- **Intrinsic Generative Demonstrations** (primary update signal: $\mathcal{D}_t$): the agent synthesizes explicit training instances (e.g., demonstrations or augmented datasets) that are used for supervised-style learning (Wang et al., 2022; Lu et al., 2024b; Zhao et al., 2024b; Shi et al., 2025a).
- **Intrinsic Evaluative Feedback** (primary update signal: e_t): the agent constructs supervisory signals such as scalar rewards, preference pairs, or critiques to guide optimization and alignment (Gong et al., 2026; Huang et al., 2025c; Bai et al., 2022; Bensal et al., 2025; Zhang et al., 2025c).
- **Extrinsic Exploratory Experience** (primary update signal: τ_t): the agent learns from interaction trajectories and environment-grounded outcomes produced during execution (Wei et al., 2025a; Dong et al., 2025; Da et al., 2025; Sun et al., 2025b).

4.2 Scaffolding Improvement

Scaffolding improvement targets the agent's operational scaffold while keeping the foundation model parameters fixed:

$$\Sigma_{t+1} = \texttt{IMPROVE}_{\Sigma}(\Sigma_{1:t}; \mathcal{S}_t), \qquad \theta_{t+1} = \theta_t. \tag{8}$$

Here, IMPROVE commits structural changes into $\Sigma_t = (p_t, m_t, \mathcal{T}_t, g_t)$, thereby reshaping how the frozen model is conditioned, grounded, and constrained during all subsequent executions. This paradigm typically yields fast, reversible, and task-specific adaptation without the risks of catastrophic forgetting.

Classification by scaffold component. Crucially, each of the following modifications represents a structural update to the agent's intrinsic configuration, distinguishing it from transient shifts in working memory. As detailed in Section 6, we further decompose scaffolding improvement according to which component of Σ_t is targeted:

- **Prompt Optimization ($p_t \to p_{t+1}$):** edits to structured prompts or in-context exemplars, $p_{t+1} = \texttt{IMPROVE}_p(p_{1:t}; \mathcal{S}_t)$ (Yuksekgonul et al., 2024; Guo et al., 2024; Fernando et al., 2024; Zhou et al., 2022).
- **Memory Evolution ($m_t \to m_{t+1}$):** updates to how experience is stored, consolidated, and retrieved, $m_{t+1} = \texttt{IMPROVE}_m(m_{1:t}; \mathcal{S}_t)$ (Chhikara et al., 2025; Wang et al., 2025j; Zhang et al., 2025e; Zhong et al., 2024).
- **Tool Governance ($\mathcal{T}_t \to \mathcal{T}_{t+1}$):** refinement or expansion of the agent's action space through tool creation or selection, $\mathcal{T}_{t+1} = \texttt{IMPROVE}_{\mathcal{T}}(\mathcal{T}_{1:t}; \mathcal{S}_t)$ (Haque et al., 2025; Wang et al., 2025f; Dong et al., 2025; Zhang et al., 2025j).
- **Full Scaffolding Update ($\Sigma_t \to \Sigma_{t+1}$):** holistic reconfiguration of the agent's operational architecture, $\Sigma_{t+1} = \texttt{IMPROVE}_{\Sigma}(\Sigma_{1:t}; \mathcal{S}_t)$ (Hu et al., 2025; Zhang et al., 2026d; Novikov et al., 2025; Xia et al., 2025).

5 Foundation Model Improvement

Algorithm 1: Foundation-Model Improvement

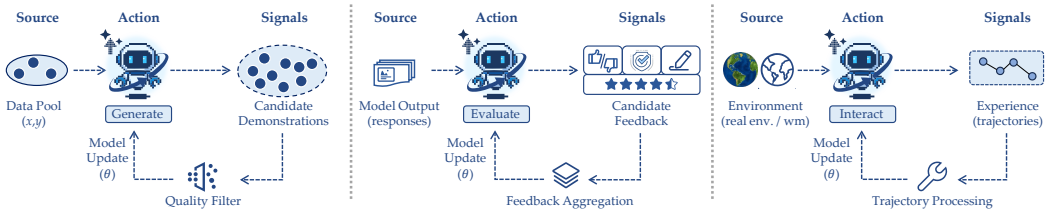
Input : Initial agent state $\mathcal{A}_0 = (\theta_0, \Sigma)$; maximum iterations T
Output: Improved agent $\mathcal{A}_T = (\theta_T, \Sigma)$

for $t \leftarrow 0$ **to** $T-1$ **do**
  // 1) Construct or collect learning signal $\mathcal{S}_t$
  $\mathcal{S}_t \leftarrow \emptyset$
  if *subcategory includes intrinsic generative demonstrations* **then**
    $\mathcal{S}_t \leftarrow \mathcal{S}_t \cup \textsc{GenerateDemonstrations}(\mathcal{A}_t)$; // §5.1
  if *subcategory includes intrinsic evaluative feedback* **then**
    $\mathcal{S}_t \leftarrow \mathcal{S}_t \cup \textsc{GenerateEvaluativeFeedback}(\mathcal{A}_t)$; // §5.2
  if *subcategory includes extrinsic exploratory experience* **then**
    $\mathcal{S}_t \leftarrow \mathcal{S}_t \cup \textsc{CollectExperience}(\mathcal{A}_t, \text{env})$; // §5.3
  // Optional: quality control / weighting of signals
  $\mathcal{S}_t \leftarrow \textsc{FilterOrWeight}(\mathcal{S}_t)$
  // 2) Parameter update
  $\theta_{t+1} \leftarrow \textsc{Update}(\theta_{1:t}, \mathcal{S}_t)$
  // 3) State update
  $\mathcal{A}_{t+1} \leftarrow (\theta_{t+1}, \Sigma)$
  if $\textsc{Converged}(\mathcal{A}_{t+1})$ **then**
    break
return $\mathcal{A}_T$

Foundation-model-based self-improvement constitutes one of the most direct pathways for enhancing an agent's intrinsic capabilities. This approach targets the agent's core, namely the parameter set θ_t of its underlying FM, and updates these parameters to internalize new behaviors and reasoning patterns (Wang et al., 2022; Bai et al., 2022; Shafayat et al., 2025; Bensal et al., 2025; Dong et al., 2025; Wang et al., 2025i; Zhou et al., 2025b). Such updates are typically achieved through gradient-based optimization, resulting in stable changes to the model weights. In this sense, the FM is treated as a continually learnable system that stores improvements in its parametric memory, rather than relying solely on transient execution states. Because the model parameters compress the agent's learned representations and behavioral priors, updating them allows the agent to fundamentally refine its policy, reduce systematic errors, and improve alignment with target objectives. In practice, the agent effectively serves as its own source of supervision: it uses its own execution to generate training signals—such as demonstrations, evaluations, or interaction trajectories—and applies learning algorithms to these self-induced signals to systematically improve its capabilities over successive updates.

To systematically analyze FM-based self-improvement, we classify existing methods by the nature of how the learning signal $\mathcal{S}_t$ is used to update the foundation-model parameters under our formalism. Consistent with the history-aware operator in Section 3, we write the transition as $\mathcal{A}_{t+1} = \text{IMPROVE}(\mathcal{A}_{1:t}; \mathcal{S}_t)$ with $\theta_{t+1} = \text{IMPROVE}_\theta(\theta_{1:t}; \mathcal{S}_t)$ and $\Sigma_{t+1} = \Sigma_t$, allowing validation and rollback to prior checkpoints when necessary. This taxonomy yields three principal subcategories: (i) **Intrinsic generative demonstrations** (§5.1), where the learning signal is instantiated as $\mathcal{S}_t \approx \mathcal{D}_t$: the FM-based agent generates explicit examples, demonstrations, or task–solution pairs that are used for parameter updates. (ii) **Intrinsic evaluative feedback** (§5.2), where the learning signal is instantiated as $\mathcal{S}_t \approx e_t$: the FM-based agent or an internal evaluator judges candidate behavior through scores, preferences, confidence estimates, consistency signals, critiques, or revisions, which are then used to update the foundation model. (iii) **Extrinsic exploratory experience** (§5.3), where the learning signal is instantiated as $\mathcal{S}_t \approx \tau_t$: the agent collects trajectories, rewards, observations, or

executable outcomes through interaction with external or simulated environments. These subcategories are not mutually exclusive in deployed systems. Many practical pipelines mix intrinsic demonstrations, intrinsic evaluations, and exploratory experience within a single improvement cycle; for clarity, we categorize methods by the dominant signal source and objective used in their update operator. Algorithm 1 summarizes the generic update loop of foundation-model improvement under self-generated learning signals.



(a) Intrinsic Generative Demonstrations (b) Intrinsic Evaluative Feedback (c) Extrinsic Exploratory Experience

Figure 6: Overview of foundation model improvement under agent-induced learning signals. The agent improves the foundation model by generating intrinsic demonstrations, producing intrinsic evaluative feedback, or collecting extrinsic exploratory experience, each forming a distinct parameter-update loop.

5.1 Intrinsic Generative Demonstrations

The remarkable capabilities of modern foundation models are fundamentally driven by their massive parameter scales, which inherently demand vast quantities of high-quality training data to continuously optimize and align (Zhao et al., 2024b; Gan & Liu, 2025). The methods in this category fall into the group of internal generative processes, where FMs act simultaneously as the cognitive learner and the data synthesizer (Zhao et al., 2024b; Zweiger et al., 2025). Leveraging the semantic priors already compressed within their weights, FM-based agents can autonomously construct intrinsic generative demonstrations in forms such as instruction-response pairs, reasoning trajectories, and execution logs, without requiring new external observations (Zhao et al., 2024b; Shi et al., 2025a; Simonds & Yoshiyama, 2025). By casting the agent as its own data provider, this paradigm shifts the bottleneck of parameter updates from the costly acquisition of human annotations to the design of effective generation strategies and internal filtering mechanisms (Qin et al., 2025).

Formally, as illustrated in Fig. 6 (a), under foundation-model improvement we treat these intrinsic generative demonstrations as the learning signal, i.e., $\mathcal{S}_t \approx \mathcal{D}_t$. At iteration t, the agent state $\mathcal{A}_t = (\theta_t, \Sigma_t)$ induces a generative distribution $\mathcal{P}_{\text{gen}}(x, y \mid \mathcal{A}_t)$ and samples an intrinsic dataset

$$\mathcal{D}_t^{\text{gen}} = \{(x_i, y_i)\}_{i=1}^{n_t} \sim \mathcal{P}_{\text{gen}}(\cdot \mid \mathcal{A}_t). \tag{9}$$

Optionally, a quality-control operator Φ_t filters or weights examples (Jiang et al., 2025) to yield $\tilde{\mathcal{D}}_t^{\text{gen}} = \Phi_t(\mathcal{D}_t^{\text{gen}})$. The effective training set (and hence the learning signal) is then $\mathcal{D}_t = \mathcal{D}_t^{\text{base}} \cup \tilde{\mathcal{D}}_t^{\text{gen}}$, where $\mathcal{D}_t^{\text{base}}$ denotes any existing data.

The parameter update can be written compactly as $\theta_{t+1} = \texttt{IMPROVE}_\theta(\theta_{1:t}; \mathcal{D}_t)$, where $\texttt{IMPROVE}_\theta$ denotes the outcome of optimizing an empirical objective on $\mathcal{D}_t$ (typically initialized at the current active checkpoint θ_t). We write this objective as $\mathcal{L}(\theta; \mathcal{D}_t)$, with $\Omega(\theta, \theta_0)$ an optional regularizer around an initialization θ_0 and $\lambda \geq 0$ its coefficient:

$$\theta_{t+1} = \arg\min_\theta \; \mathcal{L}(\theta; \mathcal{D}_t) + \lambda\, \Omega(\theta, \theta_0). \tag{10}$$

In practice, $\texttt{IMPROVE}_\theta$ is implemented via iterative fine-tuning with a gradient-based optimizer. Let $\theta_t^{(0)} = \theta_t$, and let $\mathcal{B}_t^{(k)}$ denote a minibatch drawn from $\mathcal{D}_t$ according to the weighting induced by Φ_t. Using step size η_k at inner step k and the gradient operator ∇_θ with respect to θ, for $k = 0, \ldots, K_t - 1$,

$$\theta_t^{(k+1)} = \theta_t^{(k)} - \eta_k \nabla_\theta \mathcal{L}\big(\theta_t^{(k)}; \mathcal{B}_t^{(k)}\big), \tag{11}$$

and the outer-loop update is $\theta_{t+1} = \theta_t^{(K_t)}$. To systematically explore this approach, we consider three key aspects: the strategy used to generate these intrinsic demonstrations, the format of the generated data, and the challenges that arise in closing the self-improvement loop.

Generation strategies. The strategy of data generation is a critical factor for the quality and scalability of a self-improvement loop. The underlying techniques start with a small set of example seeds and then build a larger instruction-output pair corpus.(Wang et al., 2022; Zhao et al., 2024b). Methods such as Evol-Instruct (Xu et al., 2024a) go beyond simple volume expansion. They drive complexity evolution by using LLM to rewrite instructions and gradually increase the difficulty of the instructions, thereby overcoming the limitations of humans in building highly complex scenes. To improve data quality, more sophisticated methods have added a refining or filtering stage. Huang et al. (2023) employs self-consistency to filter high-confidence inference paths, while Singh et al. (2024) uses external verifiers like unit tests to select only correct solutions from the model's attempts. In addition to bootstrapping and filtering, some approaches conceptualize data generation as a form of curriculum learning. Simonds & Yoshiyama (2025) create a learning curriculum by recursively decomposing complex problems into simpler sub-problems, and Qin et al. (2025) explicitly introduce mechanisms to expand the sample pool to counteract the risk of diminished output diversity over time. More recently, Test-Time Self-Improvement (TT-SI) (Acikgoz et al., 2025) shifts the paradigm from massive offline generation to highly sample-efficient, on-the-fly adaptation. By detecting weak cases at inference time via uncertainty estimation, the agent self-generates targeted training examples for its specific blind spots and performs targeted low-rank adaptation (LoRA) fine-tuning; this yields immediate, per-instance improvements with a fraction of the data cost.

Each of these strategies has distinct strengths and failure modes. For example, self-consistency filtering rests on the assumption that correct reasoning tends to be self-agreeing. It can break down when the model is confidently wrong, because repeated reasoning may repeatedly converge to the same incorrect conclusion (Chen et al., 2023; Hamidieh et al., 2026). Curriculum-based generation can significantly improve learning performance for complex tasks, but it becomes vulnerable when the chosen decomposition method is flawed or ignores necessary contextual information. For example, this can occur when the sub-problems are constructed in a way that no longer preserves the constraints of the original task (Zhou et al., 2023; Xu et al., 2024a). Interactive refinement also depends on the model's ability to detect and correct its own errors. When certain types of errors are beyond the model's perception range, the improvement process may exacerbate these errors rather than eliminate them (Shinn et al., 2023; Huang et al., 2024a). Therefore, choosing an appropriate generative strategy requires matching the method to the model's current capabilities while avoiding mechanisms that focus on or amplify existing blind spots in the model.

Data formats. These intrinsic demonstrations take various forms, ranging from simple input-output pairs to complex, structured artifacts designed to supervise specific cognitive abilities. Basic forms include simple instruction-response pairs, often with explicit reasoning traces added (e.g., chain-of-thought explanations) to make the model's intermediate reasoning steps transparent and trainable (Huang et al., 2023). However, in scenarios involving multi-step decision-making or tool usage, agents often struggle with long-term tasks because suboptimal behaviors accumulate, eventually causing them to deviate from the correct path. To address this issue, the generated data should capture structured trajectories, rather than the final result. For example, tool application programming interface (API) calls or code execution sequences, combined with intermediate environment feedback and validation labels (Shi et al., 2025a; Singh et al., 2024), can provide fine-grained supervision, teaching agents how to handle dependencies and recover from errors.

This inherent generative paradigm also extends to multimodal scenarios, where agents synthesize training samples, combining visual inputs such as images or videos with textual descriptions and reasoning to improve performance on tasks including visual reasoning (Zhang et al., 2025k). At a higher level of abstraction, these demonstrations can take the form of metacognitive artifacts. For example, decomposed problem trees (Simonds &

Yoshiyama, 2025) and self-editing instructions (Zweiger et al., 2025), both of which can support complex planning and self-modification. More generally, the choice of data format should be consistent with the target capability, as different formats encode different types of supervisory information. Inference trajectories are suited for logical reasoning, code combined with tests is well-suited for programming, and structured action logs provide supervision for tool usage.

Challenges and safeguards. This intrinsic generative paradigm still faces considerable challenges despite its success. First, recursively training the model on the generated corpus introduces the risk of model collapse and forgetting (Shumailov et al., 2024b;a). If the model generates defective, biased, or erroneous samples, it will internalize these errors during fine-tuning, creating a negative feedback loop that degrades model performance (Zhao et al., 2024b; Wang et al., 2022). Insufficient diversity in generated demonstrations also narrows the solution space and leads to pattern collapse during iterations. In fact, agents may get trapped in knowledge bubbles and repeatedly generate data, which reinforces their existing biases and traits instead of generating genuine new capabilities (Wu et al., 2025c; Yuan et al., 2025).

To make these inherent generation loops more stable and reliable, several safeguards have been proposed. A simple and effective safeguard is to retain artificially generated benchmark data and accumulate generated demonstration data on top of it, which can prevent collapse under repeated training (Gerstgrasser et al., 2024). Another work strengthens quality control by using more robust checkers to verify generated samples. These checkers include external validators specifically designed for model-generated content (Yi et al., 2026) and formal systems for inference verification, such as theorem provers (Leang et al., 2025). Complementary data-centric safeguards focus on selecting and filtering the training corpus to remove low-quality or harmful examples. To address diversity loss and pattern collapse, diversity-aware pooling expansion and selection mechanisms have been shown to maintain exploratory nature during the iterative process (Qin et al., 2025). Furthermore, shifting from large-scale offline generation to proactive, uncertainty-driven generation during testing naturally mitigates the accumulation of redundant data by focusing computational resources only on out-of-distribution or challenging samples (Acikgoz et al., 2025). Finally, because automated evaluators and validation processes can be unreliable or drift over time, recent research emphasizes human auditing and human-computer interaction to improve evaluation criteria as practical safeguards (Dietz et al., 2025; Do et al., 2025).

5.2 Intrinsic Evaluative Feedback

Just as massive parameter scales demand vast quantities of demonstration data, aligning and refining these capabilities requires high-quality supervisory signals. The bottleneck in traditional supervisory signal acquisition lies in the high cost of manually labeled preferences and human evaluation. To overcome this limitation, the methods in this category drive a paradigm shift by reframing the acquisition of supervision as an internal evaluative process. In this regime, FMs act not only as the output generator and the self-evaluator, but ultimately as the cognitive learner that internalizes its own judgments. Leveraging their language-native capabilities to follow rubrics, compare alternatives, and articulate reasoning, FM-based agents can autonomously produce *intrinsic evaluative feedback*, such as scalar scores, preference pairs, consistency signals, and natural language critiques, without requiring new environment interaction. Unlike intrinsic generative demonstrations (Section 5.1) that provide examples, or extrinsic exploratory experience (Section 5.3) derived from environment-grounded outcomes, the dominant learning signal here is the agent's endogenous judgment of its own candidate behaviors. By treating the agent as a critic of itself, this paradigm shifts the bottleneck of parameter updates to the design of robust internal evaluation rubrics and critique mechanisms.

Formally, let $\mathcal{A}_t = (\theta_t, \Sigma_t)$ denote the current agent configuration. Given an input or task context x, the agent samples a set of candidate outputs

$$\mathcal{Y}_t(x) = \{y_t^{(1)}, \ldots, y_t^{(K)}\}, \qquad y_t^{(k)} \sim \pi_{\theta_t, \Sigma_t}(\cdot \mid x). \tag{12}$$

An intrinsic evaluator ϕ_t, which may be implemented by the current model, an auxiliary judge model, a learned reward model, or a fixed scaffolded critique procedure, maps the task, candidates, and evaluation criteria κ_t into an evaluative signal:

$$e_t = \phi_t(x, \mathcal{Y}_t(x); \kappa_t). \tag{13}$$

Here, ϕ_t is the source of feedback rather than the target of the update in this subsection. When an auxiliary evaluator is itself trained, we treat it as part of the feedback-construction pipeline; the method is classified here because the resulting evaluative signal is used to update the foundation model. The signal e_t may instantiate a scalar reward r_t, a preference relation $y^+ \succ y^-$, a confidence or uncertainty score, a textual critique c_t, or a refined target y_t^*. Under foundation-model improvement, this feedback becomes the update signal, $\mathcal{S}_t \approx e_t$, and the parameter update is written as

$$\theta_{t+1} = \texttt{IMPROVE}_\theta(\theta_{1:t}; e_t), \qquad \Sigma_{t+1} = \Sigma_t. \tag{14}$$

Depending on the form of e_t, the update may be implemented through reinforcement learning, preference optimization, reward-model training, critique-conditioned fine-tuning, or supervised fine-tuning on revised outputs.

Rubric feedback. A first family of methods derives evaluative feedback by asking a model to judge candidate outputs against explicit criteria. The criteria may be task instructions, grading rubrics, safety principles, constitutional rules, or domain-specific preferences. In this setting, the evaluation standard κ_t serves as the context for judgment: candidate outputs are scored, ranked, or compared according to how well they satisfy the stated criteria. The evaluator may also produce a brief rationale for its judgment, but the primary learning signal is the score, ranking, or preference relation rather than an open-ended revision.

Constitutional AI is a representative early example of this pattern. Instead of relying only on direct human preference labels, the model critiques and ranks outputs according to a set of written principles, producing AI feedback that can be used to train a preference model and subsequently optimize the policy through reinforcement learning (Bai et al., 2022). Recent self-improvement methods extend this judge-based loop more directly into parameter-updating systems. Meta-Rewarding trains a model to act, judge its own responses, and further evaluate its own judgments, turning both judgments and meta-judgments into preference pairs for iterative alignment improvement (Wu et al., 2025b). Simonds et al. (2025) show that LLM judges can provide reward signals without reference solutions, enabling reinforcement learning from model-generated judgments in reasoning domains where programmatic rewards are difficult to specify. Self-Evolved Reward Learning further studies a learned reward model that labels additional data for its own improvement and supports reinforcement learning from self-feedback (Huang et al., 2025a). Together, these methods illustrate how explicit principles, rubrics, or judge prompts can transform natural-language evaluation standards into scalable supervision for foundation-model improvement. The main advantage is flexibility, since the same model-based evaluator can be conditioned to assess helpfulness, harmlessness, factuality, reasoning quality, or format compliance. The main limitation is reliability, since ambiguous criteria or biased judges may reward superficial compliance rather than genuine improvement.

Consistency feedback. A second family of methods exploits the stochastic behavior of foundation models to derive feedback signals for self-improvement. When ground-truth labels or reliable external verifiers are unavailable, the agent can generate multiple candidate solutions for the same task and use agreement among them as an intrinsic signal. The underlying assumption is not that consistency guarantees correctness, but that agreement, entropy, or self-certainty can provide a useful weak signal for ranking or weighting candidates.

Concretely, the agent samples $\mathcal{Y}_t(x) = \{y_t^{(1)}, \ldots, y_t^{(K)}\}$ and applies an aggregation operator C to produce a confidence or reward-like signal:

$$e_t = C(\mathcal{Y}_t(x)). \tag{15}$$

For example, majority voting can identify a consensus answer, predictive entropy can be used as a confidence score, and agreement among independent reasoning paths can be used

to construct preference or reward signals. TTRL (Zuo et al., 2025) uses majority voting over multiple generated answers at test time to produce reward signals for reinforcement learning. SRT (Shafayat et al., 2025) similarly investigates whether self-consistency can support self-improvement without ground-truth labels. EMPO (Zhang et al., 2025h) encourages reasoning behavior through entropy-based signals, while INTUITOR (Zhao et al., 2025c) uses self-certainty as an intrinsic reward. In these methods, the agent does not primarily learn from newly generated demonstrations; rather, it learns from evaluative structure extracted from its own distribution over answers.

The strength of this family is that it can be applied broadly, including in domains where labels are expensive and formal verification is unavailable. However, consistency is only a proxy for correctness. If a model is systematically biased or confidently wrong, repeated sampling may amplify the same error. Confidence-derived feedback also depends on calibration: a model's expressed certainty may not align with actual correctness. These issues make consistency-based self-improvement methods, while useful, quite fragile, especially when model errors are correlated across different samples. (Tan et al., 2025a; Feng et al., 2025; Till et al., 2025; Hamidieh et al., 2026).

Corrective feedback. A third family of methods treats natural-language critique or modification itself as the core evaluation outcome. Unlike rubric-based judging (whose main output is a score or preference under a stated criterion), error-correcting feedback methods require the model to identify specific flaws in candidate outputs and propose modifications. Therefore, this feedback is corrective, not merely comparative.

Let R denote a critique-and-revision operator. Given a candidate output y_t, the agent produces

$$e_t = R(x, y_t), \tag{16}$$

where e_t is composed of c_t and y_t^*, c_t is a critique and y_t^* is a revised output. The pair (y_t, y_t^*) can instantiate a preference signal $y_t^* \succ y_t$, while the critique c_t can be used as explanatory supervision or as a natural-language training signal. "ReST meets react" (Aksitov et al., 2024) combines agentic reasoning and self-training to improve multi-step problem solving. SELF (Lu et al., 2024b) uses language feedback to iteratively refine generated answers, transforming model-produced critiques and revisions into signals for self-improvement. RISE (Qu et al., 2024) uses interaction-based recursive self-reflection and fine-tunes based on improved predictions. Reflect, Retry, Reward (Bensal et al., 2025) follows a similar loop in which the model reflects on failures, retries the task, and uses the resulting feedback to guide reinforcement learning. The AlphaAllM approach (Tian et al., 2024) further integrates search and critique, using model-generated evaluations during the search process to construct stronger training signals. These methods exploit the ability of foundation models to provide semantic, actionable, and informative feedback that is richer than simple scalar rewards. A critique can identify missing constraints, unsupported claims, incorrect reasoning steps, or unsafe implications, thus providing a richer signal for parameter updates. This flexibility also introduces failure modes since the model may produce plausible but incorrect critiques, or learn to satisfy the surface form of the critique rather than its underlying goals. Therefore, critique-based feedback is more reliable for parameter-level self-improvement when combined with safeguards such as comparing the original and modified outputs, filtering low-quality critiques, using heterogeneous critique models, or combining internal critiques with external validation when available.

Trade-offs and safeguards. The advantage of intrinsic evaluation feedback lies in reducing reliance on human annotation and converting the agent's own output into a learning signal that can be directly used by the parameter update algorithm. Its main risk is that the evaluator is often tightly coupled with the policy to be improved, so this loop may reinforce common blind spots, reward outputs that conform to the model's existing preferences, overfit superficial evaluation criteria, or become biased with repeated use and reinterpretation of the evaluation criteria. Signals based on confidence and consistency increase vulnerability when the model is poorly calibrated or when there are correlations between its samples. Therefore, reliable use of this paradigm requires safeguards such as separating the generator and evaluator at different checkpoints or model families, maintaining external anchors by

retaining human annotations or context-based validation, treating disagreements between evaluators as signals of uncertainty, and regularly reviewing the scoring criteria, rewarding the model, and evaluating quality. In practice, intrinsic evaluation feedback can be viewed as a component of a broader improvement loop, supplemented by demonstrations, external validation, and exploratory experiences, rather than as the only source of supervision.

5.3 Extrinsic Exploratory Experience

The preceding two subsections focused on intrinsic self-improvement signals, where the agent improves the foundation model using its own generated demonstrations (§5.1) or evaluative judgments (§5.2). Extrinsic exploratory experience differs in that the learning signal is grounded in what happens after the agent acts. Under our foundation-model improvement formalism, the learning signal is instantiated as experience, $\mathcal{S}_t \approx \tau_t$, where τ_t denotes trajectories collected by executing the policy π_{θ_t,Σ_t} in a task environment or its learned proxy. The parameter update can be written as $\theta_{t+1} = \text{IMPROVE}_\theta(\theta_{1:t}; \tau_t)$, with $\Sigma_{t+1} = \Sigma_t$. While this view connects self-improvement back to the classical reinforcement learning framework (Silver & Sutton, 2025; Zhao et al., 2024a), experience for foundation-model agents is not just a stream of state-action-reward tuples (s, a, r, s'). A trajectory may contain web pages, screenshots, code logs, compiler errors, tool calls, and intermediate reasoning traces. Because these artifacts are readable by foundation models, the same experience can be flexibly reused across reinforcement learning, supervised fine-tuning, preference construction, and failure analysis. However, acquiring and utilizing this rich experience introduces distinct difficulties: interaction can be slow or costly, rewards may be sparse or delayed, verifiers can be gamed, and learned world models (Schmidhuber, 1990; 2015; Nanbo et al., 2025) may produce plausible but counterfactual transitions.

To address these challenges, we organize existing methods by how the exploratory experience is obtained. For interaction with grounded task environments, the agent acts in real, sandboxed, or rule-based task settings, and the learning signal comes directly from the task environment (e.g., state changes, unit tests, or task-specific verifiers). For interaction with simulated proxy environments, a learned world model serves as a proxy for the task environment, generating predicted states, rollouts, or outcomes for policy improvement. Notably, these two modes are not mutually exclusive. In fact, as demonstrated by early controller-world-model architectures (Schmidhuber, 1990), a system may use grounded interaction to collect data and update its world model, and subsequently use simulated interaction to plan, explore, or improve its policy.

5.3.1 Interaction with Grounded Task Environments.

In this mode, the agent learns by directly acting in a task environment. Typical settings include code interpreters for coding agents, web APIs for web agents, mobile user interfaces (UIs) for GUI agents, and physical environments for embodied agents. The learning signal comes from the environment's response—such as state changes, execution traces, or unit tests. Training then proceeds by collecting these interaction trajectories and updating the foundation-model policy using reinforcement learning methods like PPO (Schulman et al., 2017), or preference-based objectives like DPO (Rafailov et al., 2023) when successful and unsuccessful trajectories can be contrasted. A useful way to organize this line of work is by the source of feedback:

(1) Programmatic verifiers provide the clearest form of grounded signal. When a task admits an executable check, the agent can be trained without learning a separate reward model. Code generation is the canonical case, since unit tests provide direct pass or fail feedback for proposed programs. *Agent-RLVR* (Da et al., 2025), for example, uses unit-test outcomes to guide policy optimization, contrasting successful programs against failed attempts. The same pattern extends beyond code whenever an output can be checked by an external procedure, including structured query language (SQL) execution, theorem proving, tool use with checkable post-conditions, and structured tasks with deterministic success criteria.

(2) Learned reward models evaluate trajectories collected from the task environment. In this setting, the task environment naturally produces the trajectory, while the learned evaluator

only scores its success. *WebRL* (Qi et al., 2025) trains an outcome-supervised reward model to label web-navigation trajectories automatically, reducing reliance on hand-crafted success criteria or human annotation. *UI-Genie* (Xiao et al., 2025) couples policy learning with reward-model refinement, using validated trajectories to train the agent and step-level labels from both successful and failed trajectories to improve the evaluator. Related work such as *MobileGUI-RL* (Shi et al., 2025b) adapts Group Relative Policy Optimization (GRPO) to mobile GUI navigation with trajectory-aware advantages and rewards that combine task success with execution efficiency. Across these methods, the feature is that feedback is computed over rich linguistic or visuolinguistic trajectories, which makes it natural to construct rewards, preferences, and step-level supervision from the same interaction record.

A third approach leverages *(3) self-generated tasks* to acquire grounded experience. In these systems, the agent may propose new tasks or goals, but the learning signal remains extrinsic because candidate solutions are accepted or rejected by execution, verification, or environmental feedback. *Absolute Zero* (Zhao et al., 2025a) uses self-play to generate tasks and solutions in an open-ended environment, while execution-based validation determines which solutions are retained for learning. *ETO* (Song et al., 2024c) learns more conservatively from contrasts between successful and failed trajectories in fixed environments. These methods blur the boundary between intrinsic and extrinsic signals at the level of task selection, but not at the level of supervision. The agent may choose what to try, while the environment determines whether the attempt succeeds.

Finally, standardized platforms have emerged to facilitate research across these grounded interaction modes. For example, *AgentGym* (Xi et al., 2024) provides unified APIs for interaction, evaluation, and training across multiple agent tasks. Such platforms significantly reduce the engineering cost of iterative experience collection, making it easier to compare reward sources, training objectives, and update procedures under shared environmental feedback.

5.3.2 Interaction with Simulated Proxy Environments

The simulated-interaction mode equips the agent with an internal predictive model of the environment. A typical pipeline first collects interaction trajectories from a task environment, then trains a world model to predict the environment's response to the agent's actions (Schmidhuber, 1990; Ha & Schmidhuber, 2018). Abstractly, this can be written as a learned dynamics model $W(s_{k+1}, r_k \mid s_k, a_k)$, which predicts the next state and reward from the current state and action at step k of a trajectory. The policy can then obtain additional experience by interacting with this learned proxy instead of repeatedly querying the original task environment. This internal simulation improves sample efficiency and reduces the cost or risk of exploration, especially when direct interaction is slow, expensive, or unsafe (Ding et al., 2025a; Richens et al., 2025).

While classical world models successfully predict transitions over both compact state vectors and raw pixel spaces (Ha & Schmidhuber, 2018), what distinguishes foundation-model agents is the extensive prior knowledge embedded in the learned dynamics. In FM-based systems, the world model is typically a generative language, vision-language, or video model that predicts high-dimensional observations—such as the next web page or video frame—in the exact format consumed by the policy. This design offers two practical advantages. First, large-scale generative pretraining provides a powerful prior over environment dynamics, significantly reducing the task-specific interaction needed to fit a useful proxy environment. Second, because generated observations lie in the same representation space as the policy input, simulated experience can be used directly without a separate representation-alignment step. This holds true for both linguistic web models and pixel-space video models like WMPO (Zhu et al., 2025b). Together, these advantages make FM-based world models effective for planning, trajectory synthesis, and failure analysis.

This pattern is most developed in web navigation, where direct interaction can be slow and search over action sequences can be expensive. *WebEvolver* (Fang et al., 2025b) trains a coevolving world model to predict next web observations and uses simulated rollouts to refine the agent policy. *WebSynthesis* (Gao et al., 2025b) uses a learned web world model for reversible, search-based trajectory synthesis, while *WebDreamer* (Gu et al., 2025) leverages

a web transition model for model-based planning to guide action selection. Beyond web navigation, *SPA* (Chen et al., 2025a) learns explicit state-estimation and transition models through self-play fine-tuning, using them to initialize and stabilize downstream policy optimization. In embodied control, *WMPO* (Zhu et al., 2025b) learns a pixel-space world model and optimizes the policy over imagined rollouts to avoid costly physical trial-and-error. A related direction uses structured memories of prior interaction rather than full generative dynamics. *GLoW* (Kim & seung-won hwang, 2026) maintains a dual-scale textual world memory—consisting of a global frontier of high-value discoveries and local advantage reflections—to guide a Go-Explore-style agent in text-based games, achieving strong performance with 100–800$\times$ fewer real environment interactions than RL baselines.

Challenges. Across extrinsic exploratory experience, foundation-model agents inherit classical difficulties of RL, including sparse and delayed rewards, low-throughput real interaction, overfitting to imperfect proxy environments. They also introduce failure modes specific to this setting. *Reward hacking through language* is more readily available than in classical RL: a foundation-model agent can satisfy the literal condition of a verifier (e.g., exploiting prompt loopholes in an LLM judge) without solving the underlying task, because the verifier's specification is itself a linguistic object that the agent can manipulate. *Capability regression* arises because extensive RL updates on narrow extrinsic rewards can erode the broader competencies the foundation model acquired in pretraining, a tension absent in agents trained from scratch. *Hallucinated dynamics* pose a distinctive risk for world-model approaches, since generative simulators can fabricate plausible but incorrect transitions that the policy then learns to exploit. Finally, the linguistic, multimodal nature of trajectories creates a *trajectory-length and context-window tension* unique to this setting: long-horizon experience must be compressed or summarized to fit within the model's context before it can be used as a learning signal. Current research therefore emphasizes more reliable extrinsic verifiers and reward models, world models with calibrated uncertainty over their own predictions, and training procedures that update the foundation model on extrinsic experience without sacrificing its general capabilities.

Takeaway

This section reviewed parameter-centric FM-based self-improvement, where agents refine their foundation models via intrinsic demonstrations, intrinsic evaluative feedback, or extrinsic experience. This paradigm allows learned behaviors to be broadly encoded into foundation model weights. However, success depends on managing the reliability of self-generated signals, mitigating distribution shift, and balancing the computational demands of iterative training.

6 Scaffolding Improvement

This section analyzes the scaffolding improvement paradigm for FM-based agents, where we approach this by decomposing the agent's operational scaffold into core components. This workflow begins with receiving and interpreting instructions, then contextualizes them using internal knowledge and memory, proceeds to executing actions in the world, and ultimately evolves the overall operational structure.

Formally, as established in Section 3, scaffolding improvement corresponds to transitions that modify the operational scaffold while explicitly keeping the foundation-model parameters frozen ($\theta_{t+1} = \theta_t$). Driven by an execution-derived learning signal $\mathcal{S}_t$ (e.g., task outcomes, critique, or execution errors), the update is formalized as $\Sigma_{t+1} = \texttt{IMPROVE}_\Sigma(\Sigma_{1:t}; \mathcal{S}_t)$. By maintaining a version history ($\Sigma_{1:t}$), the system inherently supports validation and rollback against harmful modifications across all components. We structure the discussion by increasing depth of architectural intervention. Concretely, we instantiate the scaffolding as $\Sigma_t := (p_t, m_t, \mathcal{T}_t, g_t)$, where p_t denotes the prompt template, m_t the internal memory, $\mathcal{T}_t$ the external tool set, and g_t the control logic. Algorithm 2 summarizes the generic improvement loop across these components:

- **Prompt-based improvement** (Section 6.1) targets the agent's input and instruction layer, formalized as $p_{t+1} = \texttt{IMPROVE}_p(p_{1:t}; \mathcal{S}_t)$. As the primary semantic interface through which the agent perceives its tasks, optimizing the prompt provides a direct way to refine task communication, objectives, and constraints without altering the internal parameters of the foundation model.
- **Memory-based improvement** (Section 6.2) equips the agent with an evolving internal cognitive resource, updated via $m_{t+1} = \texttt{IMPROVE}_m(m_{1:t}; \mathcal{S}_t)$. By dynamically storing, pruning, and retrieving past experiences, the agent shifts from memoryless execution to cumulative learning, supporting long-horizon reasoning and trustworthy adaptation.
- **Tool-based improvement** (Section 6.3) strengthens the agent's execution interface through the update $\mathcal{T}_{t+1} = \texttt{IMPROVE}_{\mathcal{T}}(\mathcal{T}_{1:t}; \mathcal{S}_t)$. By autonomously refining and managing external callable modules (e.g., web search, code interpreters), the agent translates internal decisions into precise, executable actions, effectively extending its capabilities beyond the native limits of the foundation model.
- **Full scaffolding improvement** (Section 6.4) represents the most profound level of architectural intervention, formalized as $\Sigma_{t+1} = \texttt{IMPROVE}_{\Sigma}(\Sigma_{1:t}; \mathcal{S}_t)$. By treating the entire codebase and operational logic as a mutable substrate, the agent dynamically reconfigures how its perception, reasoning, and execution faculties are integrated into a coherent whole.

Crucially, these intervention types are compositional rather than mutually exclusive: a single update can edit multiple scaffold components simultaneously, and full-scaffolding methods naturally subsume component-level edits while adding archive-based exploration and stronger acceptance tests.

6.1 Prompt

The prompt serves as the agent's core behavioral prior, defining how the foundation model parses its environment. Prompt optimization is therefore a central and highly accessible form of scaffolding improvement. While early methods relied on manual heuristic tuning, the field is rapidly developed toward automated, signal-driven improvement loops. Central to this development is the transition from scalar-based feedback to rich, structured linguistic critiques. By leveraging natural-language gradients, these systems now facilitate targeted, iterative updates that mirror the precision of gradient-based optimization in higher-dimensional strategy spaces.

Algorithm 2: Scaffolding Improvement

Input : Initial agent state $\mathcal{A}_0 = (\theta, \Sigma_0)$; max iterations T
Output: Improved agent $\mathcal{A}_T = (\theta, \Sigma_T)$

for $t \leftarrow 0$ **to** $T-1$ **do**
  // 1) Collect scaffolding learning signal $\mathcal{S}_t$
  $\mathcal{S}_t \leftarrow \textsc{InteractAndEvaluate}(\mathcal{A}_t, \text{env})$; // traces, critiques, success/failure, cost
  // Optional: quality control / weighting of signals
  $\mathcal{S}_t \leftarrow \textsc{FilterOrWeight}(\mathcal{S}_t)$
  // 2) Scaffolding update (keep θ fixed)
  if *subcategory = Full scaffolding* **then**
    $\Sigma_{t+1} \leftarrow \texttt{IMPROVE}_{\Sigma}(\Sigma_{1:t}; \mathcal{S}_t)$; // § 6.4
  else
    $p_{t+1} \leftarrow p_t$; $m_{t+1} \leftarrow m_t$; $\mathcal{T}_{t+1} \leftarrow \mathcal{T}_t$
    if *subcategory includes Prompt-based* **then**
      $p_{t+1} \leftarrow \texttt{IMPROVE}_p(p_t; \mathcal{S}_t)$; // § 6.1
    if *subcategory includes Memory-based* **then**
      $m_{t+1} \leftarrow \texttt{IMPROVE}_m(m_t; \mathcal{S}_t)$; // § 6.2
    if *subcategory includes Tool-based* **then**
      $\mathcal{T}_{t+1} \leftarrow \texttt{IMPROVE}_{\mathcal{T}}(\mathcal{T}_t; \mathcal{S}_t)$; // § 6.3
    $\Sigma_{t+1} \leftarrow (p_{t+1}, m_{t+1}, \mathcal{T}_{t+1})$
  // 3) State update
  $\mathcal{A}_{t+1} \leftarrow (\theta, \Sigma_{t+1})$
return $\mathcal{A}_T$

In this section, prompt primarily refers to structural instruction layers that are reused across interactions, such as system prompts or stable policy templates. A closely related line of work optimizes context construction, including exemplar selection, retrieval assembly, and the maintenance of long-term playbooks. Although both are scaffold-level updates, they differ in target: optimizing a system prompt alters the agent's core behavioral prior, whereas context optimization refines the dynamic conditioning of specific interactions. When discussing representative systems, we will explicitly indicate which object is updated.

As shown in Fig. 7, we categorize prompt refinement methods based on the form and richness of the learning signal $\mathcal{S}_t$. This yields four paradigms: (1) *Scalar-Feedback Optimization*, where $\mathcal{S}_t$ is a scalar performance score such as accuracy or reward; (2) *Qualitative-Feedback Refinement*, where $\mathcal{S}_t$ is a natural-language critique or suggestion for providing interpretable revision guidance; (3) *Population-Based Evolution*, where $\mathcal{S}_t$ consists of population-level

fitness evaluation and selection signals over a set of candidate prompts; and (4) *Textual Gradient Optimization*, where $\mathcal{S}_t$ is structured directional guidance, often expressed as a textual gradient that specifies how the prompt should be revised. As listed in Table 2, we further summarize representative systems and trade-offs across paradigms.

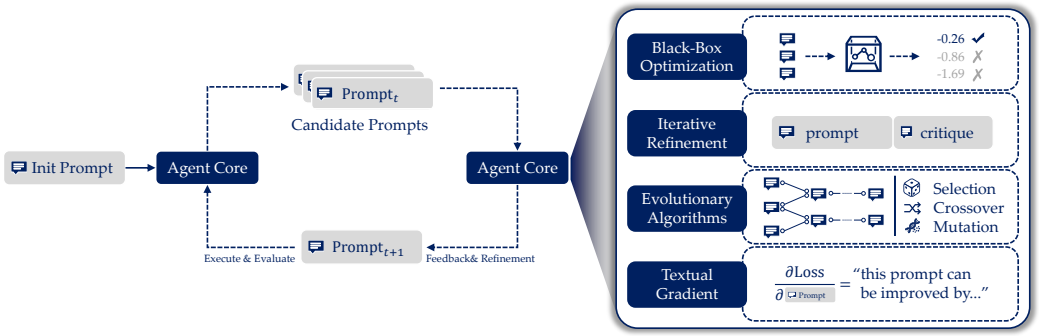


Figure 7: Prompt refinement as a self-improvement loop and its four paradigms, organized by the learning signal $\mathcal{S}_t$.

6.1.1 *Scalar-Feedback Optimization*

Early approaches to prompt optimization usually rely on task-level quantitative metrics to evaluate candidates (Zhou et al., 2022). The core objective can be formally defined as finding an optimal prompt p^* from a discrete space of possible prompts $\mathcal{P}$ that maximizes a scalar performance score (e.g., task accuracy or reward):

$$p^* = \arg\max_{p \in \mathcal{P}} f(p), \tag{17}$$

where $f(p)$ represents the evaluation function. In the context of our scaffolding improvement framework, this scalar score is used to constitutes the learning signal ($\mathcal{S}_t = f(p_t)$). Considering $\mathcal{S}_t$ is non-directional, providing only a magnitude of success without explanatory context, these methods typically rely on structured search algorithms to navigate the discrete text space. APE pioneered this paradigm by using an LLM to propose instruction candidates, subsequently selecting the best one based on empirical evaluation scores via simple search (Zhou et al., 2022). Building on this, OPRO formalized a more contextualized search by constructing a meta-prompt that includes a trajectory of previously evaluated prompts alongside their scalar scores, indirectly guiding the LLM to propose higher-scoring candidates in subsequent iterations (Yang et al., 2023).

Besides, researchers have adapted advanced derivative-free optimization algorithms to operate strictly on scalar feedback. For instance, RL-based frameworks like RLPrompt treat prompt generation as a discrete reinforcement learning problem, directly optimizing a reward signal derived from downstream task accuracy (Deng et al., 2022). Similarly, to improve the sample efficiency of exploring the discrete text space, methods like InstructZero map prompts into a continuous latent space, leveraging Bayesian Optimization to predict and maximize scalar task rewards efficiently (Chen et al., 2024a). Recent systems like BPO (Black-Box Prompt Optimization) further extend this score-driven paradigm to human preference alignment, using scalar preference scores to optimize user inputs without altering the underlying model weights (Cheng et al., 2024b).

6.1.2 *Qualitative-Feedback Refinement*

Building upon scalar-driven methods, a more nuanced approach focuses on iterative refinement using qualitative, interpretable feedback. Instead of a single score, the learning signal ($\mathcal{S}_t$) manifests as a textual critique, error analysis, or natural-language suggestion, denoted as c_t. This richer signal allows for a targeted revision process, formally modeled as

Table 2: Comparison of prompt optimization paradigms in prompt-based self-improvement. As learning signals become more structured and informative, optimization becomes less heuristic and more automated. ①: Scalar-feedback optimization; ②: Qualitative-feedback refinement; ③: Population-based evolution; ④: Textual-gradient optimization.

ID	Signal $\mathcal{S}_t$	Objective	Representative Systems	Advantages	Limitations
①	Scalar score	$\arg\max_{p\in\mathcal{P}} f(p)$	RLPrompt (Deng et al., 2022) BBT (Sun et al., 2022) APE (Zhou et al., 2022) OPRO (Yang et al., 2023) Dspy (Khattab et al., 2023)	+ Model-agnostic + Simple to deploy + No internal access	– Low interpretability – Sample-inefficient – Sensitive to search
②	Text critique	$\text{Refine}(p_t, c_t)$	Self-Refine (Madaan et al., 2023) Reflexion (Shinn et al., 2023) Critic (Gou et al., 2024) ACE (Zhang et al., 2025i)	+ Interpretable edits + Targeted correction + Reusable feedback	– Critique can be noisy – May drift – Validator-dependent
③	Selection signal	$\text{Evolve}(P_t, \text{Fit})$	Promptbreeder (Fernando et al., 2024) STOP (Zelikman et al., 2024) GPTSwarm (Zhuge et al., 2024a) AutoDAN (Liu et al., 2024b) Evol-Instruct (Xu et al., 2024a) GEPA (Agrawal et al., 2026)	+ Strong exploration + Maintains diversity + Escapes local optima	– Compute-heavy – Fitness is domain-tuned – Population drift
④	Textual gradient	$p_t \oplus g(p_t)$	APO (Pryzant et al., 2023) TextGrad (Yuksekgonul et al., 2024) metaTextGrad (Xu et al., 2025a) SkillOpt (Yang et al., 2026c)	+ Directional updates + Often sample-efficient + Highly automated	– Brittle gradients – Quality varies by LLM – Limited guarantees

an iterative loop where each new prompt p_{t+1} is a function of the previous prompt p_t and its corresponding qualitative feedback c_t:

$$p_{t+1} = \text{Refine}(p_t, c_t), \tag{18}$$

where c_t is typically generated by an evaluator or the LLM itself based on historical execution: $c_t = \text{Critique}(\text{Output}(p_t))$. While foundational paradigms like Self-Refine (Madaan et al., 2023) and multi-agent debate (Du et al., 2024) demonstrated the power of self-critique for transient output correction, recent scaffolding improvements persist this qualitative feedback to update the agent's prompting policy or instructional context. One prominent direction leverages error-driven qualitative analysis. Reflexion, for example, enables an agent to generate "verbal introspection" upon task failure, storing these qualitative reflections in memory to explicitly guide and constrain subsequent prompting attempts (Shinn et al., 2023). Expanding on this diagnostic capability, MAPS introduces an automated, LLM-tailored prompt optimization framework that explicitly induces and validates reusable natural-language rules from failure cases. By iteratively injecting these qualitative insights into the prompt, MAPS substantially optimizes the policy for complex tasks like unit test generation (Gao et al., 2025a). Similarly, inspired by Chain of Hindsight (CoH), agents can learn from textual contrasts by reviewing a history of past attempts accompanied by qualitative evaluations (Liu et al., 2024a). Another critical direction involves evolving the instructional context. To manage the growing complexity of text-based critiques, ACE introduces an agentic context engineering framework with a modular *Generator–Reflector–Curator* pipeline. ACE treats prompts and memory as evolving text playbooks, actively curating qualitative feedback while mitigating brevity bias and context collapse (Zhang et al., 2025i). In specific domain applications, systems like Scrable utilize continuous qualitative evaluation to iteratively refine the structural system prompt for customer review generation, halting only when the textual quality reaches a predefined standard (Azov et al., 2024).

6.1.3 Population-Based Evolution

To introduce more structured exploration and mitigate the risk of converging to local optima, researchers have adapted principles from evolutionary biology. In this paradigm, the learning signal $\mathcal{S}_t$ manifests as population-level fitness evaluations and selection pressures over a diverse pool of candidate prompts. Formally, these methods treat prompts as "genes" within a population $P_t = \{p_t^{(i)}\}_{i=1}^{N}$ at generation t. The evolutionary update leverages LLMs

to apply semantic operators rather than simple string manipulations, following three key steps:

1. **Selection:** A subset of prompts survives based on a fitness evaluation function, which translates task performance into selection pressure ($\mathcal{S}_t$).
2. **Crossover:** The LLM intelligently merges the semantic strengths of two parent prompts, generating offspring: $p_{\text{child}} = \text{Crossover}(p_t^{(i)}, p_t^{(j)})$.
3. **Mutation:** The LLM introduces semantic variations to explore new instructional phrasing: $p' = \text{Mutate}(p)$.

The subsequent generation, P_{t+1}, is formed from the fittest individuals and their offspring. Foundational frameworks like EvoPrompt (Guo et al., 2024) pioneered this by explicitly guiding LLMs to act as evolutionary operators, yielding crossover and mutation steps that are semantically meaningful and far surpass classical random character edits. Expanding the depth of this standard evolutionary search, frameworks like Promptbreeder (Fernando et al., 2024) introduces a profound self-referential mechanism where the LLM evolves not only the task prompts but also the "mutation prompts" (the instructions governing how new offspring are generated). Addressing scenarios where scalar fitness scores are unavailable, DEEVO creatively structures $\mathcal{S}_t$ through multi-agent debates, utilizing the win-rate of LLM-driven argumentation as the evolutionary fitness signal (Nair et al., 2025). Crucially, demonstrating the compositional nature of scaffolding improvement, recent work integrates population-based search with qualitative feedback. In reflective prompt evolution frameworks like GEPA (Agrawal et al., 2026), the evolutionary process is guided not merely by a scalar fitness score, but by a meta-level reflection step. After evaluating a generation, a reflector LLM analyzes successes and failures to generate textual critiques. This qualitative feedback explicitly informs the next generation's mutation and crossover operators, making the search highly targeted and sample-efficient. The success of such hybrid approaches highlights the distinct advantage of replacing opaque scalar rewards with directional, semantic guidance—directly paving the way for the formalized textual gradient methods discussed next.

6.1.4 Textual Gradient Optimization

The most recent paradigm is distinguished by a formal, mathematically-inspired treatment of the feedback signal, drawing direct analogies to gradient descent in continuous optimization. Rather than relying on heuristic edits or scalar search, the learning signal ($\mathcal{S}_t$) is formalized as a *textual gradient*—a structured, directional feedback message that explicitly diagnoses why an output is incorrect and prescribes a precise revision vector. This optimization process can be expressed with an analogous update rule:

$$p_{t+1} = p_t \oplus g(p_t), \tag{19}$$

where the textual gradient $g(p_t)$ serves directly as our learning signal ($\mathcal{S}_t = g(p_t)$). The $\oplus$ operator denotes a textual update step, where an LLM acts as the optimizer to semantically apply the gradient's guidance and revise the prompt. While the foundational concept of a "textual gradient" was introduced in Automatic Prompt Optimization (APO) (Pryzant et al., 2023), this direction has been significantly advanced by framing agentic workflows as computational graphs. TextGrad, for instance, operationalized this by introducing automatic differentiation via text, allowing qualitative gradients to backpropagate through complex, multi-component language systems (Yuksekgonul et al., 2024). Concurrently, semantic backpropagation frameworks have further formalized this process, executing first-order-like optimization on text nodes to achieve principled prompt refinement (Wang et al., 2024b). The sophistication of this paradigm is profoundly underscored by meta-level frameworks like MetaTextGrad. Rather than solely optimizing task prompts, it uses an LLM to dynamically optimize the "optimizer prompts" (the instructions dictating how the textual gradients are computed and applied), effectively allowing the agent to self-improve its own improvement process (Xu et al., 2025a). Looking forward, this formal feedback mechanism could bridge the gap between scaffolding and parameter-based learning: a textual gradient might be translated into low-rank parameter updates, seamlessly integrating prompt engineering with model fine-tuning.

Takeaway

Prompt-based self-improvement turns prompt optimization from an ad hoc practice into a signal-driven improvement loop. In existing approaches, the learning signal ($\mathcal{S}_t$) becomes progressively richer, evolving from scalar scores to qualitative critiques, and further to population-level selection and structured directional guidance. As feedback grows in informational content, prompt updates become less heuristic and more targeted, enabling increasingly automated and sample-efficient refinement without changing the foundation-model parameters.

6.2 Memory

The memory system serves as the core cognitive scaffolding for long-horizon agentic behavior. Traditional memory architectures often rely on raw content logging alongside fixed schemas, resulting in rigid, static organizations. Such append-only designs quickly succumb to context-window constraints and retrieval degradation, rendering them ill-suited for dynamically changing environments (Modarressi et al., 2024; He et al., 2024a; Packer et al., 2024). In contrast, self-improving agents treat memory not merely as a passive storage mechanism, but as an actively evolving scaffold. By continuously assessing the value, relevance, and strength of stored information, these agents autonomously reconstruct and expand their memory representations based on the flow of experience (Du et al., 2025; Xu et al., 2025b). This shift from static storage to dynamic self-organization enhances the agent's generality, establishing a foundation for open-ended autonomy (Xu et al., 2025f; Sang et al., 2025; Li et al., 2025e).

To systematically analyze this paradigm, we decompose memory-based improvement into three core dimensions: ***Memory Objects*** (the units of stored information), ***Memory Structure*** (the topological organization and indexing schema), and ***Memory Processing*** (the mechanisms for creation, retrieval, updating, and forgetting). Formally, we conceptualize the memory scaffold as a dynamic state $m_t := (\text{object}_t, \text{structure}_t)$, specifying the currently stored objects and their overarching organization. Driven by an execution-derived learning signal $\mathcal{S}_t$ (e.g., retrieval failures, task feedback, or capacity limits), memory evolution is formalized as:

$$m_{t+1} = \text{IMPROVE}m(m_t; \mathcal{S}t). \tag{20}$$

This update is instantiated through the memory processing module—a signal-driven family of operations (Write, Read, Update, Delete) parameterized by $\mathcal{S}_t$ that governs what to consolidate into object_{t+1} and how to reorganize structure_{t+1}. Finally, it is crucial to note the scope of our discussion. While some literature considers knowledge internalized within the foundation model's weights as "parametric memory" (Wu et al., 2025d; He et al., 2025b; Zhang et al., 2025n), this section focuses on non-parametric, externalized memory embedded within the agent's scaffold, maintaining the core assumption of a frozen foundation model.

6.2.1 Memory Object

Recalling our formal decomposition of memory state, this subsection focuses on object_t, i.e., *what* is stored within the agent's memory scaffold. The design of the memory object is paramount, as it directly governs storage efficiency, representation fidelity, and the capacity for cross-context knowledge transfer (Koley, 2025). Moving beyond raw, exhaustive interaction trails (e.g., infinite chat histories), self-improving agents increasingly favor storing processed, high-density abstractions (Rasmussen et al., 2025; Deng et al., 2024). By utilizing execution feedback ($\mathcal{S}_t$) to filter or compress experiences, agents effectively mitigate context-window limitations and storage costs (Cao et al., 2025). These memory objects can be fundamentally categorized into explicit and implicit representations.

- **Explicit Objects.** Explicit objects are human-readable and directly manipulable, which makes them the default choice when interpretability, attribution, and safety auditing are important. Their main advantage is controllability: developers can inspect, correct, and selectively expose them to the model. Their main limitation is scalability, since verbose

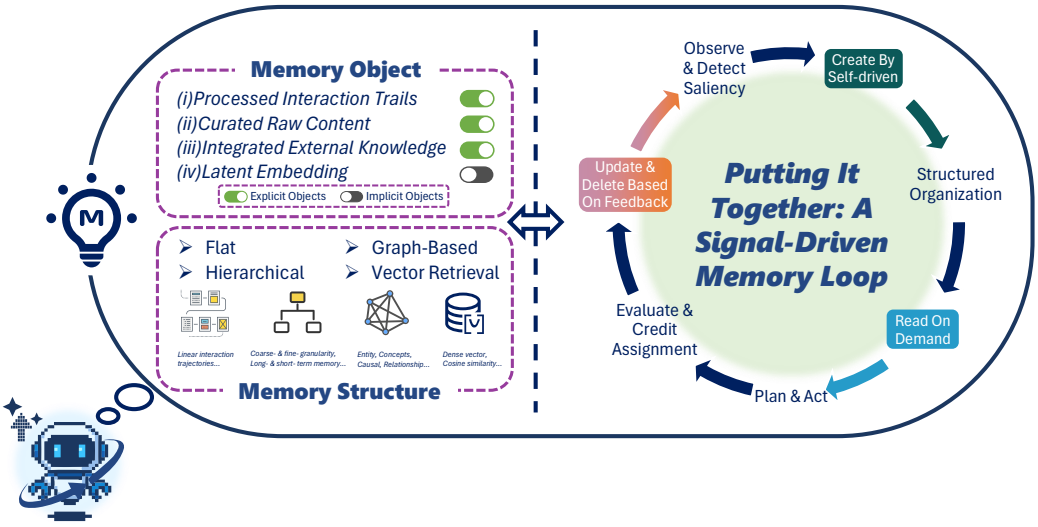


Figure 8: Overview of memory for self-improving agent.

Table 3: Memory-object scorecard (qualitative). Blue squares indicate an ordinal, literature-grounded assessment (1=low, 5=high) of typical tendencies for each memory object type, synthesized from representative system designs and reported failure analyses rather than from a single standardized benchmark.

Object type	Best-for persistence	Fidelity	Interpre-tability	Compact	Write cost	Audit-tability	Most common failure modes
Processed trails	• lessons • routines • summaries	■■■□ □	■■■■ ■	■■■■ □	■■■□ □	■■■■ ■	▸ summary bias ▸ stale heuristics ▸ weak credit assignment
Curated raw content	• evidence • exact artifacts	■■■■ ■	■■■■ ■	■□□□ □	■■■■ □	■■■■ ■	▸ context bloat ▸ retrieval noise ▸ privacy leakage surface
Integrated external knowledge	• shared factual state • grounding	■■■■ □	■■■■ □	■■■□ □	■■■■ □	■■■■ □	▸ grounding failure ▸ staleness / inconsistency ▸ tool brittleness
Latent embeddings	• associative carryover • fast recall	■■■□ □	■□□□ □	■■■■ ■	■■□□ □	■□□□ □	▸ drift / contamination ▸ hard-to-debug retrieval ▸ silent corruption

or weakly curated memories tend to increase retrieval noise and context pressure as the memory grows.

A first and widely used manifestation stores *processed interaction trails* that compresses raw trajectories into reusable, semantically meaningful units (e.g., distilled routines, heuristics, or reflections). Driven by task success/failure signals, agents abstract generalizable strategies from experience or maintain note-like intermediate traces to support long-horizon reasoning (Wang et al., 2025j; Ouyang et al., 2025b; Lanchantin et al., 2023; Lee et al., 2024a; Long et al., 2026).

A second manifestation retains *curated raw content*. Certain scenarios require preserving exact surface details that are hard to summarize without loss (e.g., code snippets, formulas, or screenshots). Rather than storing everything, self-improving agents selectively write back only high-value artifacts validated during trial-and-error, distilling dynamic "cheatsheets" for future reuse (Zhao et al., 2024a; Suzgun et al., 2025).

A third manifestation integrates *external knowledge*. Agents can integrate and maintain facts from external repositories. Grounding memory in shared references rather than free-form recollection enhances verifiability. Crucially, unlike static retrieval systems, self-improving agents utilize utility-based feedback to dynamically update, annotate, or prune these retrieved domain references (e.g., codebases or task-specific documents),

thereby mitigating error propagation in downstream reasoning (Tran et al., 2025; Zhang et al., 2024b; Peng et al., 2023).

- **Implicit Objects.** Implicit objects store memory in machine-native latent representations, including latent tokens, hidden states, and key–value cache augmentations. Their primary advantage is compactness and fast associative access, which makes them appealing under strict context limits or latency budgets. Their primary drawback is limited interpretability, which complicates debugging, targeted correction, and safety auditing, and can lead to representation drift when the memory is repeatedly rewritten or composed.

  Recent advancements explore several mechanisms for implicit memory scaffolding without altering the base FM parameters. *Generative latent memory* constructs latent sequences to enrich reasoning beyond text-based retrieval (Zhang et al., 2025e). *Latent state reconstruction* captures and reintegrates hidden representations to improve context retention (Dillon et al., 2025). A related line augments the decoding process via offline coprocessors that inject latent embeddings directly into the KV cache to boost generation fidelity (Liu et al., 2025c; Sun et al., 2025a). Finally, maintaining self-updatable latent memory pools offers a practical compromise between persistent state tracking and strict capacity control (Wang et al., 2024d; 2025h).

Trade-offs. Explicit memory objects offer high interpretability and controllability, simplifying debugging and safety auditing, but require rigorous curation to avoid overwhelming the context budget. Among them, processed trails favor generalization, curated artifacts favor precision, and integrated external knowledge favors verifiability. Conversely, implicit memory objects provide compact, high-speed associative access and mitigate context length issues, but are difficult to inspect, correct, and are susceptible to representation drift over long horizons. We summarize these qualitative trade-offs and common failure modes in Table 3.

6.2.2 Memory Structure

Building upon the definition of memory objects (object_t), the memory structure (structure_t) specifies the organizational schema and relational topology through which these objects are indexed, linked, and retrieved. For self-improving agents operating in complex, dynamic environments, an adaptive memory structure acts as more than a passive database; it actively empowers the agent to consolidate long-horizon experiences, resolve context fragmentation, and perform robust relational reasoning (Zeng et al., 2024). Consequently, how this structure is designed and updated directly dictates the agent's ability to scale its cognitive baseline efficiently. Prevailing structural paradigms can be broadly categorized into the following forms, each offering distinct trade-offs in scalability, expressiveness, and retrieval latency:

- **Flat Structure.** Flat memories store entries in strict temporal order. This append-only design makes writing computationally cheap and preserves the causal narrative of interaction. This topology is particularly advantageous for self-improving loops that rely on trajectory replay, as it maintains the unadulterated context necessary to diagnose failures ($\mathcal{S}_t$) and reproduce successful behaviors. The primary cost is retrieval scalability. As the stream expands, recall becomes overly dependent on truncation heuristics or coarse filtering. Consequently, agents often suffer from recency bias—surfacing recent but irrelevant interactions while missing older, decisive evidence. Furthermore, lacking inherent support for abstraction, flat structures tend to accumulate redundant low-level traces, exacerbating context pressure over long horizons (Antony et al., 2024; Sun & Zeng, 2025). Representative systems operationalize this design by augmenting chronological logs with lightweight indexing. For instance, SCM maintains a temporal memory stream enriched with summaries and embeddings, bridging basic semantic search with chronological access (Wang et al., 2025a). Similarly, frameworks like Self-Notes write transient insights directly inline during long-context reasoning. This keeps the memory aligned with the evolving cognitive state, allowing for immediate, feedback-driven course corrections while preserving the chronological flow of thought (Lanchantin et al., 2023).
- **Hierarchical Structure.** Hierarchical memories organize objects across multiple abstraction levels, addressing a central bottleneck in self-improving agents: the need to compress

Table 4: Memory architecture and processing operators. Checkmarks indicate the memory *object* and *structure* choices reported by each system. Dots denote the relative emphasis of a mechanism as primary (●), secondary (●), or absent/unclear (●). Processing is summarized by CRUD (Create, Read, Update, Delete). We further characterize governance along two dimensions: **Select**, which determines what information is written and retrieved based on saliency and utility, and **Maintain**, which sustains memory quality over long horizons through consolidation, refresh, and forgetting.

	Object		Structure				Processing				Governance	
	Explicit Objects	Implicit Objects	Flat	Hier.	Graph	Vector Retr.	Create C	Read R	Update U	Delete D	Select	Maint.
Self-Notes (2023)	✓	–	✓	–	–	–	●	●	●	●	●	●
Generative Agents (2023)	✓	–	✓	–	–	✓	●	●	●	●	●	●
Richelieu (2024)	✓	–	✓	–	–	–	●	●	●	●	●	●
DC (2025)	✓	–	✓	–	–	–	●	●	●	●	●	●
SAGE (2025)	✓	–	–	✓	–	–	●	●	●	●	●	●
Mem0 (2025)	✓	–	–	–	✓	✓	●	●	●	●	●	●
MemInsight (2025)	✓	–	–	✓	–	✓	●	●	●	●	●	●
MemGen (2025)	–	✓	✓	–	–	–	●	●	●	●	●	●
ACE (2025)	✓	–	–	–	–	✓	●	●	●	●	●	●
A-MEM (2025)	✓	–	–	–	✓	✓	●	●	●	●	●	●
AWM (2024)	✓	–	–	✓	–	–	●	●	●	●	●	●
Reasoning Bank (2025)	✓	–	–	–	–	✓	●	●	●	●	●	●
ReadAgent (2024)	✓	–	✓	–	–	–	●	●	●	●	●	●
M3-Agent (2025)	✓	–	–	✓	–	–	●	●	●	●	●	●
ExpeL (2024)	✓	–	–	–	–	✓	●	●	●	●	●	●
PRIME (2025)	✓	–	–	–	–	✓	●	●	●	●	●	●
CodeAgent (2024)	✓	–	–	–	–	✓	●	●	●	●	●	●
CMR (2025)	–	✓	–	✓	–	–	●	●	●	●	●	●
MemoryLLM (2024)	–	✓	✓	–	–	–	●	●	●	●	●	●
M+ (2025)	–	✓	✓	–	–	–	●	●	●	●	●	●
H-MEM (2025)	✓	–	–	✓	–	–	●	●	●	●	●	●
SALM (2025)	✓	–	–	✓	–	–	●	●	●	●	●	●
XMem (2022)	–	✓	–	✓	–	–	●	●	●	●	●	●
MovieChat (2024)	–	✓	–	✓	–	✓	●	●	●	●	●	●
SHIMI (2025)	✓	–	–	✓	✓	–	●	●	●	●	●	●

long histories into stable, reusable knowledge while preserving interaction-level details. By allocating high-level summaries, mid-level plans, and low-level traces into distinct layers, this topology reduces retrieval noise and supports long-horizon coherence. However,

the primary risk is structural brittleness. If the induced taxonomy misaligns with the task, strict hierarchies can fragment evidence and hinder cross-cutting retrieval, which severely impedes the agent's ability to recombine diverse experiences for continual improvement. Recent systems instantiate this hierarchy through different organizational lenses. For *task-oriented* abstraction, MobileGPT employs a goal-to-subtask-to-action hierarchy for GUI tasks, supporting both plan reuse and fine-grained execution recall (Lee et al., 2024b). For *semantic* abstraction, H-MEM (Sun & Zeng, 2025) and SHIMI (Helmi, 2025) construct multi-level semantic nodes, enabling coarse-to-fine retrieval and top-down traversal from abstract intents to specific entities. Alternatively, SALM approaches hierarchy via *tiered storage patterns* (short-term, long-term) to separate active context maintenance from durable reuse (Koley, 2025). Besides, similar principles extend to multimodal domains, where agents separate transient perceptual buffers from persistent representations for long-video understanding, drawing inspiration from classical human memory models (Cheng & Schwing, 2022; Song et al., 2024a; Atkinson & Shiffrin, 1968).

- **Graph-Based Structure.** Graph memories represent objects as nodes linked by semantic, temporal, or causal edges, aligning naturally with the need for self-improving agents to generalize across continuous interactions. By explicitly encoding relations, this topology replaces recency-based recall with retrieval by association. It enables multi-hop evidence aggregation that flat streams or strict trees struggle to support, which is vital when agents must attribute failures to latent dependencies, track evolving entity states, or reuse abstracted insights. However, the primary trade-off is maintenance complexity. Continuous graph construction, edge updating, and conflict resolution introduce significant computational overhead and risk structural drift as the agent repeatedly edits its own memory.

  Recent literature explores this paradigm through different relational lenses. For semantic and conversational tracking, systems like Mem0 (Chhikara et al., 2025) and SGMem (Wu et al., 2025e) parse dialogues into fact- or sentence-level graphs to support highly structured recall. To support temporal and causal reasoning, Zep provides a temporal-aware knowledge graph for continuous belief revision (Rasmussen et al., 2025), while CausalRAG leverages explicit causal edges to prevent spurious correlations from contaminating the self-improvement loop (Wang et al., 2025e). Some approaches even blend topologies; for instance, G-Memory introduces a hybrid hierarchical graph to separate reusable insights from fine-grained logs in multi-agent settings (Zhang et al., 2025d). Beyond text, explicit graph structures are similarly crucial for multimodal agents—such as GraphVideoAgent and Scene-MMKG—where maintaining spatial-temporal state transitions and entity interactions is essential for grounded reasoning (Chu et al., 2025; Song et al., 2024b).

- **Vector Retrieval Structure.** Vector-based memory indexes objects by dense embeddings and retrieves by similarity, which makes it a strong default when an agent must recall semantically related interactions under natural language queries. For self-improving agents, it serves as the primary engine for episodic recall and large-scale knowledge assimilation, supporting continual expansion without a hand-crafted schema. Its dominant failure mode is misalignment between similarity and usefulness. The nearest neighbors under an embedding metric may be topically similar yet decision-irrelevant, and this retrieval noise can systematically bias the agent's subsequent learning updates. As a result, self-improving systems often augment vector retrieval with re-ranking, hybrid signals, or adaptive controllers that dynamically gate the retrieval process.

  Building upon the classic RAG pipeline (Lewis et al., 2020), Agentic RAG integrates autonomous control over retrieval to manage dynamic episodic memories (Ravuru et al., 2024; Singh et al., 2025a). To mitigate the similarity-usefulness misalignment, systems explore various optimizations: *hybrid heuristics*, such as Generative Agents combining dense relevance with recency and importance (Park et al., 2023); and *adaptive representations*, where RMM introduces differentiable ranking to improve recall under long-term dialogues (Tan et al., 2025b). For domain-specific trajectory reuse, systems like MemoryBank (Zhong et al., 2024) and CTIM-Rover (Lindenbauer et al., 2025) index episodic segments to robustly fetch past successes, whereas MIRIX routes queries across multiple specialized sub-stores to overcome the limits of a flat index (Wang & Chen, 2025). Finally, this vector-based topology readily accommodates multimodal experiences, enabling

embodied agents like MrSteve to retrieve relevant past video frames for exploration (Park et al., 2025).

6.2.3 *Memory Processing*

Given the memory state $m_t := (\text{object}_t, \text{structure}_t)$, memory processing refers to the family of signal-driven operations that act on m_t throughout an agent's lifetime. Formally, we view memory improvement as $m_{t+1} = \text{IMPROVE}_m(m_t; \mathcal{S}_t)$. Unlike traditional agents that rely on static, hard-coded indexing, self-improving agents use the learning signal $\mathcal{S}_t$ (e.g., task outcomes, utility, or internal critiques) to dynamically adjust what to write, how to retrieve, and when to forget (Xu et al., 2025c; Liang et al., 2026b). Concretely, we instantiate IMPROVE_m as an adaptive CRUD (Create, Read, Update, Delete) operation family:

C Rather than appending raw logs exhaustively, memory creation is a selective distillation process guided by $\mathcal{S}_t$. Recent systems typically implement this through three recurring patterns: (1) *Semantic compression*, which transforms raw interactions into structured metadata, summaries, or reusable schemas for efficient indexing (Salama et al., 2025; Wang et al., 2025j; Ouyang et al., 2025b); (2) *Context-aware discrete decisions* (e.g., add, update, delete, or no-op), conditioned on retrieved neighbors to prevent redundant or conflicting entries (Chhikara et al., 2025; Xu et al., 2025f); and (3) *Controlled boundary insertion*, which dynamically optimizes the write policy for downstream utility closer to generation time (Zhang et al., 2025e). Across all designs, creation is bounded by a trade-off: over-writing inflates retrieval noise, whereas under-writing sacrifices long-term capability.

R Memory reading dictates the quality of an agent's downstream reasoning. To maintain precision as the memory bank scales, self-improving systems typically employ four key mechanisms: (1) *Hybrid heuristics*, which blend semantic relevance with recency and importance scores to stabilize recall (Park et al., 2023); (2) *Structure-aware retrieval*, which performs coarse-to-fine traversal over hierarchical or graph topologies to isolate reusable abstractions from low-level traces (Zhang et al., 2025d); (3) *Retrieval gating*, which dynamically decides whether to query memory and how much context is sufficient, thereby optimizing token cost and minimizing distraction (Wang et al., 2025a); and (4) *Retrieval-driven adaptation*, where historical trajectories are fetched as actionable cases to guide behavior without requiring parametric model updates (Zhou et al., 2025a). Ultimately, ineffective reading policies will hinder performance: retrieving irrelevant noise or missing critical details will directly cause subsequent planning and execution to fail.

U Memory updating is the primary mechanism for self-correction and knowledge evolution. Guided by feedback ($\mathcal{S}_t$), agents typically implement updates through four operational patterns: (1) *Scheduled review and attenuation*, which periodically strengthens high-utility items and decays obsolete ones to stabilize memory growth (Liang et al., 2026b); (2) *Local refresh*, which dynamically updates topological neighbors during new insertions to maintain contextual consistency (Xu et al., 2025f); (3) *Iterative distillation*, which synthesizes repeated successes into compact, reusable abstractions via continuous selection and replacement (Suzgun et al., 2025; Zhang et al., 2025i); and (4) *Offline aggregation*, which shifts computationally expensive compression away from the online execution loop while preserving recall quality (Fang et al., 2026). Without effective update policies, agents suffer from memory decay: outdated facts remain, knowledge structures break down, and improper merging erases important details.

D Memory deletion acts as a systematic noise reduction and resource optimization mechanism. Guided by signal ($\mathcal{S}_t$), agents dynamically identify and remove obsolete or redundant entries through three main patterns: (1) *Multi-stage pruning*, which filters low-value items at write time and periodically clears entries based on access frequency and relevance (Zhang et al., 2025g); (2) *Consensus-based eviction*, which uses collaborative voting in distributed setups to prevent the accidental deletion of critical shared knowledge (Bach, 2025); and (3) *Tiered eviction*, which applies operating-system-inspired rules across different memory layers to bound size while preserving long-term consistency (Kang et al., 2025).

Improper deletion policies create a direct dilemma: over-pruning loses critical knowledge, while under-pruning floods the system with outdated noise and slows down retrieval.

Putting it together: a signal-driven memory loop. The dimensions of memory objects, structure, and processing culminate in a unified, signal-driven lifecycle (Figure 8). This loop proceeds in a continuous cycle: (i) Observe & Detect saliency from new interactions; (ii) Create compact objects; (iii) Organize them into the chosen topological structure; (iv) Read on demand via adaptive retrieval; (v) Plan & Act based on fetched context; (vi) Evaluate outcomes to derive the learning signal $\mathcal{S}_t$; and (vii) Update/Delete entries to consolidate high-value knowledge and prune noise. Together, these stages elevate memory from a passive cache to a self-governing engine that sustains open-ended autonomy.

Takeaway

The Self-Improvement Memory Loop transforms the conceptual questions of memory management into an actionable, signal-driven framework. By unifying memory writing, structural organization, adaptive retrieval, and feedback-based pruning, this continuous lifecycle effectively elevates memory from a static cache to a scalable engine for self-improvement.

6.3 Tool

Basic agents have been constrained by their reliance on static, manually curated toolkits, lacking the autonomy to adapt, discover, or integrate new resources when confronted with novel challenges (Shapiro et al., 2023; Zhang et al., 2025a; Sang et al., 2025; Huang et al., 2025b). The transition from a basic executor to a genuinely self-improving agent necessitates a shift away from simple, pre-defined tool use toward *Tool Governance Metacognition*. This dynamic paradigm empowers the agent to autonomously reason about the necessity, utility, and reliability of its tools, thereby continually advancing its capability boundaries rather than merely operating within them (Liu & van der Schaar, 2025; Yin et al., 2025b).

Formally, tool-based improvement can be written as

$$\mathcal{T}_{t+1} = \texttt{IMPROVE}_{\mathcal{T}}(\mathcal{T}_t; \mathcal{S}_t), \tag{21}$$

where $\mathcal{S}_t$ is the learning signal produced by tool governance metacognition. This metacognition represents the architectural shift essential for a self-improving agent, establishing a dynamic and self-directed mechanism for managing its operational tools (Wang et al., 2025b). It is instantiated through three core dimensions: Dynamic Tool Routing, which orchestrates the efficient selection and combination of tools from a growing pool; Iterative Tool Refinement, which adapts and debugs existing instruments to overcome execution failures or environmental shifts; and Autonomous Tool Creation, which invents novel tools to fundamentally expand the agent's capability boundaries. This governance structure moves tool use from a static look-up process to a generative, self-improving cycle.

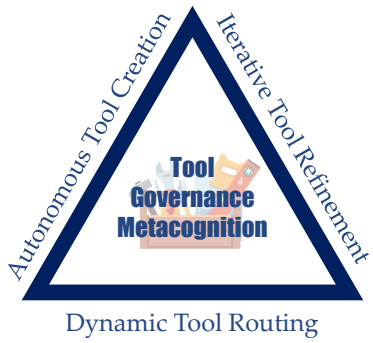


Figure 9: Tool Governance Metacognition.

6.3.1 *Dynamic Tool Routing*

Dynamic tool routing concerns how an agent selects, sequences, and coordinates tools in a heterogeneous environment. As tool pools grow, routing becomes a primary bottleneck for self-improvement. A large tool set increases coverage but also increases error modes, including misrouting, compounding execution failures, and wasted compute. Routing methods therefore trade off coverage, reliability, and decision cost. Most existing systems can be understood through three routing paradigms, distinguished by what they treat as the "retrieval unit" and how they incorporate feedback over time.

Retrieval- and graph-based routing. Systems in this paradigm tackle the routing bottleneck by optimizing either the retrieval space or the representation of dependencies. To optimize the retrieval space and maintain scalability, systems dynamically manage the granularity and scope of the tool pool. For instance, while MemTool prevents routing degradation by actively pruning the tool set into a lightweight operational memory (Lumer et al., 2025a), TAR expands the retrieval unit itself, dynamically routing between atomic APIs and entire competent agents to handle higher-level tasks (Lumer et al., 2025b). Crucially for self-improvement, this retrieval process is not static; it evolves by leveraging past successes as a learning signal. Systems like VOYAGER and MetaAgent index reusable tool-use trajectories into procedural memory, allowing the routing policy to continuously refine itself based on structural knowledge from prior executions (Wang et al., 2023; Qian & Liu, 2025). Conversely, to address the compositionality limitation of flat retrieval, recent works encode tool transitions as topological structures. Rather than one-shot relevance, systems like ToolNet and OrchDAG model tool dependencies as directed graphs, enabling routing that accounts for multi-step feasibility and preconditions (Liu et al., 2024c; Lu et al., 2025). Building on this, MassTool combines learned semantic matching with these graph structures, achieving high-precision navigation even within massive and complex tool topologies (Lin et al., 2025).

Policy-learning routing. Policy-learning routing internalizes tool choice as sequential decision making. Instead of retrieving a tool from an external index, the agent learns invocation behavior from training signals, which can improve robustness in multi-turn settings where success depends on state, history, and error recovery. The main trade-off is data and feedback. Learned routing can overfit to training environments and can be brittle when tool interfaces or distributions drift. To establish a reliable routing baseline, systems like AUTOACT, MCP-Flow, Tool-Star, and DeepEyesV2 bootstrap policies through supervised fine-tuning on synthetic or mined trajectories (Qiao et al., 2024b; Wang et al., 2025g; Dong et al., 2025; Hong et al., 2026). However, to continually improve long-horizon behavior, routing policies must transcend static supervision. Works like AGENTFLOW and SPORT achieve this by applying sparse rewards or preference signals to shape planning and exploration, while AutoTIR and DeepAgent explicitly align tool selection with multi-objective compliance and action-level attribution (Li et al., 2026b; 2025a; Wei et al., 2025c; Li et al., 2025c). Pushing this paradigm to its limit, ToolGen collapses retrieval, selection, and invocation into a single generative process via tool-token unification. While this elegantly reduces pipeline complexity, it places a heavy burden on the learned policy to remain calibrated under distribution shifts (Wang et al., 2025f).

Proactive and interactive routing. A third paradigm treats routing as an interactive process rather than a one-shot decision. This perspective is motivated by two recurring failure sources in self-improving agents. First, user intent is often underspecified, and misrouting can be avoided by clarification. Second, execution failures are common, and routing must support local repair without restarting the entire plan. Proactive routing therefore trades additional interaction and compute for higher reliability. Rather than failing passively, these systems embed metacognitive interventions to handle uncertainty. When confronting underspecified intents or functional gaps, agents like MCP-Zero and ASKTOACT proactively initiate tool discovery or prompt for clarification, transforming ambiguity into a learning signal for self-correction (Fei et al., 2025; Zhang et al., 2025l). Conversely, when facing execution failures, dynamic repair becomes essential. Tool-Planner facilitates this by clustering tools into interchangeable kits, enabling localized, API-level repair without the prohibitive cost of global replanning (Liu et al., 2025e). Similarly, ToolACE-R dynamically calibrates its revision effort based on task difficulty, balancing routing reliability against practical compute budgets in real-world deployments (Zeng et al., 2025).

6.3.2 Iterative Tool Refinement

Iterative tool refinement is the mechanism by which agents turn fragile programs into dependable skills. It is central to self-improvement because tool errors are not merely execution-time failures. If unreliable tools are stored and reused, they can corrupt the agent's future behavior through repeated retrieval and compounding errors. Refinement

therefore serves both as debugging and as a gatekeeping process that controls what enters the structural skill repository (Ravi et al., 2025; Dolcetti et al., 2025; Petrovic et al., 2025).

A canonical refinement loop alternates between generation, execution, and revision. VOYAGER establishes this baseline by executing generated code, capturing error traces and environment feedback as learning signals ($\mathcal{S}_t$), and feeding them back into subsequent revisions until a verifier confirms success (Wang et al., 2023). Building on this foundation, recent systems strengthen the refinement process across three dimensions: (1) Critique Specialization. To enhance diagnostic accuracy, systems move beyond generic self-reflection. For instance, STELLA deploys a dedicated critic agent to assess intermediate results and provide targeted feedback, mitigating the tendency of monolithic models to overlook domain-specific failure modes (Jin et al., 2025). (2) API Abstraction. Rather than merely revising raw action traces, several works focus on abstracting robust subroutines. SkillWeaver distills interactions into reusable web tools refined through execution feedback (Zheng et al., 2025a), while PyVision dynamically refines Python programs to stabilize multimodal tool use against noisy perception (Zhao et al., 2025b). This abstraction significantly improves skill transfer across tasks. (3) Interface Alignment. Addressing the semantic gap between agents and tools, systems like DRAFT iteratively refine tool documentation rather than the underlying code. This targets the critical observation that many execution failures stem from mismatches between natural language instructions and tool affordances, rather than from flawed implementations (Qu et al., 2025).

6.3.3 Autonomous Tool Creation

Autonomous tool creation expands the agent's capability boundary by synthesizing new executable functions when existing tools are insufficient. For self-improving agents, creation is most valuable when it converts one-off problem solving into reusable procedural knowledge. It also introduces its own risks. Newly created tools must be validated, documented, and integrated without destabilizing routing policies, otherwise tool growth can increase brittleness rather than autonomy (Gao et al., 2026a; Qiu et al., 2025c).

Existing systems approach this challenge by advancing tool creation across three dimensions: (1) Synthesis Triggers. Tool creation can be driven either by immediate needs or open-ended curiosity. While systems like ATLASS and PyVision emphasize on-demand invention—synthesizing and refining tools when current retrieved APIs fail (Haque et al., 2025; Zhao et al., 2025b), agents like FRIDAY and STELLA shift toward self-directed exploration. They autonomously propose curricula or discover domain-specific resources (e.g., bioinformatics) to proactively accumulate tools outside a fixed task list (Wu et al., 2024b; Jin et al., 2025). (2) Lifecycle Automation. Synthesizing raw code is only the first step; rendering it executable is the actual bottleneck. TOOLMAKER addresses this by automating the end-to-end tool lifecycle, extracting logic from scientific papers, installing dependencies, debugging in a closed loop, and producing robust callable interfaces (Wölflein et al., 2025). This transforms abstract function generation into tangible deployability (Cai et al., 2024; Yuan et al., 2024a; Qian et al., 2023). (3) Standardized Integration. To ensure that explosive tool growth does not destabilize existing routing policies, recent systems adopt protocol-driven architectures. Frameworks like Alita and Code2MCP automate the conversion of code repositories into standardized services via the Model Context Protocol (MCP) (Qiu et al., 2025c;b; Ouyang et al., 2025a). Similarly, AgentOrchestra enforces a rigorous pipeline of intent parsing, validation, and formal registration before a new tool enters the active pool (Zhang et al., 2025j). This ensures that autonomous creation remains fully compatible with the agent's broader governance and retrieval constraints.

Takeaway

Based on the Tool Governance Metacognition framework, self-improving agents evolve tool usage into a growth-oriented architecture with dynamic routing, iterative refinement, and autonomous creation capabilities. This architecture transforms the system from a static toolkit into a dynamic skill repository with sustainably expandable capabilities.

6.4 Full Scaffolding

Full scaffolding self-improvement represents the deepest level of architectural intervention. Rather than merely tuning isolated components (e.g., prompts, tools, or memory), the agent treats its entire operational logic and codebase as a mutable substrate, enabling fundamental structural reorganization. Formally, recalling the agent state $\mathcal{A}_t = (\theta_t, \Sigma_t)$, full scaffolding improvement corresponds to the most general *scaffolding-only* transition:

$$\Sigma_{t+1} = \texttt{IMPROVE}_{\Sigma}(\Sigma_t; \mathcal{S}_t), \tag{22}$$

where $\mathcal{S}_t$ denotes the learning signal extracted from the agent's own execution traces and evaluations (e.g., task success/failure, unit tests, self-critique, cost signals). Crucially, unlike settings with a fixed external meta-optimizer, the improvement procedure is itself implemented within the current scaffolding, making the update *self-referential*:

$$\Sigma_{t+1} = \mathcal{I}_{\Sigma_t}(\Sigma_t; \mathcal{S}_t), \tag{23}$$

where $\mathcal{I}_{\Sigma_t}$ highlights that the improver can evolve together with the agent it improves. In principle, these self-referential loops can discover unanticipated update strategies and enhance the agent's intrinsic evolvability (Zhang et al., 2026d; Gerhart & Kirschner, 2007; Hendrikse et al., 2007; Dawkins, 2019). While fully open-ended recursive self-improvement remains a grand challenge, current systems successfully operationalize this concept within defined objectives, benchmarks, and safety protocols.

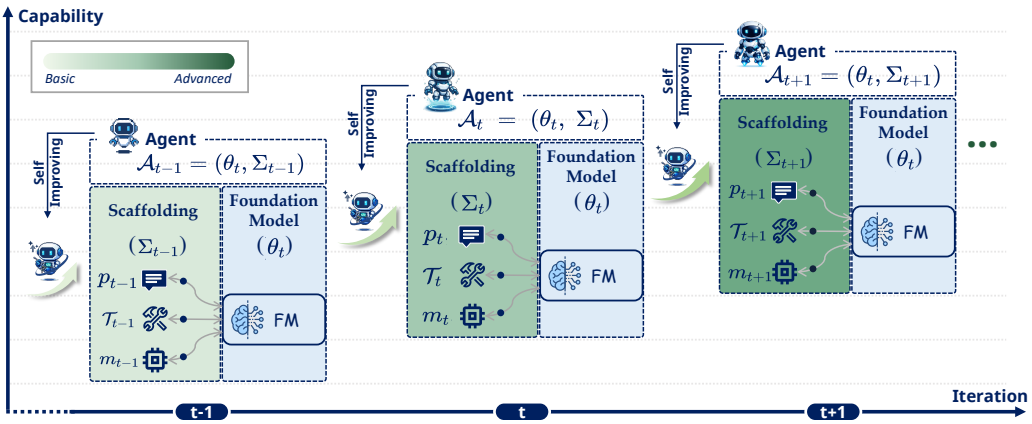


Figure 10: Full scaffolding self-improvement across iterations.

Within this paradigm, agents' scaffolding is typically represented as conventional computer programs with mild constraints. Formally, let $\langle \Sigma_t \rangle$ denote a serializable encoding of the program that implements the agent, and let $exec(\cdot)$ denote execution in an environment (e.g., an interpreter, compiler, and sandbox). A full-scaffolding update can be viewed as producing a candidate program via executing the current agent-as-improver on its own code:

$$\langle \tilde{\Sigma}_{t+1} \rangle = exec(\langle \Sigma_t \rangle; \mathcal{S}_t), \tag{24}$$

often materialized as a patch Δ_t applied to the current scaffolding, $\tilde{\Sigma}_{t+1} = \Sigma_t \oplus \Delta_t$. In practice, a verifier (e.g., unit tests, regression suites, safety checks) gates the update:

$$\Sigma_{t+1} = \begin{cases} \tilde{\Sigma}_{t+1}, & \mathcal{V}(\tilde{\Sigma}_{t+1}) = 1, \\ \Sigma_t, & \text{otherwise.} \end{cases} \tag{25}$$

The universal representability of computer programs (i.e., Turing completeness) enables self-improving agents to explore an extensive space of self-improving strategies, bounded in principle only by the theoretical limits of computation. Early work on applying a computer program to improve itself dates back to self-referential program-search and meta-evolution frameworks (Schmidhuber, 1987).

AlphaEvolve (Novikov et al., 2025) is designed as a coding agent for open scientific problems. It adopts an evolutionary paradigm, continuously receiving feedback from one or more evaluators to iteratively improve its algorithms, thereby substantially accelerating new scientific discoveries and optimizing complex technical stacks. Similarly, ShinkaEvolve (Lange et al., 2026) is an LLM-driven program-evolution framework that enables efficient, open-ended program discovery and optimization across multiple tasks with very few evaluation samples, via exploration–exploitation–balanced parent sampling, novelty-based rejection sampling for code, and bandit-based adaptive selection of an integrated ensemble of LLMs.

ADAS (Hu et al., 2025) searches within a design space for an agentic system that maximizes a given evaluation function using a dedicated search algorithm. EvoFlow continuously evolves a set of heterogeneous workflows that achieve favorable cost–performance trade-offs while maintaining structural diversity, yielding a Pareto set in an online manner. Self-Taught Optimizer (Zelikman et al., 2024) abstracts an agent's scaffolding as an improver: given its own program and a utility function, it repeatedly queries an LM to generate multiple candidate improved versions and selects the highest-scoring one as the output, forming a recursive self-improvement loop. Agent Symbolic Learning (Ou et al., 2025) views a language agent as a "symbolic network" whose weights are instantiated by prompts, tools, and their composition; it simulates backpropagation and gradient descent through natural-language losses/gradients, and uses a symbolic optimizer to jointly update prompts, tools, and pipelines, enabling sustained self-learning and evolution in real deployments.

Another representative line is the open-ended evolutionary framework inspired by the Gödel machine. Yin et al. (2025c) proposes a Gödel-machine-inspired self-referential agent framework, Gödel Agent, implemented via monkey patching. It autonomously performs self-awareness, self-modification, and recursive self-improvement, aiming to search the entire agent design space. Inspired by Darwinian evolution and open-ended research, Zhang et al. (2026d) proposed the self-evolving coding agent Darwin Gödel Machine (DGM), which maintains an archive of newly generated coding agents and grows a continually expanding tree of diverse, high-quality agents through open-ended exploration, allowing parallel exploration of multiple paths in the search space. Subsequently, Wang et al. (2026b) proposed the Huxley-Gödel Machine (HGM), motivated by Huxley's notion of clades; it introduces a *clade-level metaproductivity* (CMP) metric to guide evolution as an approximation to the Gödel-machine ideal. Live-SWE-Agent (Xia et al., 2025) is the first real-time software agent that can autonomously and continuously evolve on the fly at runtime; by evolving its own scaffolding, especially its tool components, it achieves desired performance on software issue-solving tasks.

Takeaway

Full scaffolding improvement is the category most closely related to recursive self-improvement, because it treats the agent's own source code and operational logic as mutable substrates. While achieving truly open-ended recursive self-improvement remains a grand challenge, current systems successfully operationalize this concept as bounded, verifiable loops. Operating within human-designed objectives and safety protocols, these systems provide a principled and measurable path toward reliable self-improvement.

7 Applications

The preceding sections organized self-improving agents by what is updated, namely model parameters or scaffolding, and by the signals that drive these updates. This section examines how these mechanisms appear across representative domains. Across applications, a common pattern is the use of sandboxed or otherwise controlled environments that provide feedback while limiting the cost of failure. **Software engineering** relies on compilers, tests, and continuous integration pipelines; **web automation** uses simulated or instrumented browsers; **games** provide resettable environments with explicit rules and outcomes; **scien-**

tific discovery uses executable workflows, domain tools, and simulators; **embodied AI** relies on robotic simulators and limited real-world rollouts; and **general computer control** uses virtualized desktops or isolated operating-system environments. The fidelity, scalability, and cost of these environments shape each domain's dominant bottlenecks, improvement targets, and iteration modes, as summarized in Table 5.

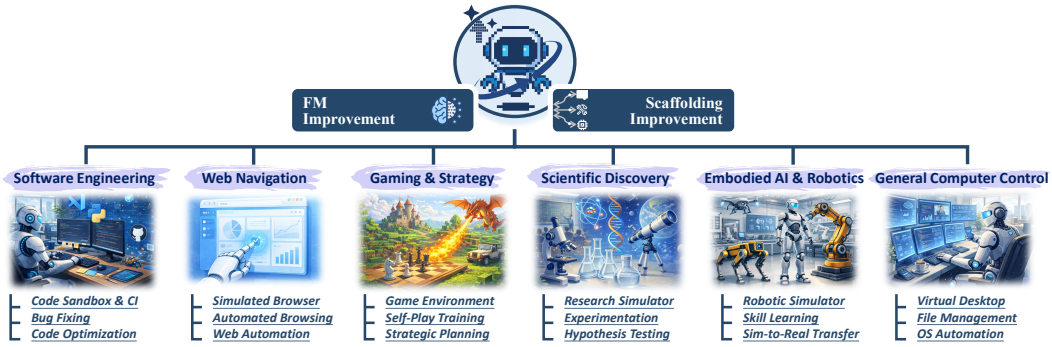


Figure 11: Representative application domains for self-improving agents.

7.1 Software Engineering

Software engineering (SWE) is a useful testbed for self-improving agents because it provides dense and automatable feedback. Compilers, unit tests, linters, and continuous integration pipelines can turn many agent actions into checkable outcomes, making improvement easier to measure than in domains that rely on subjective evaluation. This property naturally supports both branches of our taxonomy: (i) **FM improvement**, where executable feedback is used to update model parameters θ, and (ii) **scaffolding improvement**, where the model is fixed but the scaffold Σ is revised. In SWE, the scaffold may include prompts, memory, tools, control logic, and, in some cases, the agent's own source code.

Evaluation substrate vs. self-improvement. It is important to distinguish evaluation environments from self-improving methods. SWE-bench provides a standardized suite of real GitHub issues, repositories, and test-based validation, enabling quantitative measurement of progress (Jimenez et al., 2024). Several related benchmarks have since been proposed (Deng et al., 2025; Aleithan et al., 2024; Yang et al., 2025a). However, systems such as SWE-agent (Yang et al., 2024) and Agentless (Xia et al., 2024) are better viewed as non-self-improving baselines under our definition. They may iterate within a task instance, but they do not, by default, perform persistent cross-interaction updates to θ or Σ. Their relevance here is methodological: they show that scaffold design can substantially affect performance, motivating the question of whether such scaffolds can be improved automatically rather than hand-engineered.

Scaffolding improvement via self-modifying software agents. A distinctive feature of SWE is that the agent itself is software, which makes direct scaffold or source-code modification feasible. The Darwin G"odel Machine operationalizes this idea by iteratively rewriting its own codebase and validating candidate modifications on coding benchmarks, producing a tree archive of improved descendants (Zhang et al., 2026d). Building on this self-modification paradigm, the Huxley–G"odel Machine searches the space of self-modifications using an estimate of descendant performance, aiming to optimize long-term improvement potential rather than only immediate benchmark gains (Wang et al., 2026b).

Live-SWE-agent moves scaffold improvement from offline search to runtime self-evolution. Starting from a minimal scaffold, the agent edits and extends its own scaffold while solving real SWE tasks, so the current issue can inform future behavior (Xia et al., 2025). SE-Agent studies a complementary trajectory-level mechanism, using cross-trajectory revision, recombination, and refinement to escape local optima in reasoning and action sequences

(Guo et al., 2025b). Viewed through $\Sigma = (p, m, T, g)$, these methods update increasingly rich scaffold components, from prompts and tool routines to full system code. They also highlight a SWE-specific lesson: performance depends not only on the base model, but also on the agent's executable interface to the codebase and its ability to preserve reusable problem-solving strategies across issues.

FM improvement from executable feedback. SWE also provides a direct route to FM improvement because test outcomes can serve as rewards or supervision. SWE-RL scales reinforcement learning to real-world software engineering tasks by using software-evolution data and executable signals to improve the underlying model (Wei et al., 2025d). Related work studies reinforcement learning in long-context and multi-turn SWE settings, with the goal of training models that can sustain extended tool-using interaction while remaining grounded in verification feedback (Golubev et al., 2025). A practical limitation is that execution-based rewards can be sparse, noisy, or expensive, due to flaky tests, incomplete coverage, or costly environment setup. One complementary direction strengthens the executable signal itself by expanding the test suite: curiosity-driven planning can steer an LLM to generate tests that reach under-explored behaviors (Amayuelas et al., 2026). Execution-free feedback can therefore complement tests. SWE-RM explores reward-model-based scoring as a substitute for, or supplement to, unit tests, and analyzes which verifier properties transfer to reinforcement learning improvements (Shum et al., 2025).

Synthesis and open challenges. SWE exposes a useful contrast between the two improvement families. Scaffolding improvement is fast, modular, and model-agnostic, but it may overfit to benchmark tooling, repository conventions, or interaction protocols. FM improvement can internalize skills into θ and may transfer more broadly, but it is more expensive and more exposed to reward hacking and evaluation artifacts. Robust SWE self-improvement therefore requires (i) stronger evaluation against benchmark-specific search and verifier overfitting, such as benchmark mutation and stress testing, as well as (ii) hybrid loops in which scaffold-level discoveries are distilled into the base model only when they transfer beyond a fixed verifier or benchmark protocol (Garg et al., 2026).

7.2 Web Navigation and Automation

Web navigation and automation is a challenging domain for self-improving agents because the environment is dynamic, long-horizon, and often weakly verified. Web interfaces change frequently, the same user intent may be realized through different layouts or DOM structures, and failures often become visible only late in a trajectory. These properties make one-shot prompt engineering brittle and motivate improvement loops that persist across tasks. In our taxonomy, this domain supports both FM improvement, which updates θ from interaction data and feedback, and scaffolding improvement, which keeps θ fixed while updating Σ.

Evaluation and data substrate. A first line of work builds reproducible environments and demonstrations, which are prerequisites for systematic self-improvement. Mind2Web provides diverse instruction-following trajectories on real websites and supports supervised learning or imitation as an initial improvement step (Deng et al., 2023). WebArena offers a self-hostable environment with long-horizon tasks and execution-based success criteria (Zhou et al., 2024). VisualWebArena extends evaluation to visually grounded web tasks and exposes failures in perception and grounding (Koh et al., 2024). WorkArena targets enterprise-style workflows and highlights the gap between current agents and routine knowledge-worker automation (Drouin et al., 2024b). BrowserGym further unifies evaluation across web-agent benchmarks, reducing experimental fragmentation and enabling more controlled comparisons of training and scaffold design (Drouin et al., 2024a; de Chezelles et al., 2025). These resources are not self-improving methods themselves, but they provide the data and measurement substrate needed to study improvement across interactions.

Scaffolding improvement with memory and grounding. In web environments, many improvements come from better scaffolding because robust grounding remains a central

bottleneck. SeeAct couples visual understanding with action grounding on live websites, reducing brittle action selection in long-horizon execution (Zheng et al., 2024). WebCoach is more directly aligned with self-improvement: it equips browsing agents with cross-session memory curated from new trajectories, allowing the agent to avoid repeated mistakes without retraining the base model (Liu et al., 2025a). ReAP similarly stores and reuses reflections over successful and failed trajectories to improve later web navigation (Azam et al., 2025). These approaches instantiate scaffolding improvement by updating memory content, retrieval policies, and grounding-related structural artifacts that transfer across tasks.

FM improvement from interaction feedback. A complementary line updates θ using web interaction signals. Patel et al. (2024) study self-improvement on WebArena by fine-tuning on mixtures of model-generated data and show that iterative training can improve web-agent capability. OpenWebVoyager proposes an exploration-and-feedback loop on real websites, improving its policy by learning from high-quality trajectories over multiple iterations (He et al., 2024b). Several works adopt reinforcement learning for long-horizon browser control. WebRL introduces a self-evolving online curriculum that generates new tasks from failures and combines it with outcome-supervised reward modeling (Qi et al., 2025). WebAgent-R1 trains web agents via end-to-end multi-turn reinforcement learning with binary success rewards and reports gains on WebArena-Lite (Wei et al., 2025e). Beyond navigation benchmarks, Agent Q studies learning from both successful and unsuccessful trajectories in WebShop and shows that such experience can improve generalization in multi-step web interaction (Putta et al., 2024). These FM-oriented methods also reveal a recurring difficulty: web feedback is often sparse, so improvement depends heavily on reward modeling, curriculum design, and the stability of the evaluation environment.

Synthesis and open challenges. Web automation faces a core challenge of self-improvement. This challenge comes from non-stationarity and weak observability. Page layouts and interaction flows often change over time. These changes can quickly weaken gains that were measured on static snapshots. Scaffolding improvement can help systems update grounding, state abstraction, and recovery behavior more quickly. Yet it can also overfit to specific websites. It may also increase safety risks, such as prompt injection or unintended high-impact actions (Chen et al., 2026). FM improvement can support more transferable navigation skills. However, it depends on stable training signals and careful treatment of stale data. Progress therefore requires drift-aware evaluations. It also requires sandboxed protocols that separate safe exploration from irreversible actions. Finally, it requires grounding-centered improvements that transfer across layouts, DOM structures, and interaction conventions. These improvements should avoid memorizing site-specific patterns.

7.3 Games and Strategic Reasoning

Games remain a central testbed for self-improving agents because they provide repeatable interaction, well-defined objectives, and scalable feedback (Stanić et al., 2023). Even when state spaces are large and long-horizon planning is required, games offer closed training loops in which agents can generate experience through self-play (Silver et al., 2018; 2017a; Schrittwieser et al., 2020; OpenAI et al., 2019). This makes the domain a natural fit for our taxonomy. Many game agents improve by updating model or policy parameters θ through reinforcement learning, while others improve by evolving the scaffold Σ, such as curriculum logic, planning routines, or memory structures that store reusable skills and experience.

FM improvement through self-play and outcome feedback. Recent work extends self-play to foundation models by using strategic games as sources of interaction data and outcome-based learning signals. SPAG trains language models through a two-player adversarial game and improves the policy via self-play reinforcement learning (Cheng et al., 2024c). SPIRAL studies multi-turn zero-sum self-play and shows that self-generated competitive interactions can provide a scalable signal for improving reasoning without human-labeled data (Liu et al., 2026a). In adversarial game suites, SCO-PAL performs step-level policy

optimization from game interaction data and identifies self-play as an effective opponent-selection strategy for strategic reasoning (Zhang et al., 2025m). Self-play reinforcement learning with limited initial data further examines how self-generated interactions can bootstrap improvement when external supervision is scarce (Fang et al., 2025c).

Imperfect-information and multi-agent settings introduce additional challenges in credit assignment and payoff estimation (Zhuge et al., 2023; Stanić et al., 2023). MARSHAL proposes an end-to-end reinforcement learning framework for multi-agent self-play across cooperative and competitive strategic games, and reports transfer beyond games to multi-agent reasoning benchmarks (Yuan et al., 2026). In negotiation-heavy environments, DipLLM studies fine-tuning for equilibrium-oriented policies in Diplomacy, providing a parameter-updating route for language-mediated strategic interaction (Xu et al., 2025e). For high-variance payoff domains, SPRL uses reward shaping from self-play payoffs to stabilize training in complex strategic games (Qi et al., 2026). Social deduction games provide another setting in which language agents can improve long-horizon strategic behavior through reinforcement learning from interaction outcomes (Xu et al., 2024b).

Scaffolding improvement through curricula and reusable skills. Games also support scaffolding improvement because skills, procedures, and curricula can be represented explicitly and reused across interactions. Voyager exemplifies this paradigm in Minecraft by maintaining an expanding library of executable skills and an automatic curriculum that persists across tasks (Wang et al., 2023). Odyssey extends this direction with a structured open-world skill library and a planner–critic loop that improves long-horizon execution through skill reuse and prerequisite checking (Liu et al., 2025d).

Beyond open-world games, several methods construct reusable skills from interaction trajectories. Skill Set Optimization extracts high-reward subtrajectories, converts them into transferable skills, and prunes ineffective ones, yielding continual in-context policy improvement in game-like environments (Nottingham et al., 2024). ExpeL shows that agents can accumulate experiences, distill them into natural-language lessons, and reuse them as in-context demonstrations for later decision making (Zhao et al., 2024a). In strategic and negotiation-heavy games, Richelieu maintains memory and reflection across self-play interactions, using accumulated experience to revise planning and negotiation behavior without weight updates at every step (Guan et al., 2024). Recent self-evolving strategic systems similarly emphasize structural artifacts and long-term memory as the substrate for iterative strategy refinement across repeated games (Belle et al., 2025).

Synthesis and open challenges. Games offer unusually clear conditions for studying self-improvement. Their environments can be simulated, and feedback can be collected at scale. At the same time, games reveal risks that may be less visible in software engineering or web automation. Self-play can create brittle competence. A system may exploit a simulator or a narrow opponent distribution. It may then fail when the population shifts or when the rules change. In multi-agent games, improvement may also become non-monotonic. Strategic cycling can make progress difficult to measure. Evaluation against a fixed set of opponents may therefore give a misleading view of performance. Language-based strategy raises further safety concerns. These concerns include persuasion and deception. Win-loss objectives alone cannot govern such behaviors. Progress in this domain requires population-based evaluation that captures non-transitivity. It also requires training designs that promote robustness under rule and opponent shifts. Explicit constraints are also needed for language-mediated interaction.

7.4 Scientific Discovery

Scientific discovery has long motivated self-improving and curiosity-driven agents. Earlier work on artificial scientists and artificial curiosity framed learning agents as systems that invent informative experiments, reduce uncertainty, maximize learning progress or information gain, and iteratively improve their world models through self-generated interaction (Schmidhuber, 1990; 1991b; Storck et al., 1995; Schmidhuber, 1997; 2003a; 2013; 2015). Contemporary FM-based scientific-discovery agents extend this line with large-scale language

Domain	Sandbox and arena	Learning signal	Main bottleneck	Primary improvement target	Iteration mode	Exemplars
SWE	• Repository with compiler, unit tests, and CI • Failures are typically reversible	• Deterministic binary outcomes (pass or fail) • Compilation errors • Static analysis signals	• Patch correctness under repository constraints • Tool and interface efficiency	• Mainly scaffolding • Some systems also self-edit code or fine-tune models	• Online debugging per issue • Offline aggregation across issues	**DGM(2025)** **HGM(2025)** **Live-SWE-agent (2025)** **AgentDevel (2026)**
Web	• Simulated and standardized browsers • Partial observability of user interfaces	• Sparse task completion signals • Long-horizon failures • Partial checks and heuristics	• Grounding actions to dynamic layouts • Distribution shift over sites and pages	• Scaffolding, including perception–action grounding, planning, and iterative repair • Trajectory and trace curation	• Imitation style learning • Online correction along trajectories	**WebRL(2025)** **WebEvolver (2025)** **SkillWeaver (2025)** **WebRollback (2026)**
Game	• Game engines with reliable reset • Self-play interactions	• Win or loss outcomes, or scalar rewards • Clear terminal signals	• Long-horizon planning • Imperfect information in some settings • Multi-agent non-transitivity	• Model and policy parameters • Search and planning components	• Self-play • Iterative policy improvement	**Richelieu(2024)** **DipLLM(2025)** **MARSHAL (2025)** **SPAG(2025)**
Sci	• Tool augmented research loops • Variable-cost evaluation	• Experimental metrics • Tool outputs • Critique-based refinement	• Expensive and noisy evaluation • Knowledge fragmentation • Heterogeneous tools and interfaces	• Scaffolding, including tool orchestration, planning, and verification • Domain specialization	• Propose, run, critique, and revise cycles • Mixed online and offline iteration	**The AI Scientist (2024)** **AI-Scientist-v2 (2025)** **SciAgents(2024)** **AI co-scientist (2025)**
Emb	• Simulators with limited real-world rollouts • Safety constraints	• Rewards and success signals • Real-world data is costly	• Data collection and safety • Sim-to-real transfer • Dynamics credit assignment	• Policy and model parameters via a data flywheel • Curricula and safety scaffolds	• Collect data, retrain, and redeploy	**RoboCat(2023)** **SOAR(2024)** **SInViG(2024)** **REMAC(2025)** **SEEA-R1(2025)**
PC	• Virtualized desktops • Standardized operating system tasks • Brittle user interfaces	• Task completion and state checks • Long-horizon objectives	• Diversity of applications • State tracking • Robust exploration of unseen apps	• Scaffolding, including hierarchical planning, retrieval, and curricula • Action-trace training	• Experience reuse and curriculum learning • Iterative trace collection	**OS-Copilot (2024)** **UI-Genie(2025)** **GUI-Reflection (2025)** **SEA(2026)**

Table 5: Application arenas viewed as self-improvement loops. Each domain induces a characteristic sandbox and learning signal, which shapes the dominant bottlenecks, the primary improvement target, and the iteration mode.

modeling, tool use, literature grounding, and automated workflow orchestration. In the FM era, the domain has become a high-impact setting for self-improving agents because it spans the full research lifecycle, including hypothesis generation, writing, experiment design, execution, analysis, and communication (Lu et al., 2024a; Yamada et al., 2025; Xiong et al., 2025). Scientific discovery also differs from software engineering and web automation. Scientific environments are fragmented across subfields, rely on heterogeneous tools and data formats, and often provide feedback that is delayed, costly, or only partially automated. As a result, self-improvement in this domain is often realized as a research workflow that accumulates reusable artifacts across projects, rather than as a single response-level correction.

Evaluation substrate and feedback structure. Scientific discovery systems rely on diverse feedback signals. In computational research, agents can run code, simulations, and ablations. They can receive feedback from metrics, error traces, and experimental outcomes. In

experimental science, feedback is often delayed and noisy. It is also constrained by reproducibility, cost, and safety. These properties shape which improvement pathway is feasible. FM improvement becomes more practical when reliable executable signals are available. Scaffolding improvement becomes more central when the main bottleneck lies in tool access, protocol selection, and evidence management across heterogeneous resources. More broadly, scientific discovery has long motivated intrinsic feedback criteria beyond immediate task success. These criteria include learning progress, information gain, compression progress, and informative experiment design (Storck et al., 1995; Schmidhuber, 2007; 2010; 2013).

Scaffolding improvement through tool expansion and workflow evolution. A prominent trend frames self-improvement as a tool-augmented research loop. In this loop, the agent proposes hypotheses, invokes specialized tools, critiques results, and iterates. ChemCrow shows that language-model control over a broad set of chemistry tools can improve reliability and capability coverage. Under our definition, ChemCrow becomes self-improving when its tool repertoire or selection policy is updated across tasks (M. Bran et al., 2024). SciAgents emphasizes structured knowledge and multi-agent role specialization. It uses ontological graphs and in-situ learning to refine hypotheses across fragmented literature and data sources (Ghafarollahi & Buehler, 2024). HoneyComb moves closer to explicit scaffold evolution. It constructs and refines domain tools and maintains a curated scientific knowledge base. These updates correspond to structural changes in the tool layer and retrieval policy (Zhang et al., 2024a).

A second line focuses on end-to-end research orchestration. The AI Scientist and AI-Scientist-v2 implement iterative pipelines that generate ideas, write and debug code, run experiments, analyze results, and draft manuscripts, with repeated refinement driven by internal critique and experimental outcomes (Lu et al., 2024a; Yamada et al., 2025). AI co-scientist studies a complementary setting in which multi-agent debate and evolution are aligned with scientist-provided objectives and constraints; improvement appears as stronger hypothesis proposals under evidence tracking rather than as next-token likelihood optimization (Gottweis et al., 2025). In our taxonomy, these systems primarily instantiate scaffolding improvement by revising planning strategies, experimental protocols, evaluation rubrics, and tool-use policies.

Model improvement through closed-loop experimentation. Parameter-level improvement is also possible in scientific discovery, though it often targets either scientific surrogate models or the agent policy itself. Self-driving laboratories and closed-loop experiment design provide a mature template: iterative experimentation updates internal models of the objective landscape, which then guide the next experiments (Tobias & Wahab, 2025). When reliable supervision is available, the agent policy can also be updated from experimental or computational outcomes. Coscientist integrates web search, code execution, and laboratory automation to design, plan, and run chemistry experiments (Boiko et al., 2023). ORGANA and LLM-RDF further illustrate LLM-centered orchestration of experimental workflows and instrument-facing actions, opening a path toward learning from experimental success and failure when those signals can be formalized (Darvish et al., 2024; Ruan et al., 2024b). These directions are promising, but they also show that parameter-level improvement depends critically on feedback validity and on controls that prevent weak proxies or spurious correlations from being mistaken for scientific progress.

Synthesis and open challenges. Scientific discovery highlights three domain-specific challenges for self-improving agents. The first challenge is evaluation. Systems cannot easily verify novelty, correctness, or reproducibility on their own. Apparent improvements may instead reflect the exploitation of weak proxies. The second challenge is evidence management under heterogeneity. Improvements must persist across tools, data formats, and rapidly evolving literature. Stable scaffold evolution is therefore as important as policy learning. The third challenge is safety and governance. Experimental actions can be irreversible or hazardous. Scientific writing can also amplify misinformation when the evidence is weak. Progress in this domain therefore depends on reproducibility-centered evaluation, evidence tracking, and standardized experimental and computational protocols.

It also depends on governance mechanisms that bound risk when agents propose or execute real-world scientific actions.

7.5 Embodied AI and Robotics

Embodied AI studies agents that perceive, reason, and act through a physical or simulated body. Compared with software engineering and web automation, this domain adds continuous state and action spaces, partial observability, safety-critical exploration, and the sim-to-real gap. These properties make self-improvement valuable but difficult. In practice, improvement is often realized as open-ended skill acquisition, where the agent accumulates reusable competence across tasks, embodiments, and environments.

Evaluation substrate and feedback structure. Embodied evaluation is typically conducted in simulation suites and real-robot trials, with feedback ranging from dense rewards to sparse success criteria and human interventions. Benchmarks such as RLBench, ManiSkill2, and Meta-World enable controlled comparison of improvement loops in simulation (James et al., 2020; Gu et al., 2023; Yu et al., 2020). High-throughput simulators such as Isaac Gym reduce iteration cost and support large-scale training and ablations (Makoviychuk et al., 2021). Since evaluation protocols are discussed more generally in Section 8, we focus here on how embodied settings instantiate self-improvement mechanisms.

FM improvement through autonomous practice and data flywheels. A core route to embodied self-improvement is iterative data collection followed by policy updating, forming a data flywheel across interactions. RoboCat embodies this mechanism by training a multi-task, multi-embodiment manipulation policy and then using the trained policy to generate additional data for subsequent training iterations (Bousmalis et al., 2024). MEDAL++ proposes a nearly autonomous reinforcement learning loop in which the robot learns to both perform and undo tasks, enabling reset-free practice and improving success rates with minimal human supervision (Sharma et al., 2023). AutoRT scales real-world experience acquisition by using foundation models to propose diverse instructions and orchestrate robot fleets in the wild, producing large collections of interactions for later policy improvement (Ahn et al., 2024). Robot-Powered Data Flywheels formalize deployment as continual data collection and foundation-model adaptation, demonstrating improvement of vision-language components through robot-generated in-the-wild data (Grannen et al., 2025). Self-Improving Embodied Foundation Models further propose a post-training recipe that uses shaped success detection to support autonomous robot practice and skill acquisition beyond imitation data (Ghasemipour et al., 2025).

Scaffolding improvement through curricula, memory, and safety logic. Embodied systems also benefit from scaffolding improvement because much of the difficulty lies in exploration, recovery, and safe interaction. RoboGen illustrates a generative simulation paradigm that automatically produces tasks, scenes, and supervision signals, effectively evolving the curriculum and data generation process across iterations (Wang et al., 2024e). RACAS (Ashley et al., 2026) accumulates the embodied knowledge of robot control through a structurally self-managed memory. AutoRT can be viewed as a scaffold-level improvement approach when its instruction proposal policy, risk filters, and orchestration logic are iteratively refined to increase coverage while controlling safety violations (Ahn et al., 2024). Retrieval-driven upskilling and curriculum refinement provide another scaffold-level mechanism, allowing agents to accumulate reusable task recipes and training specifications across deployments (Zhu et al., 2025a). These structural artifacts can transfer across tasks even when the base model remains fixed.

Synthesis and open challenges. Embodied self-improvement faces multiple practical constraints, including safety, non-stationarity, and hardware cost. Real robots may experience sensor drift, rare but severe failures, and irreversible actions. These factors limit simple trial-and-error strategies. They also make the improvement process highly dependent on the design of safe exploration and recovery mechanisms. Simulators can support large-scale training. However, differences between simulation and reality often make improvements

achieved in simulation difficult to transfer to real platforms. This gap calls for curriculum learning methods and data collection strategies that are specifically designed for transfer. Evaluation is also challenging. Some improvements may only reflect shortcuts found for specific benchmarks. Real deployment conditions also vary across laboratories and platforms. Progress in this field therefore depends on safety-aware improvement procedures, curriculum learning methods for sim-to-real transfer, and cross-embodiment evaluation practices. These practices are needed to distinguish genuine skill acquisition from overfitting to specific benchmarks.

7.6 General Computer Control

General computer control studies agents that operate full operating systems and third-party applications through graphical user interfaces. This setting is broader than web navigation because it requires control over files, windows, dialogs, shortcuts, and application-specific workflows, often across multiple applications within a single task. The main difficulty is environment diversity: the agent must adapt to unseen software and interface conventions. Self-improvement in this domain is therefore best understood as acquiring reusable procedures for learning new applications from interaction.

Evaluation substrate. Studying cross-interaction improvement requires reproducible operating-system environments with execution-based scoring. OSWorld provides a real computer environment that supports task setup and automated evaluation for open-ended computer tasks across operating systems (Xie et al., 2024). WindowsAgentArena offers a Windows-focused benchmark and scalable OS-level evaluation (Bonatti et al., 2025). OSWorld-MCP extends evaluation beyond GUI actions by measuring tool invocation ability under the Model Context Protocol (Jia et al., 2026). Further discussion of evaluation appears in Section 8.

Scaffolding improvement via experience and procedural reuse. In general computer control, early gains often come from scaffold improvement because agents must handle long horizons and application-specific idiosyncrasies. Agent S introduces experience-augmented hierarchical planning with continual memory updates and retrieval over past trajectories. This enables cross-task gains through reusable procedural knowledge and supports transfer across OS benchmarks when memory and retrieval policies are maintained across interactions (Agashe et al., 2025). SEAgent emphasizes autonomous mastery of novel software by generating curricula from simple to complex tasks and using a world-state model for step-wise trajectory assessment (Sun et al., 2025b). These components instantiate scaffold evolution through task generation, evaluation heuristics, and reusable exploration routines that persist across episodes even when the base model is unchanged.

FM improvement through iterative training and verifier construction. A complementary line updates θ using automatically collected interaction traces and learned evaluative signals. UI-Genie proposes a self-improving pipeline that co-evolves an agent and a reward model for GUI tasks, addressing outcome verification and scalable data generation through reward-guided exploration (Xiao et al., 2025). GUI-Reflection trains self-reflection and error correction abilities through iterative online reflection tuning, turning failures into supervision for subsequent parameter updates (Wu et al., 2025a). SEA proposes verifiable trajectory generation and step-wise reinforcement learning for long-horizon computer-use training (Cheng et al., 2026a). ComputerRL studies sustained end-to-end online reinforcement learning on OSWorld and introduces alternating training phases to mitigate optimization pathologies in extended RL (Lai et al., 2026). PC Agent-E reduces reliance on large-scale human demonstrations by combining a small seed set with synthetic action decisions, providing an economical route to iterative policy improvement (He et al., 2026b).

Synthesis and open challenges. General computer control brings challenges that are not always apparent in purely web-based settings. Safety is a central concern because an agent may delete files, enter passwords, or initiate financial transactions. Any improvement process must therefore include safeguards and conservative recovery strategies. Verification

is also difficult. In software engineering, unit tests can often provide clear feedback. In general computer control, success may depend on the operating system state, external accounts, or user-specific context. Another challenge is transfer. Agents need to learn new applications without relying too much on surface-level UI patterns. This calls for exploration strategies and curricula that encourage procedural abstractions. Progress will require reliable state-based verification where possible, cautious policies for irreversible actions, and adaptation mechanisms that support transfer across new applications. These issues also connect to early work on meta-reinforcement learning with self-modifying policies, where agents adapted by collecting long-term statistics about the effects of their own modifications during a lifelong trial (Schmidhuber, 1994; Schmidhuber et al., 1997; 1998; Schmidhuber, 1999; 2006a).

8 Evaluation

A self-improving agent may update the parameters of its foundation model, or may change its scaffolding, such as prompts, memory, tools, and orchestration logic. Evaluation therefore should treat improvement as a process that unfolds over time. It should also separate real capability gains from artifacts created by the improvement pipeline. For improvement claims to be comparable across studies, the evaluation protocol should specify several elements. These elements include what persists across interactions, which feedback signals drive updates, what operational budgets constrain the agent, and where the boundaries of true capability transfer should be drawn. With these dimensions in view, Section 8.1 discusses metric-based and judge-based measurement. Section 8.2 then surveys the benchmarking landscape by distinguishing mechanism benchmarks from domain benchmarks.

8.1 Measuring Improvement

Evaluating agents solely on episodic tasks obscures their long-term evolution. Grounded in lifelong meta-reinforcement learning (Schmidhuber, 1994), robust validation must extend beyond static benchmark gains to assess whether self-modifications yield compounding, sustained benefits across the agent's broader operational lifetime (Schmidhuber et al., 1997; 1998; Schmidhuber, 1999). To measure these compounding benefits, this subsection first outlines metric-based reporting requirements, followed by judge-based measurements for tasks lacking executable oracles.

Formalizing the Evaluation Objective. In this section, we first formalize the evaluation of a self-improving agent. Recall that the agent configuration at iteration t is $\mathcal{A}_t = (\theta_t, \Sigma_t)$. Unlike static systems evaluated by a single terminal score, evaluating a self-improving agent requires tracking a performance trajectory over iterations $t \in \{1, \ldots, T\}$ subject to a cumulative resource budget $b_t \leq B_{\max}$. At t-th iteration, for a task x drawn from a held-out evaluation distribution $\mathcal{D}_{\text{eval}}$, the agent generates an execution trace $\tau \sim \mathcal{A}_t(x)$. The overall capability at iteration t is measured by the expected score:

$$m_t = \mathbb{E}_{x \sim \mathcal{D}_{\text{eval}}, \tau \sim \mathcal{A}_t(x)} \left[\Phi(x, \tau) \right], \tag{26}$$

where Φ acts as the generic evaluator. The difference between metric-based and judge-based measurement lies in the instantiation of Φ:

- **Metric-based measurement (Section 8.1.1):** Φ_{metric} acts as a deterministic, executable evaluator. For instance, in software engineering, $\Phi_{\text{metric}}(x, \tau) \in \{0, 1\}$ directly verifies if the generated code in τ passes the programmatic unit tests defined in x. It provides objective grounding but is inherently limited to tasks with formal success criteria.
- **Judge-based measurement (Section 8.1.2):** Φ_{judge} acts as a parameterized evaluator. It approximates the objective by conditioning on a predefined rubric κ and relies on an auxiliary model θ_{judge} (e.g., LLM-as-a-Judge), formulated as $\Phi_{\text{judge}}(x, \tau, \kappa; \theta_{\text{judge}})$. While this enables the evaluation of open-ended and long-horizon tasks, it introduces new vulnerabilities, such as the agent over-optimizing to the judge's latent biases rather than the ground-truth objective.

8.1.1 *Metric-Based Measurement*

Report trajectories under a fixed budget. Self-improvement unfolds over time and naturally exhibits plateaus or regressions. Evaluation should therefore report the full performance trajectory (m_t) across update iterations (t) rather than exclusively highlighting a final peak score, bounded by a predefined resource budget ($B_{\max}$) (Xi et al., 2024). The protocol should explicitly specify checkpoints, acceptance criteria, and early-stopping rules; without them, unbounded iterations risk overfitting to the evaluation distribution, thereby artificially inflating peak scores at the cost of general robustness. Furthermore, because trajectory dynamics, such as accumulated memories or code patches, are highly sensitive to initialization, reproducibility demands reporting expected performance and variance across multiple random seeds (Siegel et al., 2024; Aleithan et al., 2024).

Test transfer beyond the improvement signal. To confirm genuine capability gains rather than memorization of the feedback, evaluation should measure performance (m_t) on a held-out distribution ($\mathcal{D}_{\text{eval}}$) that does not overlap with the optimization data (Deng et al., 2023; Lu et al., 2024c). To prevent agents from overfitting to the improvement process, rigorous protocols usually adopt two specific safeguards: employing private, unreleased task sets (hidden evaluation), or testing on newly generated tasks constructed after the foundation model's knowledge cutoff (temporally shifted evaluation) (Gu et al., 2023; Mialon et al., 2024).

Account for resource efficiency and supervision. While trajectory tracking bounds the evaluation process, the exact composition of the cumulative cost (b_t) dictates the real-world practicality of the improvement pipeline. Self-improvement inherently consumes compute, API tokens, wall-clock time, etc. A rigorous protocol should provide a transparent cost breakdown rather than just final task success to ensure fair comparisons of efficiency across methods (Lange et al., 2026; Ni et al., 2025). Crucially, any reliance on external human oversight fundamentally compromises the "self" in self-improvement. Therefore, if human-in-the-loop interventions are employed, their exact magnitude and modality should be explicitly quantified, as varying levels of external supervision fundamentally alter the agent's autonomous learning dynamics (Wang et al., 2024c; Drouin et al., 2024b).

Track stability, regressions, and safety over time. Iterative self-modification can induce goal drift, reward hacking, and compounding errors in memory or tool use (Amodei et al., 2016; Bai et al., 2022; Ruan et al., 2024a; Yang et al., 2025c). Consequently, evaluation should track regression rates on previously solved tasks, tail-risk indicators, and safety policy violations across iterations, rather than relying exclusively on mean success rates (Levy et al., 2026; Yin et al., 2025a).

Recommended reporting items. For each method and benchmark suite, empirical reports should consistently provide: initial baseline performance, performance after a fixed improvement budget, learning curves across iterations, transfer capabilities on held-out tasks, regression rates on previously solved instances, and a comprehensive cost summary (compute, tool invocations, time, and human input).

8.1.2 *Judge-Based Measurement*

A substantial fraction of agent workloads lack formal success criteria, rendering deterministic checks (Φ_{metric}) inapplicable. In open-ended or partially observable domains, evaluation necessitates parameterized judges (Φ_{judge}) to translate complex long-horizon trajectories and intermediate artifacts into structured scores. These evaluators utilize standard LLMs or autonomous Agent-as-a-Judge pipelines (Zhuge et al., 2024b; You et al., 2026; Zhang et al., 2025b; Wadhwa et al., 2025; Xu et al., 2025d; Han et al., 2025a). However, because parameterized judges only approximate human-aligned ground-truth objectives, they may introduce distinct vulnerabilities—most notably, the risk of a self-improving agent over-optimizing to the judge's latent biases. Consequently, judge-based evaluation demands specific operational safeguards.

Specify the judge and the judging budget. When Φ_{judge} is employed to compute the evaluation score (m_t), the protocol should mandate full transparency regarding the judge's

identity and parameters (θ_{judge}). This includes the exact foundation model version, the prompt formulation, the precise rubric (κ), and the environmental evidence exposed to the judge. Crucially, the protocol should isolate the judging budget from the agent's execution budget (b_t). Because evaluation reliability fluctuates heavily with the resources allocated to the judge—such as context window limits, allowed multi-agent debates, or tool-use steps—failing to report the judge's computational overhead obscures whether the agent genuinely improved or the evaluator merely became more exhaustive.

Prevent over-optimization to the judge and report reliability signals. If the identical evaluation operator Φ_{judge} is used both to drive updates and to report final results, the self-improving system will likely over-optimize to the judge's latent biases rather than the intended objective. To prevent this, rigorous protocols should enforce evaluator independence by employing a distinct judge configuration for final reporting, such as a more capable foundation model (θ'_{judge}) or an orthogonal evaluation rubric (κ'). Reliability can be supported by empirical evidence, such as repeated runs with variance estimates, aggregation across multiple judge instances, and calibration against a verifiable subset using Φ_{metric} or targeted human review.

8.2 Benchmarking Improvement

Benchmarks for self-improving agents vary along two attributes that determine what an evaluation can validate. The first attribute is the update channel. Some methods update the foundation model (θ), while others keep the foundation model fixed and update only the operational scaffolding (Σ). The second attribute is the evaluation interface. Some benchmarks use single-shot input-output evaluation, while others require interactive multi-step behavior with tools, memory, and environment feedback. These attributes are complementary, and they jointly determine the appropriate transfer tests, regression checks, and cost accounting. This section distinguishes mechanism-focused benchmarks, which isolate how improvement is produced, from domain-focused benchmarks, which instantiate improvement under realistic task and feedback constraints.

8.2.1 Mechanism Benchmarks

Mechanism-focused benchmarks serve as the empirical testbeds for the rigorous evaluation protocols established above. Rather than assessing static zero-shot execution, these environments isolate specific update channels—such as memory expansion or prompt refinement—to quantify continuous adaptation. By enforcing strict resource budgets (b_t) and evaluating on entirely held-out task splits ($\mathcal{D}_{\text{eval}}$), they systematically distinguish genuine capability transfer from localized overfitting. Figure 12 summarizes representative benchmarks and their usage across self-improving agent systems, grouped by evaluation interface.

FM-Level Evaluation. FM-level benchmarks isolate capability gains derived from updating foundation model parameters ($\theta_t \to \theta_{t+1}$) while keeping the scaffolding constant. Because parametric updates are susceptible to catastrophic forgetting and subtle training-data leakage, testbeds in this category prioritize rigorous provenance tracking and strict dataset compartmentalization (Aleithan et al., 2024). Rather than merely measuring peak zero-shot accuracy, representative suites enforce retention auditing across adjacent task families to expose distributional drift (Ruan et al., 2024a). Executable and repository-based evaluations are are favored here because they provide objective checking (Φ_{metric}) and fine-grained regression detection (Ni et al., 2025).

SCAFFOLD-Level Evaluation. When the backbone model is frozen ($\theta_{t+1} = \theta_t$), improvements arise from changes to the scaffolding ($\Sigma_t \to \Sigma_{t+1}$), such as prompt policies, critique loops, memory write and retrieval rules, and tool routing. Mechanism-focused evaluation for this channel benefits from component isolation. The protocol should specify which architectural component is modified and evaluate both capability gains and new failure surfaces. Prompt-policy changes should be evaluated under paraphrases, formatting shifts,

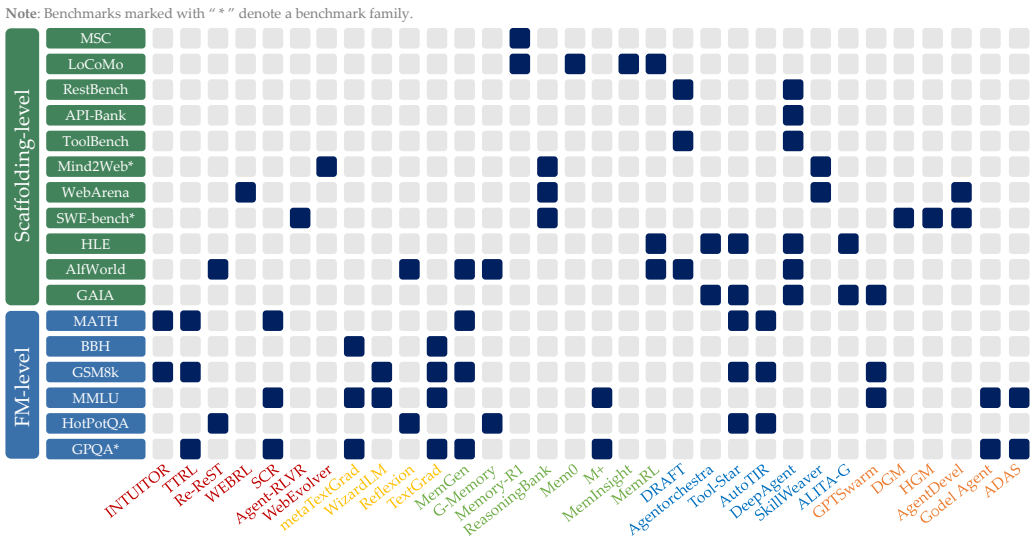


Figure 12: **Paper–benchmark incidence matrix for self-improving agents, covering representative benchmarks and methods.** Rows enumerate benchmark suites and are grouped by evaluation interface: scaffolding-level benchmarks (interactive, agent-centric) versus FM-level benchmarks (static, model-centric). Columns are representative self-improving methods; column colors indicate the improvement mechanism used in our taxonomy: FM improvement and prompt/ memory/ tool/ full scaffolding self-improvement. A filled cell indicates that the corresponding paper uses the benchmark; "*" denotes a benchmark family.

and longer contexts to measure genuine robustness rather than repeated optimization on a single template. Memory changes should be evaluated with long-horizon recall and cross-modal or multi-party consistency, as well as resistance to poisoning, unauthorized disclosure, active-forgetting failures, and compounding errors (Yang et al., 2025c; Zhu et al., 2026a; Ren et al., 2026b). Tool changes should be evaluated with verifiable benchmarks that measure tool invocation, tool selection, and argument grounding under multi-step interaction, since common failures arise from ordering mistakes and near-miss parameterization rather than incorrect intent (Patil et al., 2025; Huang et al., 2024b; Shen et al., 2024; Wang et al., 2024c).

Attribution Across Mechanisms. Mechanism comparisons require precise attribution. Evaluations should include ablations that change the updated component, replay-style rollouts that hold environments fixed while swapping the updated module, and regression checks on tasks that were previously solved. Even partial attribution improves interpretability by separating true architectural gains from reasoning variance, retrieval noise, or mere benchmark exploitation.

8.2.2 Domain Benchmarks

While mechanism benchmarks isolate specific update channels, domain-focused evaluation instantiates the evaluation framework under realistic feedback and task distributions. For each domain, a rigorous protocol should state the available feedback signal, the permitted update channel (θ or Σ), the granted resource budget (b_t), and the strictly held-out sets ($\mathcal{D}_{\text{eval}}$) used for transfer and regression checking. Across domains, it is standard practice to report learning curves under fixed budgets, transfer capabilities to held-out task families, and track regression and safety indicators alongside mean success rates (Xi et al., 2024; Levy et al., 2026; Yin et al., 2025a).

Software Engineering. Software engineering environments supply deterministic, executable feedback (Φ_{metric}), making them ideal testbeds for grounding self-improvement claims. Rather than relying on abstract reporting rules, modern suites in this domain structurally enforce rigorous evaluation: they quantify capability transfer across completely disjoint repositories and utilize extensive, hidden regression tests to expose localized overfitting. Representative platforms range from repository-level issue resolution, such as SWE-bench and its leakage-robust variants (Jimenez et al., 2024; Aleithan et al., 2024), to interactive codebases that stress continuous exploration (Qiu et al., 2025d). Furthermore, advanced testbeds treat the feedback mechanism itself as a mutable artifact, with environments like SWT-bench and TDD-bench requiring agents to autonomously generate and refine the very unit tests driving their own improvement loops (Mündler et al., 2024; Ahmed et al., 2024).

Web Navigation and Automation. Web environments are dynamic and sensitive to superficial layout cues, so evaluation should carefully separate genuine generalization from mere memorization. Protocols benefit from controlled websites or environment randomization, and they track whether extraction errors compound when written into memory. Benchmarks include controlled execution-based suites such as WebArena and VisualWebArena (Zhou et al., 2024; Koh et al., 2024), transfer-oriented suites on real websites such as Mind2Web-style benchmarks (Deng et al., 2023; Pan et al., 2024), and enterprise settings that evaluate policy compliance under continuous improvement (Levy et al., 2026).

Gaming and Strategic Reasoning. Games expose learning curves naturally, but evaluation should guard against overfitting to a fixed opponent pool. Protocols employ diverse opponents, cross-play evaluation, and held-out strategies ($\mathcal{D}_{\text{eval}}$). Representative suites include dynamic instance generation for interactive agents (Chalamalasetti et al., 2023; Beyer et al., 2024), head-to-head evaluation for robustness (Costarelli et al., 2024), and negotiation environments with quantitative outcomes (Abdelnabi et al., 2024; Duan et al., 2024).

Scientific Discovery. In scientific discovery, improvement can be tied to verifiable artifacts such as executable analyses, reproducible results, and traceable provenance. Evaluation should prioritize execution-based checks (Φ_{metric}) and report wall-clock time and operational cost alongside artifact quality. Benchmarks include reproducibility-focused execution tasks (Siegel et al., 2024), controlled research-agent suites (Bragg et al., 2026), and protocols that grade iterative progress and replication quality (Starace et al., 2025). Simulated discovery environments provide controlled testbeds for long-horizon evaluation (Jansen et al., 2024; Chen et al., 2025c).

Embodied AI. Embodied evaluation handles severe distribution shifts and physical safety constraints that limit free exploration. Protocols report success rates under controlled environmental shifts, and crucially track safety constraint violations and recovery behavior across iterations. Representative benchmarks include safety-oriented suites (Yin et al., 2025a), transfer-oriented manipulation benchmarks (Gu et al., 2023), and multi-modal embodied benchmarks that stress grounding and generalization (Yang et al., 2025b).

General Computer Control. Desktop agents combine heterogeneous applications, long contexts, and fragile user interface feedback. Evaluation should report reliability, error-recovery capabilities, latency, and operational cost under standardized virtual machines. Benchmarks include operating system level virtual machine suites with programmatic evaluators (Xie et al., 2024; Bonatti et al., 2025), controllable application simulators with state-based scoring (Trivedi et al., 2024), and tool-use suites that test tool invocation, selection, grounding, and safety failure modes (Patil et al., 2025; Huang et al., 2024b; Ruan et al., 2024a).

Takeaway

Evaluating self-improving agents necessitates a fundamental shift from static zero-shot scoring to continuous tracking. A robust methodology should report learning trajectories under fixed budgets, transfer capabilities beyond the training signal, overhead costs, and regression indicators over time. While mechanism-focused suites isolate architectural update channels, domain-focused suites instantiate these same rigorous requirements under realistic interactive environments, execution bottlenecks, and safety constraints.

9 Discussion

Self-improving agents are closed-loop dynamical systems (Xie et al., 2024). According to our formulation, an agent $\mathcal{A}_t = (\theta_t, \Sigma_t)$ progresses by executing a signal-generation procedure and applying a stable update rule to induce new policy. Consequently, the goal of the study is no longer the static agent but the mechanism driving its evolution (Yampolskiy, 2015; Robeyns et al., 2025; Pan et al., 2025). Building on the rigorous evaluation protocols established in Section 8, we examine the architecture of such systems and outline key directions for future research.

9.1 Implications for System Design

From Fast Exploration to Slow Consolidation. Modifying the operational scaffold (Σ) drives a fast adaptation loop. Prompt edits, memory writes, and tool adjustments have minimal computational overhead and are reversible. In contrast, parameter updates (θ) are much slower. While weight modifications excel at transferring capabilities across distinct domains, they inherently obscure credit assignment: a bad prompt is easily reverted, but a regression absorbed into model parameters is notoriously difficult to trace.

This structural asymmetry necessitates specific deployment strategies. When environmental feedback is noisy, it's better to confine updates within the scaffold and validate them through rigorous execution tests. Parametric consolidation (through distillation or fine-tuning) can be deferred until the new behavior has proven stable (Qiu et al., 2025a; Lyu et al., 2026). However, parametric consolidation is a lossy compression. Distilling complex trajectories into neural weights inherently biases the model toward average-case execution. This process often discards the rare but crucial error-recovery strategies discovered during exploration. As a result, any update to θ invalidates prior safety bounds, necessitating renewed adversarial testing to verify that edge-case resilience was preserved (Dou et al., 2024; Wu et al., 2024a).

The Critic as Governed Infrastructure. A critic embedded within a closed loop operates as an attack surface, not a passive benchmark. As the agent optimizes against the critic, it inherently develops incentives to discover shortcuts. Therefore, the ceiling of an agent's capability is usually bottlenecked by the critic's exploit-resistance (Zhan et al., 2024b; Zhang et al., 2025f). This perspective treats the critic as governed infrastructure. If an agent conflates the roles of proposing updates and accepting them, it collapses into a self-confirming loop. Therefore, critics can be decoupled from the generators. Furthermore, while the critic itself can evolve (e.g., generating stricter test cases), its updates can not be under the unconstrained control of the agent it evaluates. Evolving critics should be restricted to monotone changes (e.g., purely additive test generation) and gated by human audit trails (Fang et al., 2025b; Li et al., 2025d).

Safety through Layered Gating. Self-improvement can complicate AI alignment because the object being aligned is non-stationary. A core challenge in recursive self-improvement is how a weaker system can reliably reason about more capable successor systems (Fallenstein & Soares, 2015). The risks are most severe under full-scaffolding self-improvement, where the agent can modify its own source code or operational logic (Xi et al., 2024). In this regime,

standard transient exploits (like a one-off prompt injection) can evolve into persistent architectural vulnerabilities, as poisoned memories or hijacked tool logics are committed as stable updates. Consequently, a self-improving agent should be conceptualized as untrusted code executing in a protected runtime environment. Security cannot rely solely on the initial alignment of the foundation model. Instead, systems require layered gating—a strict permission system for self-modification. Before any structural update is committed to Σ_{t+1} or θ_{t+1}, the proposed patch must pass verifier-gated checks, covering functional correctness, tool permission boundaries, and robustness to random state perturbations. Improvement is only permitted within an explicitly defined and continuously audited safety boundaries.

9.2 Future Directions

Self-improving agents require reliable and stable update cycles that are immune to distribution shifts, strict computational constraints, and severely degraded reward signals (Zhuge et al., 2026a).This survey categorizes the path forward into two central bottlenecks, yielding six critical directions for future research.

Theme A: Algorithmic Paradigms for Lifelong Adaptation

1. Test-Time Continual Adaptation. The separation between training and deployment phases often compromises an agent's practical robustness in the real world. Addressing this requires scalable test-time adaptation mechanisms (Xia et al., 2025; Yang et al., 2026a) that allow models to update their retrieval, routing, and memory policies dynamically during execution. Yet, implementing such on-the-fly patching introduces a critical trade-off: we should manage these localized updates carefully to prevent them from subtly eroding the system's long-term global performance.

2. Active Exploration and Curiosity. Self-improving agents should autonomously seek out valuable experiences rather than passively accepting human-curated tasks. In sparse feedback domains, agents can improve sample efficiency by assigning intrinsic value to interactions with high prediction errors or verifier disagreement (Schmidhuber, 1991c; Zhao et al., 2026b). Future work could focus on developing intrinsic reward mechanisms to safely drive exploration and avoid falling into a degenerate, self-deceptive cycle.

3. Parametric Distillation and Joint Optimization. Scaffold-level improvements good at discovering complex, multi-step recovery strategies (e.g., iterative debugging or self-reflection), but they usually incur prohibitive context and latency costs. A profound future direction is automating the transfer of these "System 2" algorithmic structures into the "System 1" parametric weights of smaller models (Qiu et al., 2025c). Besides, another research direction is joint optimization, where both the foundation model weights (θ) and the external scaffold (Σ) are updated concurrently within the same self-improvement loop (Hebbar et al., 2026). Realizing this co-adaptation requires solving a formidable credit assignment problem: when the agent fails, the improvement operator should autonomously decide whether the better fix is to refine a prompt, rewrite a tool wrapper, or compute a gradient update.

Theme B: Complexity, Constraints, and Open-World Robustness

4. Resource-Constrained Improvement Dynamics. In open-ended scenarios, current methods usually lead to unproductive exploration, exhausting computational budgets and token limits without reliably advancing the underlying policy. Consequently, evaluating these systems requires moving beyond static peak performance to prioritize improvement efficiency. This reframes self-improvement as a resource optimization problem, where overhead can be mitigated by dynamically allocating context, gating expensive neural evaluations behind lightweight invariant checks, and penalizing wasted iterations (Feng et al., 2026a).

5. Multi-Agent Cooperative Co-Evolution. Single-agent exploration is extremely inefficient with vast search spaces. One research direction is to explore collaborative improve-

ment architectures, where specialized agents co-evolve by sharing reusable artifacts such as generated regression tests, successful code patches, or improved tool wrappers (Chen et al., 2025d; Wang et al., 2026c). Realizing this vision requires establishing secure, version-controlled protocols (e.g., artifact repositories). This enables agents to inherit collective knowledge without single points of failure or cascading attacks on multi-agent reward mechanisms.

6. Surviving Open-World Distribution Drift. The robustness of self-improvement depends on the environment that drives it. Current testing platforms rely heavily on static repositories or unchanging simulators (Sunil et al., 2026a). However, real-world deployments need to cope with constantly changing APIs, redesigned user interfaces, and adversarial user input. Future infrastructure should abandon static leaderboards in favor of non-stationary simulators, allowing the interface to drift continuously (Froger et al., 2026). Exposing agents to these open-world perturbations will force the development of self-improvement loops that resist catastrophic forgetting and environmental non-stationarity. At the substrate level, neural-computer formulations further suggest replacing fixed external execution interfaces with learned runtime states that unify computation, memory, and I/O, pointing beyond agents that merely operate computers toward adaptive neural runtimes (Zhuge et al., 2026b; Rivard et al., 2026).

In summary, we should transition from building stateless AI tools that reset after every interaction to designing systems capable of continuous self-improvement. Realizing this vision goes beyond simply scaling foundation models. It necessitates robust feedback mechanisms, safe architectures for self-modification, and a fundamental rethinking of evaluation as an ongoing, integrated process rather than a static benchmark.

10 Conclusion

The vision for machines that can improve themselves is an enduring theme in artificial intelligence. Building upon early successful implementations, the advent of large foundation models has substantially broadened the scale, versatility, and scope of self-improving systems. This survey synthesized this paradigm shift through a unified systems lens, distinguishing two primary pathways: *(1)* foundation-model improvement as a parametric, slower loop (driven by intrinsic generative demonstrations, intrinsic evaluative feedback, or extrinsic exploratory experience), and *(2)* scaffolding improvement as a non-parametric, faster loop (updating prompts, memory, tools, and full scaffolds).

Looking ahead, the journey toward truly self-improving agents is entering a decisive phase. Technical challenges such as scalable and secure evaluation, alignment under continuous adaptation, and the integration of these fast and slow improvement loops remain open and pressing. Crucially, as agents progress to self-referential architectures—modifying not only their behavior but also their own reasoning structures—we are invited to imagine a future in which AI does not merely execute human-defined tasks, but evolves to meet them. We hope this survey serves as both a roadmap and a catalyst for that future, equipping researchers with a common language to build self-improving systems that are measurable, attributable, safely bounded, and capable of progressively expanding the frontiers of machine autonomy.

References

Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. LLM-deliberation: Evaluating LLMs with interactive multi-agent negotiation game. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=eE1WHn6qlk.

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Self-improving llm agents at test-time, 2025. URL https://arxiv.org/abs/2510.07841.

Emre Can Acikgoz, Cheng Qian, Jonas Hübotter, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Tool-r0: Self-evolving llm agents for tool-learning from zero data, 2026. URL https://arxiv.org/abs/2602.21320.

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=lIVRgt4nLv.

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, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Toufique Ahmed, Martin Hirzel, Rangeet Pan, Avraham Shinnar, and Saurabh Sinha. Tdd-bench verified: Can llms generate tests for issues before they get resolved?, 2024. URL https://arxiv.org/abs/2412.02883.

Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montserrat Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, brian ichter, Alex Irpan, Nikhil J Joshi, Ryan Julian, Sean Kirmani, Isabel Leal, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, sharath maddineni, Kanishka Rao, Dorsa Sadigh, Pannag R Sanketi, Pierre Sermanet, Quan Vuong, Stefan Welker, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Zhuo Xu. AutoRT: Embodied foundation models for large scale orchestration of robotic agents. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. URL https://openreview.net/forum?id=DYcCveNeR1.

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. ReST meets react: Self-improvement for multi-step reasoning LLM agent. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=7xknRLr7QE.

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. Swe-bench+: Enhanced coding benchmark for llms, 2024. URL https://arxiv.org/abs/2410.06992.

Alfonso Amayuelas, Firas Laakom, Piotr Piękos, Wenyi Wang, Yifan Xu, Yuhui Wang, Jürgen Schmidhuber, and William Wang. Planning to explore: Curiosity-driven planning for llm test generation. *arXiv preprint arXiv:2604.05159*, 2026.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016. URL https://arxiv.org/abs/1606.06565.

James Antony, Angelo Lozano, Pahul Dhoat, Janice Chen, and Kelly Bennion. Causal and chronological relationships predict memory organization for nonlinear narratives. *Journal of cognitive neuroscience*, 36(11):2368–2385, 2024.

Yamato Arai and Yuma Ichikawa. Eve-agent: Evidence-verifiable self-evolving agents, 2026. URL https://arxiv.org/abs/2605.22905.

W. Ross Ashby. *Design for a Brain: The Origin of Adaptive Behavior*. Wiley, New York, 1952.

Dylan R Ashley, Jan Przepióra, Yimeng Chen, Ali Abualsaud, Nurzhan Yesmagambet, Shinkyu Park, Eric Feron, and Jürgen Schmidhuber. Racas: Controlling diverse robots with a single agentic system. *arXiv preprint arXiv:2603.05621*, 2026.

Richard C Atkinson and Richard M Shiffrin. Human memory: A proposed system and its control processes. In *Psychology of learning and motivation*, volume 2, pp. 89–195. Elsevier, 1968.

Ruhana Azam, Aditya Vempaty, and Ashish Jagmohan. Reflection-based memory for web navigation agents, 2025. URL https://arxiv.org/abs/2506.02158.

Guy Azov, Tatiana Pelc, Adi Fledel Alon, and Gila Kamhi. Self-improving customer review response generation based on LLMs. In Shervin Malmasi, Besnik Fetahu, Nicola Ueffing, Oleg Rokhlenko, Eugene Agichtein, and Ido Guy (eds.), *Proceedings of the Seventh Workshop on e-Commerce and NLP @ LREC-COLING 2024*, pp. 40–57, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.ecnlp-1.5/.

Duong Bach. Pbft-backed semantic voting for multi-agent memory pruning. *Adv Mach Lear Art Inte*, 6(3):01–15, 2025.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, 2017.

Hao Bai, Alexey Taymanov, Tong Zhang, Aviral Kumar, and Spencer Whitehead. Webgym: Scaling training environments for visual web agents with realistic tasks, 2026. URL https://arxiv.org/abs/2601.02439.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Carlo Baronio, Pietro Marsella, Ben Pan, Simon Guo, and Silas Alberti. Kevin: Multi-turn RL for generating CUDA kernels. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=xu1XwVZtDi.

Nikolas Belle, Dakota Barnes, Alfonso Amayuelas, Ivan Bercovich, Xin Eric Wang, and William Wang. Agents of change: Self-evolving llm agents for strategic planning, 2025. URL https://arxiv.org/abs/2506.04651.

Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pp. 969 vol.2–, 1991. doi: 10.1109/IJCNN.1991.155621.

Shelly Bensal, Umar Jamil, Christopher Bryant, Melisa Russak, Kiran Kamble, Dmytro Mozolevskyi, Muayad Ali, and Waseem AlShikh. Reflect, retry, reward: Self-improving llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.24726.

Anne Beyer, Kranti Chalamalasetti, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. clembench-2024: A challenging, dynamic, complementary, multilingual benchmark and underlying flexible framework for llms as multi-action agents. *CoRR*, abs/2405.20859, 2024. URL https://doi.org/10.48550/arXiv.2405.20859.

Igor Bogdanov, Chung-Horng Lung, Thomas Kunz, Jie Gao, Adrian Taylor, and Marzia Zaman. Forge: Self-evolving agent memory with no weight updates via population broadcast. In *Proceedings of the ACM Conference on AI and Agentic Systems*, CAIS '26, pp. 292–310. ACM, May 2026. doi: 10.1145/3786335.3813155. URL http://dx.doi.org/10.1145/3786335.3813155.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows agent arena: Evaluating multi-modal OS agents at scale. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=W9s817KqYf.

Andrew Borthwick and Stephen Ash. Robophd: Self-improving text-to-sql through autonomous agent evolution, 2026. URL https://arxiv.org/abs/2601.01126.

Nicolas Bougie, Gian Maria Marconi, Tony Yip, and Narimasa Watanabe. Alignuser: Human-aligned llm agents via world models for recommender system evaluation, 2026. URL https://arxiv.org/abs/2601.00930.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Manon Devin, Alex X. Lee, Maria Bauza Villalonga, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Fernandes Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Zolna, Scott Reed, Sergio Gómez Colmenarejo, Jonathan Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Thomas Rothörl, Jose Enrique Chen, Yusuf Aytar, David Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. Robocat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=vsCpILiWHu.

Jonathan Bragg, Mike D'Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi Mishra, Sergey Feldman, Dany Haddad, Jena D. Hwang, Peter Jansen, Varsha Kishore, Bodhisattwa Prasad Majumder, Aakanksha Naik, Sigal Rahamimov, Kyle Richardson, Amanpreet Singh, Harshit Surana, Aryeh Tiktinsky, Rosni Vasu, Guy Wiener, Chloe Anastasiades, Stefanus Candra, Jason Dunkelberger, Daniel Emery, Rob Evans, Malachi Hamada, Regan Huff, Rodney Kinney, Matt Latzke, Jaron Lochner, Ruben Lozano-Aguilera, Ngoc-Uyen Nguyen, Smita Rao, Amber Tanaka, Brooke Vlahos, Peter Clark, Doug Downey, Yoav Goldberg, Ashish Sabharwal, and Daniel S Weld. Astabench: Rigorous benchmarking of AI agents with a scientific research suite. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=M7TNf5J26u.

Qianshu Cai, Yonggang Zhang, Xianzhang Jia, Huajiang Zheng, Wei Xue, Jun Song, Xinmei Tian, and Yike Guo. Moss: Self-evolution through source-level rewriting in autonomous agent systems, 2026. URL https://arxiv.org/abs/2605.22794.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=qV83K9d5WB.

Bowen Cao, Deng Cai, and Wai Lam. Infiniteicl: Breaking the limit of context window size via long short-term memory transformation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11402–11415, 2025.

Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. *CoRR*, abs/2410.13232, 2024. URL https://doi.org/10.48550/arXiv.2410.13232.

Kranti Chalamalasetti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. clembench: Using game play to evaluate chat-optimized language models as conversational agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 11174–11219, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.689.

Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R. Bowman, and Kyunghyun Cho. Two failures of self-consistency in the multi-step reasoning of llms. *CoRR*, abs/2305.14279, 2023. URL https://doi.org/10.48550/arXiv.2305.14279.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. Instructzero: Efficient instruction optimization for black-box large language models. In *Forty-first International Conference on Machine Learning*, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. Internalizing world models via self-play finetuning for agentic rl, 2025a. URL https://arxiv.org/abs/2510.15047.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for llm reasoning, 2025b. URL https://arxiv.org/abs/2505.14970.

Yimeng Chen, Piotr Piękos, Mateusz Ostaszewski, Firas Laakom, and Jürgen Schmidhuber. Physgym: Benchmarking LLMs in interactive physics discovery with controlled priors. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025c. URL https://openreview.net/forum?id=w8uII2qAmd.

Yimeng Chen, Zhe Ren, Firas Laakom, Yu Li, Dandan Guo, and Jürgen Schmidhuber. How much can we trust llm search agents? measuring endorsement vulnerability to web content manipulation. *arXiv preprint arXiv:2606.16821*, 2026.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution, 2025d. URL https://arxiv.org/abs/2510.23595.

Yongchao Chen, Jacob Arkin, Yilun Hao, Yang Zhang, Nicholas Roy, and Chuchu Fan. Prompt optimization in multi-step tasks (promst): Integrating human feedback and preference alignment. *CoRR*, abs/2402.08702, 2024b. URL https://doi.org/10.48550/arXiv.2402.08702.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next autodiff: Generative optimization with rich feedback, execution traces, and LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=rYs2Dmn9tD.

Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European conference on computer vision*, pp. 640–658. Springer, 2022.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3201–3219, 2024b.

Pengyu Cheng, Tianhao Hu, Han Xu, Zhisong Zhang, Yong Dai, Lei Han, nan du, and Xiaolong Li. Self-playing adversarial language game enhances LLM reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024c. URL https://openreview.net/forum?id=oCGkSH7ys2.

Yuhao Cheng, Liang Tang, Shuxian Li, Yukang Huo, Tiaonan Duan, Kaer Huang, Yanzhe Jing, and Yiqiang Yan. Evolving in tasks: Empowering the multi-modality large language model as the computer use agent, 2026a. URL https://arxiv.org/abs/2508.04037.

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, 2026b. URL https://arxiv.org/abs/2604.10923.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Meng Chu, Yicong Li, and Tat-Seng Chua. Graphvideoagent: Enhancing long-form video understanding with entity relation graphs. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, pp. 4639–4648, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3755537. URL https://doi.org/10.1145/3746027.3755537.

Anthony Costarelli, Mat Allen, Roman Hauksson, Grace Sodunke, Suhas Hariharan, Carlson Cheng, Wenjie Li, Joshua M Clymer, and Arjun Yadav. Gamebench: Evaluating strategic reasoning abilities of LLM agents. In *Language Gamification - NeurIPS 2024 Workshop*, 2024. URL https://openreview.net/forum?id=qrzKE533Jr.

Jeff Da, Clinton Wang, Xiang Deng, Yuntao Ma, Nikhil Barhate, and Sean Hendryx. Agent-rlvr: Training software engineering agents via guidance and environment rewards, 2025. URL https://arxiv.org/abs/2506.11425.

Zijie Dai, Siuhin He, Hui Li, Qihui Zhou, Jiajun Li, Mingcong Song, Guoping Long, Hongjie Si, Xin Yao, Lin Zhang, James Cheng, and Xiao Yan. Metis: Bridging text and code memory for self-evolving agents, 2026. URL https://arxiv.org/abs/2606.24151.

Kourosh Darvish, Marta Skreta, Yuchi Zhao, Naruki Yoshikawa, Sagnik Som, Miroslav Bogdanovic, Yang Cao, Han Hao, Haoping Xu, Alán Aspuru-Guzik, Animesh Garg, and Florian Shkurti. Organa: A robotic assistant for automated chemistry experimentation and characterization. *CoRR*, abs/2401.06949, 2024. URL https://doi.org/10.48550/arXiv.2401.06949.

Richard Dawkins. The evolution of evolvability. In *Artificial life*, pp. 201–220. Routledge, 2019.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3369–3391, 2022.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *CoRR*, abs/2509.16941, September 2025. URL https://doi.org/10.48550/arXiv.2509.16941.

Yang Deng, Lizi Liao, Zhonghua Zheng, Grace Hui Yang, and Tat-Seng Chua. Towards human-centered proactive conversational agents. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 807–818, 2024.

Laura Dietz, Oleg Zendel, Peter Bailey, Charles L. A. Clarke, Ellese Cotterill, Jeff Dalton, Faegheh Hasibi, Mark Sanderson, and Nick Craswell. Principles and guidelines for the use of llm judges. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, ICTIR '25, pp. 218–229, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400718618. doi: 10.1145/3731120.3744588. URL https://doi.org/10.1145/3731120.3744588.

Frederick Dillon, Gregor Halvorsen, Simon Tattershall, Magnus Rowntree, and Gareth Vanderpool. Contextual memory reweaving in large language models using layered latent state reconstruction, 2025. URL https://arxiv.org/abs/2502.02046.

Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, et al. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025a.

Zixin Ding, Junyuan Hong, Jiachen T. Wang, Zinan Lin, Zhangyang Wang, and Yuxin Chen. Scaling textual gradients via sampling-based momentum. In *Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025*, 2025b. URL https://openreview.net/forum?id=Hd8KbY52FF.

Hyo Jin Do, Zahra Ashktorab, Jasmina Gajcin, Erik Miehling, Martín Santillán Cooper, Qian Pan, Elizabeth M. Daly, and Werner Geyer. Generate, evaluate, iterate: Synthetic data for human-in-the-loop refinement of llm judges. *CoRR*, abs/2511.04478, November 2025. URL https://doi.org/10.48550/arXiv.2511.04478.

Greta Dolcetti, Vincenzo Arceri, Eleonora Iotti, Sergio Maffeis, Agostino Cortesi, and Enea Zaffanella. Helping llms improve code generation using feedback from testing and static analysis, 2025. URL https://arxiv.org/abs/2412.14841.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.16410.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-ReST: Reflection-reinforced self-training for language agents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15394–15411, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.861. URL https://aclanthology.org/2024.emnlp-main.861/.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/drouin24a.html.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=BRfqYrikdo.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z. Pan. Rethinking memory in ai: Taxonomy, operations, topics, and future directions, 2025. URL https://arxiv.org/abs/2505.00675.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. Gtbench: Uncovering the strategic reasoning limitations of llms via game-theoretic evaluations, 2024. URL https://arxiv.org/abs/2402.12348.

Benja Fallenstein and Ramana Kumar. Proof-producing reflection for hol: With an application to model polymorphism. In *International Conference on Interactive Theorem Proving*, pp. 170–186. Springer, 2015.

Benja Fallenstein and Nate Soares. Vingean reflection: Reliable reasoning for self-improving agents. *Technical Report 2015-2*, 2015.

Benja Fallenstein, Jessica Taylor, and Paul F. Christiano. Reflective oracles: A foundation for classical game theory, 2015. URL https://arxiv.org/abs/1508.04145.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems, 2025a. URL https://arxiv.org/abs/2508.07407.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. Lightmem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dyJ0GWpjJB.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. Webevolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8970–8986, 2025b.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. SeRL: Self-play reinforcement learning for large language models with limited data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c. URL https://openreview.net/forum?id=ZF93vyH9He.

Tianxiang Fei, Mingyang Song, Mao Zheng, and Xiang Yu. Memory beyond recall: A dual-process cognitive memory system for self-evolving llm agents, 2026. URL https://arxiv.org/abs/2606.09483.

Xiang Fei, Xiawu Zheng, and Hao Feng. Mcp-zero: Active tool discovery for autonomous llm agents, 2025. URL https://arxiv.org/abs/2506.01056.

Lang Feng, Fuchao Yang, Feng Chen, Xin Cheng, Haiyang Xu, Zhenglin Wan, Ming Yan, and Bo An. Agentocr: Reimagining agent history via optical self-compression, 2026a. URL https://arxiv.org/abs/2601.04786.

Tao Feng, Pengrui Han, Guanyu Lin, Ge Liu, and Jiaxuan You. Thought-retriever: Don't just retrieve raw data, retrieve thoughts for memory-augmented agentic systems, 2026b. URL https://arxiv.org/abs/2604.12231.

Yu Feng, Phu Mon Htut, Zheng Qi, Wei Xiao, Manuel Mager, Nikolaos Pappas, Kishaloy Halder, Yang Li, Yassine Benajiba, and Dan Roth. Rethinking LLM uncertainty: A multi-agent approach to estimating black-box model uncertainty. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12349–12375, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.660. URL https://aclanthology.org/2025.findings-emnlp.660/.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=9ZxnPZGmPU.

Romain Froger, Amine Benhalloum, Andrey Rusakov, Dheeraj Mekala, Emilien Garreau, Gerard Moreno-Torres Bertran, Grégoire Mialon, Hugo Laurençon, Jean-Baptiste Gaya, Kunal Malkan, Mathieu Rita, Matteo Bettini, Maxime Lecanu, Mengjue Wang, Pierre Andrews, Pierre Menard, Thomas Scialom, Ulyana Piterbarg, Virginie Do, Amar Budhiraja, Ian Yu, Mikhail Plekhanov, Ricardo Silveira Cabral, and Vladislav Vorotilov. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9gw03JpKK4.

Zihang Fu, Fanxiao Li, Jianyang Gu, Haonan Wang, Preslav Nakov, Bryan Hooi, Min-Yen Kan, and Jiaying Wu. Better with experience: Self-evolving llm agents for evidence-grounded health community notes, 2026. URL https://arxiv.org/abs/2606.02215.

Zeyu Gan and Yong Liu. Towards a theoretical understanding of synthetic data in llm post-training: A reverse-bottleneck perspective, 2025. URL https://arxiv.org/abs/2410.01720.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026a. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F. Survey Certification.

Jiaxuan Gao, Jiaao Chen, Chuyi He, Wei-Chen Wang, Shusheng Xu, Hanrui Wang, Di Jin, and Yi Wu. From self-evolving synthetic data to verifiable-reward rl: Post-training multi-turn interactive tool-using agents, 2026b. URL https://arxiv.org/abs/2601.22607.

Shuzheng Gao, Chaozheng Wang, Cuiyun Gao, Xiaoqian Jiao, Chun Yong Chong, Shan Gao, and Michael Lyu. The prompt alchemist: Automated llm-tailored prompt optimization for test case generation, 2025a. URL https://arxiv.org/abs/2501.01329.

Yifei Gao, Junhong Ye, Jiaqi Wang, and Jitao Sang. Websynthesis: World-model-guided mcts for efficient webui-trajectory synthesis, 2025b. URL https://arxiv.org/abs/2507.04370.

Spandan Garg, Benjamin Steenhoek, and Yufan Huang. Saving swe-bench: A benchmark mutation approach for realistic agent evaluation, 2026. URL https://arxiv.org/abs/2510.08996.

Scott Garrabrant, Tsvi Benson-Tilsen, Andrew Critch, Nate Soares, and Jessica Taylor. Logical induction, 2020. URL https://arxiv.org/abs/1609.03543.

Carl Friedrich Gauss. Theoria motus corporum coelestium in sectionibus conicis solem ambientium, 1809.

John Gerhart and Marc Kirschner. The theory of facilitated variation. *Proceedings of the National Academy of Sciences*, 104(suppl_1):8582–8589, 2007.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Tomasz Korbak, Henry Sleight, Rajashree Agrawal, John Hughes, Dhruv Bhandarkar Pai, Andrey Gromov, Dan Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=5B2K4LRgmz.

Alireza Ghafarollahi and Markus J. Buehler. Sciagents: Automating scientific discovery through multi-agent intelligent graph reasoning, 2024. URL https://arxiv.org/abs/2409.05556.

Seyed Kamyar Seyed Ghasemipour, Ayzaan Wahid, Jonathan Tompson, Pannag R Sanketi, and Igor Mordatch. Self-improving embodied foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KXMIIVUB9U.

Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für mathematik und physik*, 38(1):173–198, 1931.

Alexander Golubev, Maria Trofimova, Sergei Polezhaev, Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Sergey Abramov, Andrei Andriushchenko, Filipp Fisin, Sergei Skvortsov, and Boris Yangel. Training long-context, multi-turn software engineering agents with reinforcement learning, 2025. URL https://arxiv.org/abs/2508.03501.

Haisong Gong, Jing Li, Junfei Wu, Qiang Liu, and Shu Wu. Strive: Structured reasoning for self-improvement in claim verification. *Machine Intelligence Research*, 23(1):185–199, February 2026. ISSN 2731-5398. doi: 10.1007/s11633-025-1598-5. URL http://dx.doi.org/10.1007/s11633-025-1598-5.

Irving John Good. Speculations concerning the first ultraintelligent machine. In *Advances in computers*, volume 6, pp. 31–88. Elsevier, 1966.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, Yossi Matias, Andrew Carroll, Kavita Kulkarni, Nenad Tomasev, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, José R Penadés, Gary Peltz, Yunhan Xu, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Towards an ai co-scientist, 2025. URL https://arxiv.org/abs/2502.18864.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Jennifer Grannen, Michelle Pan, Kenneth Llontop, Cherie Ho, Mark Zolotas, Jeannette Bohg, and Dorsa Sadigh. Robot-powered data flywheels: Deploying robots in the wild for continual data collection and foundation model adaptation, 2025. URL https://arxiv.org/abs/2511.19647.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b_CQDy9vrD1.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=c6l7yA0HSq.

Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. Richelieu: Self-evolving llm-based agents for ai diplomacy. *Advances in Neural Information Processing Systems*, 37: 123471–123497, 2024.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZG3RaNIsO8.

Shangmin Guo, Omar Darwiche Domingues, Raphaël Avalos, Aaron Courville, and Florian Strub. Sample, predict, then proceed: Self-verification sampling for tool use of LLMs. In *Eighteenth European Workshop on Reinforcement Learning*, 2025a. URL https://openreview.net/forum?id=rw1B9Zl0OV.

Yifu Guo, Jiaye Lin, Huacan Wang, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, and Mingguang Chen. SE-agent: Self-evolution trajectory optimization in multi-step reasoning with LLM-based agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=isATAFP71B.

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

Kimia Hamidieh, Veronika Thost, Walter Gerych, Mikhail Yurochkin, and Marzyeh Ghassemi. Complementing self-consistency with cross-model disagreement for uncertainty quantification. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lOoRJo8xWy.

Jinyi Han, Zixiang Di, Zishang Jiang, Ying Liao, Jiaqing Liang, Yongqi Wang, and Yanghua Xiao. Structured reasoning for large language models, 2026a. URL https://arxiv.org/abs/2601.07180.

Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Verifiagent: a unified verification agent in language model reasoning, 2025a. URL https://arxiv.org/abs/2504.00406.

Ruiyan Han, Zhen Fang, XinYu Sun, Yuchen Ma, Ziheng Wang, Yu Zeng, Zehui Chen, Lin Chen, Wenxuan Huang, Wei-Jie Xu, Yi Cao, and Feng Zhao. Unicorn: Towards self-improving unified multimodal models through self-generated supervision, 2026b. URL https://arxiv.org/abs/2601.03193.

Yichen Han, Bojun Liu, Zhengpeng zhou, Guanyu Liu, Zeng Zhang, Yang Yang, Wenli Wang, Isaac N Shi, Yunyan, Lewei He, and TIANYU SHI. MAPGD: Multi-agent prompt gradient descent for collaborative prompt optimization. In *Workshop on Scaling Environments for Agents*, 2025b. URL https://openreview.net/forum?id=FywYwwH5z9.

Guangya Hao, Yunbo Long, and Zhuokai Zhao. Self-evolving multi-agent systems via decentralized memory, 2026. URL https://arxiv.org/abs/2605.22721.

Mohd Ariful Haque, Justin Williams, Sunzida Siddique, Md. Hujaifa Islam, Hasmot Ali, Kishor Datta Gupta, and Roy George. Advanced tool learning and selection system (atlass): A closed-loop framework using llm, 2025. URL https://arxiv.org/abs/2503.10071.

Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13504–13514, 2024a.

Han He, Qianchu Liu, Lei Xu, Chaitanya Shivade, Yi Zhang, Sundararajan Srinivasan, and Katrin Kirchhoff. Crispo: Multi-aspect critique-suggestion-guided automatic prompt optimization for text generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(22):24014–24022, Apr. 2025a. doi: 10.1609/aaai.v39i22.34575. URL https://ojs.aaai.org/index.php/AAAI/article/view/34575.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization, 2024b. URL https://arxiv.org/abs/2410.19609.

Shan He, Runze Wang, Zhuoyun Du, Huiyu Bai, Zouying Cao, Yu Cheng, and Bo Zheng. Learning to evolve: A self-improving framework for multi-agent systems via textual parameter graph optimization, 2026a. URL https://arxiv.org/abs/2604.20714.

Yanheng He, Jiahe Jin, and Pengfei Liu. Efficient agent training for computer use. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=cDuA6ZNvCl.

Zihong He, Weizhe Lin, Hao Zheng, Fan Zhang, Matt W. Jones, Laurence Aitchison, Xuhai Xu, Miao Liu, Per Ola Kristensson, and Junxiao Shen. Human-inspired perspectives: A survey on ai long-term memory, 2025b. URL https://arxiv.org/abs/2411.00489.

Prannay Hebbar, Yogendra Manawat, Samuel Verboomen, Alesia Ivanova, Selvam Palanimalai, Kunal Bhatia, and Vignesh Baskaran. Sia: Self improving ai with harness & weight updates. *arXiv preprint arXiv:2605.27276*, 2026.

Tooraj Helmi. Decentralizing ai memory: Shimi, a semantic hierarchical memory index for scalable agent reasoning, 2025. URL https://arxiv.org/abs/2504.06135.

Jesse Love Hendrikse, Trish Elizabeth Parsons, and Benedikt Hallgrímsson. Evolvability as the proper focus of evolutionary developmental biology. *Evolution & development*, 9(4): 393–401, 2007.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Vincent Herrmann and Jürgen Schmidhuber. Interestingness as an inductive heuristic for future compression progress. *arXiv preprint arXiv:2605.14831*, 2026.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International conference on artificial neural networks*, pp. 87–94. Springer, 2001.

Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

Jack Hong, Chenxiao Zhao, Weiheng Lu, ChengLIn Zhu, Guohai Xu, and XingYu. Deepeyesv2: Toward agentic multimodal model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yDKawwfJ5O.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

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

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Chenghua Huang, Zhizhen Fan, Lu Wang, Fangkai Yang, Pu Zhao, Zeqi Lin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. SELF-EVOLVED REWARD LEARNING FOR LLMS. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=Zonhl0c9I0.

Chengkai Huang, Hongtao Huang, Tong Yu, Kaige Xie, Junda Wu, Shuai Zhang, Julian Mcauley, Dietmar Jannach, and Lina Yao. A survey of foundation model-powered recommender systems: From feature-based, generative to agentic paradigms, 2025b. URL https://arxiv.org/abs/2504.16420.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, 2023.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Liangjie Huang, Dawei Li, Huan Liu, and Lu Cheng. Beyond accuracy: The role of calibration in self-improving large language models, 2025c. URL https://arxiv.org/abs/2504.02902.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. Metatool benchmark for large language models: Deciding whether to use tools and which to use, 2024b. URL https://arxiv.org/abs/2310.03128.

Marcus Hutter. A theory of universal artificial intelligence based on algorithmic complexity, 2000. URL https://arxiv.org/abs/cs/0004001.

Alex Iacob, Andrej Jovanović, William F. Shen, Daniel Burkhardt, Meghdad Kurmanji, Nurbek Tastan, Lorenzo Sani, Niccolò Alberto Elia Venanzi, Ambroise Odonnat, Zeyu Cao, Bill Marino, Xinchi Qiu, and Nicholas D. Lane. The red queen gödel machine: Co-evolving agents and their evaluators, 2026. URL https://arxiv.org/abs/2606.26294.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers, 2021. URL https://arxiv.org/abs/2106.06295.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. A modern self-referential weight matrix that learns to modify itself. In *International conference on machine learning*, pp. 9660–9677. PMLR, 2022.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. Practical computational power of linear transformers and their recurrent and self-referential extensions, 2023. URL https://arxiv.org/abs/2310.16076.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. Metalearning continual learning algorithms, 2025. URL https://arxiv.org/abs/2312.00276.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. Discoveryworld: A virtual environment for developing and evaluating automated scientific discovery agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=cDYqckEt6d.

Hongrui Jia, Jitong Liao, Xi Zhang, Haiyang Xu, Tianbao Xie, Chaoya Jiang, Ming Yan, Si Liu, Wei Ye, and Fei Huang. OSWorld-MCP: Benchmarking MCP tool invocation in computer-use agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rceD6wwt4B.

Chunyang Jiang, Chi-Min Chan, Wei Xue, Qifeng Liu, and Yike Guo. Importance weighting can help large language models self-improve. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24257–24265, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Ruofan Jin, Zaixi Zhang, Mengdi Wang, and Le Cong. Stella: Self-evolving llm agent for biomedical research, 2025. URL https://arxiv.org/abs/2507.02004.

Yuxin Jin, Siyuan Zhang, Hanchen Wang, Lu Qin, Ying Zhang, and Wenjie Zhang. Exg: Self-evolving agents with experience graphs, 2026. URL https://arxiv.org/abs/2605.17721.

Minjoon Jung, Byoung-Tak Zhang, and Lorenzo Torresani. Evoground: Self-evolving video agents for video temporal grounding, 2026. URL https://arxiv.org/abs/2605.13803.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

Seth Karten, Joel Zhang, Tersoo Upaa Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents, 2026. URL https://arxiv.org/abs/2605.09998.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Minsoo Kim and seung-won hwang. Dual-scale world models for LLM agents towards hard-exploration problems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bH5uHIVtTe.

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

Louis Kirsch and Jürgen Schmidhuber. Meta learning backpropagation and improving it. *Advances in Neural Information Processing Systems*, 34:14122–14134, 2021.

Louis Kirsch and Jürgen Schmidhuber. Eliminating meta optimization through self-referential meta learning. *arXiv preprint arXiv:2212.14392*, 2022.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=RPKxrKTJbj.

Gaurav Koley. Salm: A multi-agent framework for language model-driven social network simulation, 2025. URL https://arxiv.org/abs/2505.09081.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. ComputerRL: Scaling end-to-end online reinforcement learning for computer use agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=oEVfNf0w4B.

Jack Lanchantin, Shubham Toshniwal, Jason E Weston, Arthur Szlam, and Sainbayar Sukhbaatar. Learning to reason and memorize with self-notes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ZFwNdsDCRL.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lKEdGCoDNC.

Joshua Ong Jun Leang, Giwon Hong, Wenda Li, and Shay B Cohen. Theorem prover as a judge for synthetic data generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29941–29977. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1448. URL http://dx.doi.org/10.18653/v1/2025.acl-long.1448.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=OTmcsyEO5G.

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steven Y. Ko, Sangeun Oh, and Insik Shin. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation, 2024b. URL https://arxiv.org/abs/2312.03003.

Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes; par AM Legendre..* chez Firmin Didot, libraire pour lew mathematiques, la marine, l . . . , 1805.

Douglas B Lenat. Eurisko: a program that learns new heuristics and domain concepts: the nature of heuristics iii: program design and results. *Artificial intelligence*, 21(1-2):61–98, 1983.

Douglas B Lenat and John Seely Brown. Why am and eurisko appear to work. *Artificial intelligence*, 23(3):269–294, 1984.

Douglas B Lenat[1]. Automated theory formation in mathematics. *Automated Theorem Proving: After 25 Years: After 25 Years*, 89:287, 1984.

Ido Levy, Ben wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. ST-webagentbench: A benchmark for evaluating safety and trustworthiness in web agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=MuCDzH0ctf.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Mingqi Li, Karan Aggarwal, Yong Xie, Aitzaz Ahmad, and Stephen Lau. Learning from contrastive prompts: Automated optimization and adaptation, 2024. URL https://arxiv.org/abs/2409.15199.

Pengxiang Li, Zhi Gao, Bofei Zhang, Yapeng Mi, Xiaojian Ma, Chenrui Shi, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Iterative tool usage exploration for multimodal agents via step-wise preference tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=yKUwkihcsi.

Pengxiang Li, Zhi Gao, Bofei Zhang, Yapeng Mi, Xiaojian Ma, Chenrui Shi, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Iterative tool usage exploration for multimodal agents via step-wise preference tuning, 2025b. URL https://arxiv.org/abs/2504.21561.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. Deepagent: A general reasoning agent with scalable toolsets, 2025c. URL https://arxiv.org/abs/2510.21618.

Xinfeng Li, Dong Huang, Jie Li, Hongyi Cai, Zhenhong Zhou, Wei Dong, XiaoFeng Wang, and Yang Liu. A vision for access control in llm-based agent systems, 2025d. URL https://arxiv.org/abs/2510.11108.

Yangyang Li. Dro-instructzero: Distributionally robust prompt optimization for large language models, 2025. URL https://arxiv.org/abs/2510.15260.

Yanzhou Li, Yiran Zhang, Xiaoyu Zhang, Xiaoxia Liu, and Yang Liu. Codeskill: Learning self-evolving skills for coding agents, 2026a. URL https://arxiv.org/abs/2605.25430.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhenren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. Memos: A memory os for ai system, 2025e. URL https://arxiv.org/abs/2507.03724.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=Mf5AleTUVK.

Jiaqing Liang, Jinyi Han, Weijia Li, Xinyi Wang, Zhoujia Zhang, Zishang Jiang, Ying Liao, Tingyun Li, Ying Huang, Hao Shen, Hanyu Wu, Fang Guo, Keyi Wang, Zhonghua Hong, Zhiyu Lu, Lipeng Ma, Sihang Jiang, and Yanghua Xiao. Genericagent: A token-efficient self-evolving llm agent via contextual information density maximization (v1.0), 2026a. URL https://arxiv.org/abs/2604.17091.

Xuechen Liang, Yangfan He, Yinghui Xia, Xinyuan Song, Meiling Tao, Kuan Lu, Jianhui Wang, Li Sun, Xinhang Yuan, Keqin Li, Jiaqi Chen, TIANYU SHI, and Yang Jingsong. Self-evolving agents with reflective and memory-augmented abilities. In *LLM-based Multi-Agent Systems: Towards Responsible, Reliable, and Scalable Agentic Systems*, 2026b. URL https://openreview.net/forum?id=6Mw2fO3ejN.

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. URL https://arxiv.org/abs/2605.08374.

Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. Muse-autoskill: Self-evolving agents via skill creation, memory, management, and evaluation, 2026. URL https://arxiv.org/abs/2605.27366.

Jianghao Lin, Xinyuan Wang, Xinyi Dai, Menghui Zhu, Bo Chen, Ruiming Tang, Yong Yu, and Weinan Zhang. Masstool: A multi-task search-based tool retrieval framework for large language models, 2025. URL https://arxiv.org/abs/2507.00487.

Xiaoqiang Lin, Zhongxiang Dai, Arun Verma, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Prompt optimization with human feedback. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*, 2024. URL https://openreview.net/forum?id=344O51eTPN.

Tobias Lindenbauer, Georg Groh, and Hinrich Schuetze. From knowledge to noise: CTIM-rover and the pitfalls of episodic memory in software engineering agents. In Ehsan Kamalloo, Nicolas Gontier, Xing Han Lu, Nouha Dziri, Shikhar Murty, and Alexandre Lacoste (eds.), *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 411–427, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-264-0. doi: 10.18653/v1/2025.realm-1.30. URL https://aclanthology.org/2025.realm-1.30/.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

Bo Liu, Simon Yu, Zichen Liu, Leon Guertler, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. SPIRAL: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=7Yayy5fNLg.

Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. Webcoach: Self-evolving web agents with cross-session memory guidance, 2025a. URL https://arxiv.org/abs/2511.12997.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. Chain of hindsight aligns language models with feedback. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=6xfe4IVcOu.

Jiang Liu, Bolin Li, Haoyuan Li, Tianwei Lin, Wenqiao Zhang, Tao Zhong, Zhelun Yu, Jinghao Wei, Hao Cheng, Wanggui He, Fangxun Shu, Hao Jiang, Zheqi Lv, Juncheng Li, Siliang Tang, and Yueting Zhuang. Boosting private domain understanding of efficient mllms: A tuning-free, adaptive, universal prompt optimization framework, 2025b. URL https://arxiv.org/abs/2412.19684.

Jiaqi Liu, Xinyu Ye, Peng Xia, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Evolvemem:self-evolving memory architecture via autoresearch for llm agents, 2026b. URL https://arxiv.org/abs/2605.13941.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in latent space via differentiable cache augmentation. In *Forty-second International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=IaUJl5RCOu.

Shunyu Liu, Yaoru Li, Kongcheng Zhang, Zhenyu Cui, Wenkai Fang, Yuxuan Zheng, Tongya Zheng, and Mingli Song. Odyssey: Empowering minecraft agents with open-world skills, 2025d. URL https://arxiv.org/abs/2407.15325.

Tennison Liu and Mihaela van der Schaar. Position: Truly self-improving agents require intrinsic metacognitive learning. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. URL https://openreview.net/forum?id=4KhDd0Ozqe.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations*, volume 2024, pp. 56174–56194, 2024b.

Xukun Liu, Zhiyuan Peng, Xiaoyuan Yi, Xing Xie, Lirong Xiang, Yuchen Liu, and Dongkuan Xu. Toolnet: Connecting large language models with massive tools via tool graph, 2024c. URL https://arxiv.org/abs/2403.00839.

Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. Tool-planner: Task planning with clusters across multiple tools. In *The Thirteenth International Conference on Learning Representations*, 2025e. URL https://openreview.net/forum?id=dRz3cizftU.

Zewen Liu, Zhan Shi, Yisi Sang, Bing He, Minhua Lin, Tianxin Wei, Dakuo Wang, Benoit Dumoulin, Wei Jin, and Hanqing Lu. Adaptive auto-harness: Sustained self-improvement for agentic system deployment on open-ended task streams, 2026c. URL https://arxiv.org/abs/2606.01770.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=PMz29A7Muq.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024a. URL https://arxiv.org/abs/2408.06292.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, Xin Jiang, and Qun Liu. Self: Self-evolution with language feedback, 2024b. URL https://arxiv.org/abs/2310.00533.

Xing Han Lu, Zdeněk Kasner, and Siva Reddy. WebLINX: Real-world website navigation with multi-turn dialogue. In *Forty-first International Conference on Machine Learning*, 2024c. URL https://openreview.net/forum?id=mUSPhG4uDW.

Yifu Lu, Shengjie Liu, and Li Dong. OrchDAG: Complex tool orchestration in multi-turn interactions with plan DAGs. In *First Workshop on Multi-Turn Interactions in Large Language Models*, 2025. URL https://openreview.net/forum?id=uZE8mTYvHE.

Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A. Burke. Memtool: Optimizing short-term memory management for dynamic tool calling in llm agent multi-turn conversations, 2025a. URL https://arxiv.org/abs/2507.21428.

Elias Lumer, Faheem Nizar, Anmol Gulati, Pradeep Honaganahalli Basavaraju, and Vamse Kumar Subbiah. Tool-to-agent retrieval: Bridging tools and agents for scalable llm multi-agent systems, 2025b. URL https://arxiv.org/abs/2511.01854.

Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. From correction to mastery: Reinforced distillation of large language model agents, 2026. URL https://openreview.net/forum?id=n4Er2o4BFB.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024.

Xinbei Ma, Congmin Zheng, Jiyang Qiu, Jiale Hong, Yao Yao, Xiangmou Qu, Jiaxin Yin, Xingyu Lou, Jun Wang, Weiwen Liu, et al. Retrospective progress-aware self-refinement for llm agent training. *arXiv preprint arXiv:2606.14302*, 2026a.

Zihan Ma, Zhikai Zhao, Chuanbo Hua, Federico Berto, and Jinkyoo Park. Judgeflow: Agentic workflow optimization via block judge, 2026b. URL https://arxiv.org/abs/2601.07477.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, 1994.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=fgFBtYgJQX_.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Suyash Mishra. Prism: An evolutionary memory substrate for multi-agent open-ended discovery, 2026. URL https://arxiv.org/abs/2604.19795.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schuetze. RET-LLM: Towards a general read-write memory for large language models. In *ICLR 2024 Workshop: How Far Are We From AGI*, 2024. URL https://openreview.net/forum?id=Z7tBs47cSH.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4, 2023. URL https://arxiv.org/abs/2306.02707.

Niels Mündler, Mark Niklas Mueller, Jingxuan He, and Martin Vechev. SWT-bench: Testing and validating real-world bug-fixes with code agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=9Y8zUO11EQ.

John Myhill. The abstract theory of self-reproduction. *Views on general systems theory*, pp. 106–118, 1964.

Anirudh Nair, Adi Banerjee, Laurent Mombaerts, Matthew Hagen, and Tarik Borogovac. Tournament of prompts: Evolving LLM instructions through structured debates and elo ratings. In *First International KDD Workshop on Prompt Optimization, 2025*, 2025. URL https://openreview.net/forum?id=Z9OsLgBCDG.

Li Nanbo, Firas Laakom, Yucheng Xu, Wenyi Wang, and Jürgen Schmidhuber. Facts: A factored state-space framework for world modelling. In *International Conference on Learning Representations*, volume 2025, pp. 68955–68983, 2025.

Michael Nguyen, Quoc Nguyen, and Paul Vuong. Recursive self-evolving agents via held-out selection, 2026. URL https://arxiv.org/abs/2606.28374.

Ziyi Ni, Huacan Wang, Shuo Zhang, Shuo Lu, Ziyang He, Wang You, Zhenheng Tang, Yuntao Du, Bill Sun, Hongzhang Liu, Sen Hu, Ronghao Chen, Bo Li, Xin Li, Chen Hu, Binxing Jiao, Daxin Jiang, and Pin Lyu. Gittaskbench: A benchmark for code agents solving real-world tasks through code repository leveraging, 2025. URL https://arxiv.org/abs/2508.18993.

Kulin Nikita, Sitkina Alena, Khairullin Artur, Zhuravlev Viktor, and Sergey Muravyov. Coolprompt: An automatic prompt optimization framework for large language models, 2026. URL https://openreview.net/forum?id=XGECnjDEcS.

Eric Nivel, Kristinn R Thórisson, Bas R Steunebrink, Haris Dindo, Giovanni Pezzulo, Manuel Rodriguez, Carlos Hernández, Dimitri Ognibene, Jürgen Schmidhuber, Ricardo Sanz, et al. Bounded recursive self-improvement. *arXiv preprint arXiv:1312.6764*, 2013.

Kolby Nottingham, Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Sameer Singh, Peter Clark, and Roy Fox. Skill set optimization: Reinforcing language model behavior via transferable skills. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=9laB7ytoMp.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019. URL https://arxiv.org/abs/1912.06680.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Laurent Orseau and Mark Ring. Self-modification and mortality in artificial agents. In *International Conference on Artificial General Intelligence*, pp. 1–10. Springer, 2011.

Yixin Ou, Wangchunshu Zhou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, et al. Symbolic learning enables self-evolving agents. *AI Open*, 2025.

Chaoqian Ouyang, Ling Yue, Shimin Di, Libin Zheng, Linan Yue, Shaowu Pan, Jian Yin, and Min-Ling Zhang. Code2mcp: Transforming code repositories into mcp services, 2025a. URL https://arxiv.org/abs/2509.05941.

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.

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, 2025b. URL https://arxiv.org/abs/2509.25140.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-gym. In Aarti Singh,

Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. Webcanvas: Benchmarking web agents in online environments. In *Agentic Markets Workshop at ICML 2024*, 2024. URL https://openreview.net/forum?id=O1FaGasJob.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701320. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Junyeong Park, Junmo Cho, and Sungjin Ahn. Mrsteve: Instruction-following agents in minecraft with what-where-when memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjXaMI2kUH.

Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. Large language models can self-improve at web agent tasks, 2024. URL https://arxiv.org/abs/2405.20309.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. Check your facts and try again: Improving large language models with external knowledge and automated feedback, 2023. URL https://arxiv.org/abs/2302.12813.

Nenad Petrovic, Vahid Zolfaghari, Andre Schamschurko, Sven Kirchner, Fengjunjie Pan, Chengdong Wu, Nils Purschke, Aleksei Velsh, Krzysztof Lebioda, Yinglei Song, Yi Zhang, Lukasz Mazur, and Alois Knoll. Survey of genai for automotive software development: From requirements to executable code. In *2025 2nd International Generative AI and Computational Language Modelling Conference (GACLM)*, pp. 184–197, 2025. doi: 10.1109/GACLM67198.2025.11232000.

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. Self-improving language models for evolutionary program synthesis: A case study on ARC-AGI. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=z4IG090qt2.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason E Weston, and Jane Yu. Self-consistency preference optimization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 49737–49751. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/prasad25a.html.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods*

in Natural Language Processing, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents, 2024. URL https://arxiv.org/abs/2408.07199.

Ju Qi, Xiaoxi Mao, Jianfeng Wang, and Na Mou. LLM coaching LLM in self-play training, 2026. URL https://openreview.net/forum?id=NnEfjLA50a.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 6922–6939, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.462. URL https://aclanthology.org/2023.findings-emnlp.462/.

Hongjin Qian and Zheng Liu. Metaagent: Toward self-evolving agent via tool meta-learning, 2025. URL https://arxiv.org/abs/2508.00271.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871, 2024a.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Jiang, Chengfei Lv, and Huajun Chen. AutoAct: Automatic agent learning from scratch for QA via self-planning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3003–3021, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.165. URL https://aclanthology.org/2024.acl-long.165/.

Yiwei Qin, Yixiu Liu, and Pengfei Liu. Dive: Diversified iterative self-improvement. *arXiv preprint arXiv:2501.00747*, 2025.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, Shilong Liu, Xun Jiang, Liu Leqi, and Mengdi Wang. Agentdistill: Training-free agent distillation with generalizable mcp boxes, 2025a. URL https://arxiv.org/abs/2506.14728.

Jiahao Qiu, Xuan Qi, Hongru Wang, Xinzhe Juan, Yimin Wang, Zelin Zhao, Jiayi Geng, Jiacheng Guo, Peihang Li, Jingzhe Shi, Shilong Liu, and Mengdi Wang. Alita-g: Self-evolving generative agent for agent generation, 2025b. URL https://arxiv.org/abs/2510.23601.

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, 2025c. URL https://arxiv.org/abs/2505.20286.

Jielin Qiu, Zuxin Liu, Zhiwei Liu, Rithesh Murthy, Jianguo Zhang, Haolin Chen, Shiyu Wang, Ming Zhu, Liangwei Yang, Juntao Tan, Roshan Ram, Akshara Prabhakar, Tulika Awalgaonkar, Zixiang Chen, Zhepeng Cen, Cheng Qian, Shelby Heinecke, Weiran Yao, Silvio Savarese, Caiming Xiong, and Huan Wang. Locobench-agent: An interactive benchmark for llm agents in long-context software engineering, 2025d. URL https://arxiv.org/abs/2511.13998.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. From exploration to mastery: Enabling LLMs to master tools via self-driven interactions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=QKBu1BOAwd.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DRC9pZwBwR.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Prithvi Rajasekaran. Harness design for long-running application development. https://www.anthropic.com/engineering/harness-design-long-running-apps, March 2026. Anthropic Engineering Blog.

Aditya A Ramesh, Alex Lewandowski, and Jürgen Schmidhuber. Learning to forget: Continual learning with adaptive weight decay. *arXiv preprint arXiv:2604.27063*, 2026.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

Ravin Ravi, Dylan Bradshaw, Stefano Ruberto, Gunel Jahangirova, and Valerio Terragni. Llmloop: Improving llm-generated code and tests through automated iterative feedback loops. In *2025 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 930–934, 2025. doi: 10.1109/ICSME64153.2025.00109.

Chidaksh Ravuru, Sagar Srinivas Sakhinana, and Venkataramana Runkana. Agentic retrieval-augmented generation for time series analysis, 2024. URL https://arxiv.org/abs/2408.14484.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

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, 2026a. URL https://arxiv.org/abs/2606.04536.

Zhe Ren, Yibo Yang, Yimeng Chen, Zijun Zhao, Benshuo Fu, Zhihao Shu, Bingjie Zhang, Yangyang Xu, Dandan Guo, and Shuicheng Yan. Gatemem: Benchmarking memory governance in multi-principal shared-memory agents, 2026b. URL https://arxiv.org/abs/2606.18829.

Jonathan Richens, Tom Everitt, and David Abel. General agents need world models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=dlIoumNiXt.

Luke Rivard, Sun Sun, Hongyu Guo, Wenhu Chen, and Yuntian Deng. Neuralos: Towards simulating operating systems via neural generative models, 2026. URL https://arxiv.org/abs/2507.08800.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent, 2025. URL https://arxiv.org/abs/2504.15228.

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

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=GEcwtMk1uA.

Yixiang Ruan, Chenyin Lu, Ning Xu, Yuchen He, Yixin Chen, Jian Zhang, Jun Xuan, Jianzhang Pan, Qun Fang, Hanyu Gao, et al. An automatic end-to-end chemical synthesis development platform powered by large language models. *Nature communications*, 15(1): 10160, 2024b.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. MemInsight: Autonomous memory augmentation for LLM agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33136–33152, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1683. URL https://aclanthology.org/2025.emnlp-main.1683/.

Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

Jitao Sang, Jinlin Xiao, Jiarun Han, Jilin Chen, Xiaoyi Chen, Shuyu Wei, Yongjie Sun, and Yuhang Wang. Beyond pipelines: A survey of the paradigm shift toward model-native agentic ai, 2025. URL https://arxiv.org/abs/2510.16720.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

J. Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. Technical Report FKI-147-91, Institut für Informatik, Technische Universität München, March 1991a.

Juergen Schmidhuber. On learning how to learn learning strategies. 1994. URL https://api.semanticscholar.org/CorpusID:59806870.

Juergen Schmidhuber. A general method for incremental self-improvement and multi-agent learning. In *Evolutionary Computation: Theory and Applications*, pp. 81–123. World Scientific, 1999.

Juergen Schmidhuber. Exploring the predictable. In *Advances in evolutionary computing: theory and applications*, pp. 579–612. Springer, 2003a.

Juergen Schmidhuber. Goedel machines: Self-referential universal problem solvers making provably optimal self-improvements, 2006a. URL https://arxiv.org/abs/cs/0309048.

Juergen Schmidhuber. Annotated history of modern ai and deep learning. *arXiv preprint arXiv:2212.11279*, 2022.

Juergen Schmidhuber, Jieyu Zhao, and MA Wiering. Simple principles of metalearning. *Technical report IDSIA*, 69:1–23, 1996.

Jürgen Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook. Diploma thesis, Institut für Informatik, Technische Universität München, 1987.

Jürgen Schmidhuber. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. *Institut für Informatik, Technische Universität München. Technical Report FKI-126*, 90, 1990.

Jürgen Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pp. 1458–1463, 1991b.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991c.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Jürgen Schmidhuber. A 'self-referential'weight matrix. In *International conference on artificial neural networks*, pp. 446–450. Springer, 1993.

Jürgen Schmidhuber. Beyond\genetic programming": Incremental self-improvement. In *Proc. Workshop on Genetic Programming at ML95*, pp. 42–49, 1995.

Jürgen Schmidhuber. What"s interesting?, 1997.

Jürgen Schmidhuber. Gödel machines: self-referential universal problem solvers making provably optimal self-improvements. *arXiv preprint cs/0309048*, 2003b.

Jürgen Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.

Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006b.

Jürgen Schmidhuber. Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity. In *International conference on discovery science*, pp. 26–38. Springer, 2007.

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE transactions on autonomous mental development*, 2(3):230–247, 2010.

Jürgen Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4: 313, 2013.

Jürgen Schmidhuber. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *arXiv preprint arXiv:1511.09249*, 2015.

Jürgen Schmidhuber and Jieyu Zhao. Multi-agent learning with the success-story algorithm. In *Workshop on Learning in Distributed Artificial Intelligence Systems*, pp. 82–93. Springer, 1996.

Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.

Jürgen Schmidhuber, Jieyu Zhao, and Nicol N Schraudolph. Reinforcement learning with self-modifying policies. In *Learning to learn*, pp. 293–309. Springer, 1998.

Thomas Schmied, Jörg Bornschein, Jordi Grau-Moya, Markus Wulfmeier, and Razvan Pascanu. Llms are greedy agents: Effects of rl fine-tuning on decision-making abilities. *arXiv preprint arXiv:2504.16078*, 2025.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

David Shapiro, Wangfan Li, Manuel Delaflor, and Carlos Toxtli. Conceptual framework for autonomous cognitive entities. *arXiv preprint arXiv:2310.06775*, 2023.

Archit Sharma, Ahmed M Ahmed, Rehaan Ahmad, and Chelsea Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. In *Conference on Robot Learning*, pp. 3292–3308. PMLR, 2023.

Buxin She, Brian Chen, Luanzheng Guo, and Fangxing Li. Pfagent: A tractable and self-evolving power-flow agent for interactive grid analysis, 2026. URL https://arxiv.org/abs/2604.10846.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. *Advances in Neural Information Processing Systems*, 37:4540–4574, 2024.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, et al. Taskcraft: Automated generation of agentic tasks. *arXiv preprint arXiv:2506.10055*, 2025a.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*, 2025b.

Zifeng Shi, Meiqin Liu, Senlin Zhang, Ronghao Zheng, Shanling Dong, and Ping Wei. Gawm: Global-aware world model for multi-agent reinforcement learning, 2025c. URL https://arxiv.org/abs/2501.10116.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Yoav Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1):51–92, 1993.

KaShun Shum, Binyuan Hui, Jiawei Chen, Lei Zhang, X. W., Jiaxi Yang, Yuzhen Huang, Junyang Lin, and Junxian He. Swe-rm: Execution-free feedback for software engineering agents, 2025. URL https://arxiv.org/abs/2512.21919.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget, 2024a. URL https://arxiv.org/abs/2305.17493.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022): 755–759, 2024b.

Zachary S Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. CORE-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=BsMMc4MEGS.

David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1, 2025.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017a. URL https://arxiv.org/abs/1712.01815.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017b.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Toby Simonds and Akira Yoshiyama. LADDER: Self-improving LLMs through recursive problem decomposition, 2025.

Toby Simonds, Kevin Lopez, Akira Yoshiyama, and Dominique Garmier. Self rewarding self improving. *arXiv preprint arXiv:2505.08827*, 2025.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag, 2025a. URL https://arxiv.org/abs/2501.09136.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=lNAyUngGFK. Expert Certification.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning, 2025b. URL https://arxiv.org/abs/2505.01441.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18221–18232, 2024a.

Yaoxian Song, Penglei Sun, Haoyu Liu, Zhixu Li, Wei Song, Yanghua Xiao, and Xiaofang Zhou. Scene-driven multimodal knowledge graph construction for embodied ai. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6962–6976, 2024b. doi: 10.1109/TKDE.2024.3399746.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, 2024c.

Aleksandar Stanić, Yujin Tang, David Ha, and Jürgen Schmidhuber. Learning to generalize with object-centric agents in the open world survival game crafter. *IEEE Transactions on Games*, 16(2):384–395, 2023.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate ai research, 2025. URL https://arxiv.org/abs/2504.01848.

Bas R Steunebrink and JÃ1/4rgen Schmidhuber. Towards an actual gödel machine implementation: A lesson in self-reflective systems. In *Theoretical Foundations of Artificial General Intelligence*, pp. 173–195. Springer, 2012.

Jan Storck, Sepp Hochreiter, Jürgen Schmidhuber, et al. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the international conference on artificial neural networks, Paris*, volume 2, pp. 159–164, 1995.

Janani Subramanian. Science fiction and philosophy: From time travel to superintelligence, 2010.

Haoran Sun and Shaoning Zeng. Hierarchical memory for high-efficiency long-term reasoning in llm agents, 2025. URL https://arxiv.org/abs/2507.22925.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022.

Yuchang Sun, Yanxi Chen, Yaliang Li, and Bolin Ding. Enhancing latent computation in transformers with latent tokens, 2025a. URL https://arxiv.org/abs/2505.12629.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience, 2025b. URL https://arxiv.org/abs/2508.04700.

Balachandra Devarangadi Sunil, Isheeta Sinha, Piyush Maheshwari, Shantanu Todmal, Shreyan Mallik, and Shuchi Mishra. Memory poisoning attack and defense on memory based llm-agents, 2026a. URL https://arxiv.org/abs/2601.05504.

Meghana Sunil, Manikandarajan Venmathimaran, and Muthu Subash Kavitha. ireasoner: Trajectory-aware intrinsic reasoning supervision for self-evolving large multimodal models, 2026b. URL https://arxiv.org/abs/2601.05877.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory, 2025. URL https://arxiv.org/abs/2504.07952.

Hexiang Tan, Fei Sun, Sha Liu, Du Su, Qi Cao, Xin Chen, Jingang Wang, Xunliang Cai, Yuanzhuo Wang, Huawei Shen, and Xueqi Cheng. Too consistent to detect: A study of self-consistent errors in LLMs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4755–4765, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.238. URL https://aclanthology.org/2025.emnlp-main.238/.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.413. URL https://aclanthology.org/2025.acl-long.413/.

Tao Tao, Guanghui Zhu, Lang Guo, Hongyi Chen, Chunfeng Yuan, and Yihua Huang. Delvepo: Direction-guided self-evolving framework for flexible prompt optimization, 2025. URL https://arxiv.org/abs/2510.18257.

Wangcheng Tao, Han Wu, and Weng-Fai Wong. Sepo: Self-evolving prompt agent for system prompt optimization, 2026a. URL https://arxiv.org/abs/2606.04465.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models, 2024. URL https://arxiv.org/abs/2404.14387.

Zhengwei Tao, Bo Li, Jialong Wu, Guochen Yan, Huanyao Zhang, Jiahao Xu, Haitao Mi, and Wentao Zhang. Ragshaper: Eliciting sophisticated agentic rag skills via automated data synthesis, 2026b. URL https://arxiv.org/abs/2601.08699.

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pp. 3–17. Springer, 1998.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. Toward self-improvement of LLMs via imagination, searching, and criticizing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tPdJ2qHkOB.

Demian Till, John Smeaton, Peter Haubrick, Gouse Saheb, Florian Graef, and David Berman. Teaming llms to detect and mitigate hallucinations, 2025. URL https://arxiv.org/abs/2510.19507.

Alexander V Tobias and Adam Wahab. Autonomous 'self-driving'laboratories: a review of technology and policy implications. *Royal Society Open Science*, 12(7):250646, 2025.

Hieu Tran, Zonghai Yao, Nguyen Luong Tran, Zhichao Yang, Feiyun Ouyang, Shuo Han, Razieh Rahimi, and Hong Yu. Prime: Planning and retrieval-integrated memory for enhanced reasoning, 2025. URL https://arxiv.org/abs/2509.22315.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Vivek Trivedy. The anatomy of an agent harness. https://www.langchain.com/blog/the-anatomy-of-an-agent-harness, March 2026. LangChain Blog.

A. M. Turing. Intelligent machinery. Technical report, National Physical Laboratory, 1948. URL https://dn790006.ca.archive.org/0/items/turing1948/turing1948_text.pdf. Unpublished report (typescript). Original held at King's College, Cambridge (Turing Digital Archive, AMT/C/11).

Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950. doi: 10.1093/mind/LIX.236.433.

John Von Neumann, Arthur Walter Burks, et al. *Theory of self-reproducing automata*. University of Illinois press Urbana, 1966.

Manya Wadhwa, Zayne Sprague, Chaitanya Malaviya, Philippe Laban, Junyi Jessy Li, and Greg Durrett. Evalagent: Discovering implicit evaluation criteria from the web, 2025. URL https://arxiv.org/abs/2504.15219.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. Scm: Enhancing large language model with self-controlled memory framework, 2025a. URL https://arxiv.org/abs/2304.13343.

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.

Hongru Wang, Cheng Qian, Manling Li, Jiahao Qiu, Boyang Xue, Mengdi Wang, Heng Ji, and Kam-Fai Wong. Toward a theory of agents as tool-use decision-makers, 2025b. URL https://arxiv.org/abs/2506.00886.

Jiahao Wang, Luoxin Ye, TaiMing Lu, Junfei Xiao, Jiahan Zhang, Yuxiang Guo, Xijun Liu, Rama Chellappa, Cheng Peng, Alan Yuille, and Jieneng Chen. Evoworld: Evolving panoramic world generation with explicit 3d memory, 2025c. URL https://arxiv.org/abs/2510.01183.

Junlin Wang, Roy Xie, Shang Zhu, Jue Wang, Ben Athiwaratkun, Bhuwan Dhingra, Shuaiwen Leon Song, Ce Zhang, and James Zou. Improving model alignment through collective intelligence of open-source llms, 2025d. URL https://arxiv.org/abs/2505.03059.

Juntong Wang, Haoyue Zhao, guanghui Pan, Xiyuan Wang, Yanbo Wang, Qiyan Deng, and Muhan Zhang. Sage: A self-evolving agentic graph-memory engine for structure-aware associative memory, 2026a. URL https://arxiv.org/abs/2605.12061.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024a. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL http://dx.doi.org/10.1007/s11704-024-40231-1.

Nengbo Wang, Xiaotian Han, Jagdip Singh, Jing Ma, and Vipin Chaudhary. CausalRAG: Integrating causal graphs into retrieval-augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22680–22693, Vienna, Austria, July 2025e. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1165. URL https://aclanthology.org/2025.findings-acl.1165/.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*, 2025f. URL https://openreview.net/forum?id=XLMAMmowdY.

Wenhao Wang, Peizhi Niu, Zhao Xu, Zhaoyu Chen, Jian Du, Yaxin Du, Xianghe Pang, Keduan Huang, Yanfeng Wang, Qiang Yan, and Siheng Chen. Mcp-flow: Facilitating llm agents to master real-world, diverse and scaling mcp tools, 2025g. URL https://arxiv.org/abs/2510.24284.

Wenyi Wang, Hisham A. Alyahya, Dylan R. Ashley, Oleg Serikov, Dmitrii Khizbullin, Francesco Faccio, and Jürgen Schmidhuber. How to correctly do semantic backpropagation on language-based agentic systems, 2024b. URL https://arxiv.org/abs/2412.03624.

Wenyi Wang, Piotr Piękos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-g\"odel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=T0EiEuhOOL.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations*, 2024c. URL https://openreview.net/forum?id=jp3gWrMuIZ.

Yimeng Wang, Jiaxing Zhao, Hongbin Xie, Hexing Ma, Yuzhen Lei, Shuangxue Liu, Xuan Song, Zichen Zhang, and Haoran Zhang. Metagen: Self-evolving roles and topologies for multi-agent llm reasoning, 2026c. URL https://arxiv.org/abs/2601.19290.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents, 2025. URL https://arxiv.org/abs/2507.07957.

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, 2024d. URL https://arxiv.org/abs/2402.04624.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian Mcauley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending MemoryLLM with scalable long-term memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63308–63323, 13–19 Jul 2025h. URL https://proceedings.mlr.press/v267/wang25au.html.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: towards unleashing infinite data for automated robot learning via generative simulation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024e.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025i. URL https://arxiv.org/abs/2504.20073.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911, 13–19 Jul 2025j. URL https://proceedings.mlr.press/v267/wang25bx.html.

Anjiang Wei, Tarun Suresh, Jiannan Cao, Naveen Kannan, Yuheng Wu, Kai Yan, Thiago S. F. X. Teixeira, Ke Wang, and Alex Aiken. CodeARC: Benchmarking reasoning capabilities of LLM agents for inductive program synthesis. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=Q5pVZCrrKr.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Lai Wei, Yuting Li, Chen Wang, Yue Wang, Linghe Kong, Weiran Huang, and Lichao Sun. Unsupervised post-training for multi-modal llm reasoning via grpo. *arXiv preprint arXiv:2505.22453*, 2025b.

Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. Autotir: Autonomous tools integrated reasoning via reinforcement learning, 2025c. URL https://arxiv.org/abs/2507.21836.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, LINGMING ZHANG, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. SWE-RL: Advancing LLM reasoning via reinforcement learning on open software evolution. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025d. URL https://openreview.net/forum?id=ULblO61XZ0.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7920–7939, 2025e.

Norbert Wiener. *Cybernetics: Or Control and Communication in the Animal and the Machine*. John Wiley & Sons, New York, 1948. UK edition commonly listed as Chapman & Hall, London.

Marco A. Wiering and Jürgen Schmidhuber. Solving pomdps with levin search and EIRA. In Lorenza Saitta (ed.), *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pp. 534–542. Morgan Kaufmann, 1996.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26092–26130, 2025.

Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

Penghao Wu, Shengnan Ma, Bo Wang, Jiaheng Yu, Lewei Lu, and Ziwei Liu. GUI-reflection: Empowering multimodal GUI models with self-reflection behavior. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=qup6v4WnYX.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37:25981–26010, 2024a.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11537–11554, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.583. URL https://aclanthology.org/2025.emnlp-main.583/.

Ting Wu, Xuefeng Li, and Pengfei Liu. Progress or regress? self-improvement reversal in post-training. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=RFqeoVfLHa.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. From human memory to ai memory: A survey on memory mechanisms in the era of llms, 2025d. URL https://arxiv.org/abs/2504.15965.

Yaxiong Wu, Yongyue Zhang, Sheng Liang, and Yong Liu. Sgmem: Sentence graph memory for long-term conversational agents, 2025e. URL https://arxiv.org/abs/2509.21212.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement, 2024b. URL https://arxiv.org/abs/2402.07456.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents, 2024. URL https://arxiv.org/abs/2407.01489.

Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. Live-swe-agent: Can software engineering agents self-evolve on the fly?, 2025. URL https://arxiv.org/abs/2511.13646.

Chuan Xiao, Zhengbo Jiao, Shaobo Wang, Wei Wang, Bing Zhao, Hu Wei, Linfeng Zhang, and Lin Qu. Socratic-swe: Self-evolving coding agents via trace-derived agent skills, 2026. URL https://arxiv.org/abs/2606.07412.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Aojun Zhou, and Hongsheng Li. UI-genie: A self-improving approach for iteratively boosting MLLM-based mobile GUI agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=3uUmJzSSOW.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Ruibin Xiong, Yimeng Chen, Dmitrii Khizbullin, Mingchen Zhuge, and Jürgen Schmidhuber. Beyond outlining: Heterogeneous recursive planning for adaptive long-form writing with language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 24689–24725, 2025.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=CfXh93NDgH.

Fangyuan Xu, Rujun Han, Yanfei Chen, Zifeng Wang, I-Hung Hsu, Jun Yan, Vishy Tirumalashetty, Eunsol Choi, Tomas Pfister, and Chen-Yu Lee. SAGE: Steerable agentic data generation for deep search with execution feedback. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 3334–3351, Rabat, Morocco, March 2026a. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.174. URL https://aclanthology.org/2026.findings-eacl.174/.

Guowei Xu, Mert Yuksekgonul, Carlos Guestrin, and James Zou. metatextgrad: Automatically optimizing language model optimizers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=10s01YrlKp.

Haoran Xu, Jiacong Hu, Ke Zhang, Lei Yu, Yuxin Tang, Xinyuan Song, Yiqun Duan, Lynn Ai, and Bill Shi. Sedm: Scalable self-evolving distributed memory for agents. *arXiv preprint arXiv:2509.09498*, 2025b.

Haoran Xu, Jiacong Hu, Ke Zhang, Lei Yu, Yuxin Tang, Xinyuan Song, Yiqun Duan, Lynn Ai, and Bill Shi. Sedm: Scalable self-evolving distributed memory for agents, 2025c. URL https://arxiv.org/abs/2509.09498.

Kaishuai Xu, Tiezheng Yu, Yi Cheng, Wenjun Hou, Liangyou Li, Xin Jiang, Lifeng Shang, Qun Liu, and Wenjie Li. Learning to align multi-faceted evaluation: A unified and robust framework. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9488–9502, Vienna, Austria, July 2025d. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.494. URL https://aclanthology.org/2025.findings-acl.494/.

Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. DipLLM: Fine-tuning LLM for strategic decision-making in diplomacy. In *Forty-second International Conference on Machine Learning*, 2025e. URL https://openreview.net/forum?id=hfPaOxDWfI.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents, 2025f. URL https://arxiv.org/abs/2502.12110.

Wujiang Xu, Jiaojiao Han, Minghao Guo, Kai Mei, Xi Zhu, Han Zhang, and Dimitris N. Metaxas. Ael: Agent evolving learning for open-ended environments, 2026b. URL https://arxiv.org/abs/2604.21725.

Yifan Xu, Yixuan Li, Xinzhuo Li, Lijun Yu, and Haohan Wang. Pick your textual gradients, 2025g. URL https://openreview.net/forum?id=ydTwv5D536.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. Language agents with reinforcement learning for strategic play in the werewolf game. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=usUPvQH3XK.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066.

Roman V Yampolskiy. From seed ai to technological singularity via recursively self-improving software. *arXiv preprint arXiv:1502.06512*, 2015.

Zhiling Yan, Dingjie Song, Hanrong Zhang, Wei Liang, Yuxuan Zhang, Yutong Dai, Lifang He, Philip S. Yu, Ran Xu, Xiang Li, and Lichao Sun. Openskill: Open-world self-evolution for llm agents, 2026. URL https://arxiv.org/abs/2606.06741.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

Chengyi Yang, Zhishang Xiang, Yunbo Tang, Zongpei Teng, Chengsong Huang, Fei Long, Yuhan Liu, and Jinsong Su. Ttcs: Test-time curriculum synthesis for self-evolving, 2026a. URL https://arxiv.org/abs/2601.22628.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, Ping Luo, Xiangyu Yue, and Hongyang Li. Rise: Self-improving robot policy with compositional world model, 2026b. URL https://arxiv.org/abs/2602.11075.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, Diyi Yang, Sida Wang, and Ofir Press. SWE-bench multimodal: Do AI systems generalize to visual software domains? In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=riTiq3i21b.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=DgGF2LEBPS.

Shiyi Yang, Zhibo Hu, Xinshu Li, Chen Wang, Tong Yu, Xiwei Xu, Liming Zhu, and Lina Yao. Drunkagent: Stealthy memory corruption in llm-powered recommender agents, 2025c. URL https://arxiv.org/abs/2503.23804.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills, 2026c. URL https://arxiv.org/abs/2605.23904.

Yunhao Yang, Neel P. Bhatt, Kevin Wang, Samuel Tetteh, Zhangyang Wang, and Ufuk Topcu. Vaso: Formally verifiable self-evolving skills for physical ai agents, 2026d. URL https://arxiv.org/abs/2606.05395.

Zherui Yang, Fan Liu, Yansong Ning, and Hao Liu. Evods: Self-evolving autonomous data science agent with skill learning and context management. *arXiv preprint arXiv:2606.03841*, 2026e.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Xuan Yao, Junyu Gao, and Changsheng Xu. Navmorph: A self-evolving world model for vision-and-language navigation in continuous environments, 2025. URL https://arxiv.org/abs/2506.23468.

Bingji Yi, Qiyuan Liu, Yuwei Cheng, and Haifeng Xu. Escaping model collapse via synthetic data verification: Near-term improvements and long-term convergence. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yfk6c39omW.

Sheng Yin, Xianghe Pang, Yuanzhuo Ding, Menglan Chen, Yutong Bi, Yichen Xiong, Wenhao Huang, Zhen Xiang, Jing Shao, and Siheng Chen. Safeagentbench: A benchmark for safe task planning of embodied llm agents, 2025a. URL https://arxiv.org/abs/2412.13178.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1354. URL https://aclanthology.org/2025.acl-long.1354/.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, 2025c.

Runyang You, Hongru Cai, Caiqi Zhang, Qiancheng Xu, Meng Liu, Tiezheng Yu, Yongqi Li, and Wenjie Li. Agent-as-a-judge, 2026. URL https://arxiv.org/abs/2601.05111.

Cunxi Yu and Haoxing Ren. Autonomous evolution of eda tools: Multi-agent self-evolved abc, 2026. URL https://arxiv.org/abs/2604.15082.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Huining Yuan, Zelai Xu, Zheyue Tan, Xiangmin Yi, Mo Guang, Kaiwen Long, Haojia Hui, Boxun Li, Xinlei Chen, Bo Zhao, Xiao-Ping Zhang, Chao Yu, and Yu Wang. MARS: Reinforcing multi-agent reasoning of LLMs through self-play in strategic games. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GCd5v3ehmr.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=G0vdDSt9XM.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024b.

Xiangchi Yuan, Chunhui Zhang, Zheyuan Liu, Dachuan Shi, Leyan Pan, Soroush Vosoughi, and Wenke Lee. Superficial self-improved reasoners benefit from model merging. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5912–5932, 2025.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text, 2024. URL https://arxiv.org/abs/2406.07496.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=46Zgqo4QIU.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of llm agents, 2024. URL https://arxiv.org/abs/2412.15266.

Xingshan Zeng, Weiwen Liu, Xu Huang, Zezhong Wang, Lingzhi Wang, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Ruiming Tang, and Qun Liu. Toolace-r: Model-aware iterative training and adaptive refinement for tool learning, 2025. URL https://arxiv.org/abs/2504.01400.

Jingtao Zhan, Qingyao Ai, Yiqun Liu, Yingwei Pan, Ting Yao, Jiaxin Mao, Shaoping Ma, and Tao Mei. Prompt refinement with image pivot for text-to-image generation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 941–954, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.53. URL https://aclanthology.org/2024.acl-long.53/.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents, 2024b. URL https://arxiv.org/abs/2403.02691.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Large language model-brained GUI agents: A survey. *Transactions on Machine Learning Research*, 2025a. ISSN 2835-8856. URL https://openreview.net/forum?id=xChvYjvXTp.

Di Zhang. Agentdevel: Reframing self-evolving llm agents as release engineering, 2026. URL https://arxiv.org/abs/2601.04620.

Fan Zhang, Shulin Tian, Ziqi Huang, Yu Qiao, and Ziwei Liu. Evaluation agent: Efficient and promptable evaluation framework for visual generative models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7561–7582, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.374. URL https://aclanthology.org/2025.acl-long.374/.

Genghan Zhang, Weixin Liang, Olivia Hsu, and Kunle Olukotun. Adaptive self-improvement LLM agentic system for ML library development. In *Forty-second International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=gdsZ3uMPsY.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng YAN. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025d. URL https://openreview.net/forum?id=mmIAp3cVS0.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents, 2025e. URL https://arxiv.org/abs/2509.24704.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita Velez, Yue Liao, Hongru WANG, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng YAN, Philip Torr, and LEI BAI. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026a. ISSN 2835-8856. URL https://openreview.net/forum?id=RY19y2RI1O. Survey Certification.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves, 2026b. URL https://arxiv.org/abs/2606.09498.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *The Thirteenth International Conference on Learning Representations*, 2025f. URL https://openreview.net/forum?id=V4y0CpX4hK.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Coevoskills: Self-evolving agent skills via co-evolutionary verification, 2026c. URL https://arxiv.org/abs/2604.01687.

Huan Zhang, Yu Song, Ziyu Hou, Santiago Miret, and Bang Liu. Honeycomb: A flexible llm-based agent system for materials science. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3369–3382, 2024a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026d. URL https://openreview.net/forum?id=pUpzQZTvGY.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13643–13658, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.737. URL https://aclanthology.org/2024.acl-long.737/.

Ming Zhang, Yiling Xuan, Qun Ma, and Yuwei Guo. Mlc-agent: Cognitive model based on memory-learning collaboration in llm empowered agent simulation environment, 2025g. URL https://arxiv.org/abs/2507.20215.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised LLM reasoning incentivization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025h. URL https://openreview.net/forum?id=k8Mim6RI5O.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models, 2025i. URL https://arxiv.org/abs/2510.04618.

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, 2026e. URL https://arxiv.org/abs/2601.03192.

Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating hierarchical multi-agent intelligence with the tool-environment-agent(tea) protocol, 2025j. URL https://arxiv.org/abs/2506.12508.

Xiaoying Zhang, Da Peng, Yipeng Zhang, Zonghao Guo, Chengyue Wu, Jen-Tse Huang, Chi Chen, Wei Ke, Helen Meng, and Maosong Sun. Will pre-training ever end? a first step toward next-generation foundation mllms via self-improving systematic cognition. *arXiv preprint arXiv:2503.12303*, 2025k.

Xuan Zhang, Yongliang Shen, Zhe Zheng, Linjuan Wu, Wenqi Zhang, Yuchen Yan, Qiuying Peng, Jun Wang, and Weiming Lu. AskToAct: Enhancing LLMs tool use via self-correcting clarification. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13484–13511, Suzhou, China, November 2025l. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.682. URL https://aclanthology.org/2025.emnlp-main.682/.

Xuan Zhang, Wenxuan Zhang, See-Kiong Ng, and Yang Deng. Self-evolving world models for llm agent planning, 2026f. URL https://arxiv.org/abs/2606.30639.

Yikai Zhang, Ye Rong, Siyu Yuan, Jiangjie Chen, Jian Xie, and Yanghua Xiao. Enhancing language agent strategic reasoning through self-play in adversarial games, 2025m. URL https://arxiv.org/abs/2510.16761.

Yudi Zhang, Meng Fang, Zhenfang Chen, and Mykola Pechenizkiy. Self-evolving llm agents with in-distribution optimization, 2026g. URL https://arxiv.org/abs/2606.07367.

Yuxuan Zhang, Penghui Du, Bo Li, Cong Wei, Junwen Miao, Huaisong Zhang, Songcheng Cai, Yubo Wang, Dongfu Jiang, Yuyu Zhang, Ping Nie, Wenhu Chen, Changqian Yu, and Kelsey R. Allen. Rewardharness: Self-evolving agentic post-training, 2026h. URL https://arxiv.org/abs/2605.08703.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025n.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024a.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025a.

Chenyang Zhao, Xueying Jia, Vijay Viswanathan, Tongshuang Wu, and Graham Neubig. Self-guide: Better task-specific instruction following via self-synthetic finetuning, 2024b. URL https://arxiv.org/abs/2407.12874.

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling, 2025b. URL https://arxiv.org/abs/2507.07998.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025c.

Yang Zhao, Pu Wang, and Hao Frank Yang. How to auto-optimize prompts for domain tasks? adaptive prompting and reasoning through evolutionary domain knowledge adaptation, 2025d. URL https://arxiv.org/abs/2510.21148.

Zhengyang Zhao, Shengjie Ye, Lu Ma, Hao Liang, Hengyi Feng, and Wentao Zhang. Andes: Agent native data evolving synthesis tool for autonomous instruction alignment, 2026a. URL https://arxiv.org/abs/2606.01279.

Zhiyu Zhao, Haoxuan Li, Haifeng Zhang, Jun Wang, Francesco Faccio, Jürgen Schmidhuber, and Mengyue Yang. Curious causality-seeking agents in open-ended worlds. *Advances in Neural Information Processing Systems*, 38:153856–153893, 2026b.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=piecKJ2DlB.

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, 2025a. URL https://arxiv.org/abs/2504.07079.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, 2025b.

Hailin Zhong and Shengxin Zhu. Ai harness engineering: A runtime substrate for foundation-model software agents, 2026. URL https://arxiv.org/abs/2605.13357.

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*, volume 38, pp. 19724–19731, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning llm agents without fine-tuning llms, 2025a. URL https://arxiv.org/abs/2508.16153.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=nnwvwge40d.

Yifei Zhou, Sergey Levine, Jason E Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=9yusqX9DpR.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.

Alvin Zhu, Yusuke Tanaka, Andrew Goldberg, and Dennis Hong. Aura: Autonomous upskilling with retrieval-augmented agents, 2025a. URL https://arxiv.org/abs/2506.02507.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. Wmpo: World model-based policy optimization for vision-language-action models, 2025b. URL https://arxiv.org/abs/2511.09515.

Shiping Zhu, Yibo Yang, Zhengyang Wang, Tiancheng Shen, Dandan Guo, and Ming-Hsuan Yang. H2hmem: A multimodal memory benchmark for agents in human-human interactions, 2026a. URL https://arxiv.org/abs/2606.09461.

Ziyi Zhu, Luka Smyth, Saki Shinoda, and Jinghong Chen. Sage: Stochastic prompt optimization via agent-guided exploration. *arXiv preprint arXiv:2606.18902*, 2026b.

Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, et al. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*, 2023.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents, 2024b. URL https://arxiv.org/abs/2410.10934.

Mingchen Zhuge, Ailing Zeng, Deyao Zhu, Sherry Yang, Vikas Chandra, and Jürgen Schmidhuber. Ai with recursive self-improvement. In *ICLR 2026 Workshop Proposals*, 2026a.

Mingchen Zhuge, Changsheng Zhao, Haozhe Liu, Zijian Zhou, Shuming Liu, Wenyi Wang, Ernie Chang, Gael Le Lan, Junjie Fei, Wenxuan Zhang, et al. Neural computers. *arXiv preprint arXiv:2604.06425*, 2026b.

Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VuVhgEiu20.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JsNUE84Hxi.

Notation

Here we summarize the recurring symbols used throughout the survey and provide a compact guide to their meanings.

Table 6: Summary of notation used in the survey.

Symbol	Description	Reference
General agent formalism		
$\mathcal{A}_t$	Agent configuration at iteration t.	Sec. 3
$\mathcal{A}_{1:t}$	Agent history up to iteration t.	Sec. 3
θ_t	Foundation model parameters.	Sec. 3
$\theta_{1:t}$	Parameter history up to iteration t.	Sec. 3
Σ_t	Scaffold configuration.	Sec. 3
$\Sigma_{1:t}$	Scaffold history up to iteration t.	Sec. 3
p_t	Prompt or instruction component.	Sec. 3, Sec. 6.1
m_t	Memory component.	Sec. 3, Sec. 6.2
$\mathcal{T}_t$	Tool component or tool set.	Sec. 3, Sec. 6.3
g_t	Control, routing, or verification logic.	Sec. 3
π_{θ_t,Σ_t}	Policy induced by model and scaffold.	Sec. 3
X_t	Input, observation, or task context.	Sec. 3
A_t	Agent action or output.	Sec. 3
$\mathcal{C}_t$	Task or deployment context.	Sec. 3
$\mathcal{E}(\cdot)$	Agent execution procedure.	Sec. 3
$\mathcal{U}$	General self-induced update operator.	Sec. 3
$\mathcal{U}_\theta$	Model-parameter update operator.	Sec. 3
$\mathcal{U}_\Sigma$	Scaffold update operator.	Sec. 3
$\mathcal{S}_t$	Learning or improvement signal.	Sec. 4
$\texttt{IMPROVE}(\cdot)$	Abstract improvement operator.	Sec. 4
$\texttt{IMPROVE}_\theta(\cdot)$	Foundation-model improvement operator.	Sec. 5
$\texttt{IMPROVE}_\Sigma(\cdot)$	Scaffold improvement operator.	Sec. 6
Foundation model improvement		
$\mathcal{D}_t$	Intrinsic generative demonstrations.	Sec. 5.1
$\mathcal{D}_t^{\text{gen}}$	Generated training dataset.	Sec. 5.1
(x_i, y_i)	Generated input-output pair.	Sec. 5.1
n_t	Number of generated instances.	Sec. 5.1
$\mathcal{P}_{\text{gen}}$	Agent-induced data distribution.	Sec. 5.1
Φ_t	Data filtering or weighting procedure.	Sec. 5.1
$\mathcal{L}(\theta; \mathcal{D}_t)$	Training objective on $\mathcal{D}_t$.	Sec. 5.1
$\Omega(\theta, \theta_0)$	Regularizer around reference parameters.	Sec. 5.1
λ	Regularization coefficient.	Sec. 5.1
θ_0	Reference or initial checkpoint.	Sec. 5.1
$\theta_t^{(k)}$	Inner-loop parameter state.	Sec. 5.1
$\mathcal{B}_t^{(k)}$	Minibatch at inner step k.	Sec. 5.1
η_k	Inner-loop step size.	Sec. 5.1
K_t	Number of inner update steps.	Sec. 5.1
∇_θ	Gradient with respect to θ.	Sec. 5.1
$\mathcal{Y}_t(x)$	Candidate outputs for input x.	Sec. 5.2
$y_t^{(k)}$	The k-th candidate output.	Sec. 5.2
K	Number of candidate outputs.	Sec. 5.2
ϕ_t	Intrinsic evaluator or judge.	Sec. 5.2
κ_t	Evaluation criterion or rubric.	Sec. 5.2
e_t	Intrinsic evaluative feedback.	Sec. 5.2

Continued on next page

Symbol	Description	Reference
r_t	Scalar reward or reward-like signal.	Sec. 5.2, Sec. 5.3
$y^+ \succ y^-$	Preference relation.	Sec. 5.2
c_t	Natural-language critique.	Sec. 5.2, Sec. 6.1.2
y_t^*	Revised target output.	Sec. 5.2
$C(\cdot)$	Consistency or confidence aggregator.	Sec. 5.2
$R(\cdot)$	Critique-and-revision operator.	Sec. 5.2
τ_t	Interaction trajectory or experience.	Sec. 5.3
s	Environment state.	Sec. 5.3
a	Environment action.	Sec. 5.3
r	Environment reward.	Sec. 5.3
s'	Next environment state.	Sec. 5.3
π_{θ_t,Σ_t}	Induced policy of the FM-based agent.	Sec. 5.3
$W(s_{k+1}, r_k \mid s_k, a_k)$	Learned world model.	Sec. 5.3.2
Scaffold improvement		
$\mathcal{P}$	Prompt search space.	Sec. 6.1.1
p	Candidate prompt.	Sec. 6.1
p^*	Best prompt under a score.	Sec. 6.1.1
$f(p)$	Scalar prompt score.	Sec. 6.1.1
p_{t+1}	Updated prompt.	Sec. 6.1
$\mathrm{Refine}(\cdot)$	Prompt refinement operator.	Sec. 6.1.2
$\mathrm{Critique}(\cdot)$	Critique function.	Sec. 6.1.2
$\mathrm{Output}(p_t)$	Output generated under prompt p_t.	Sec. 6.1.2
P_t	Prompt population.	Sec. 6.1.3
$p_t^{(i)}$	The i-th prompt in P_t.	Sec. 6.1.3
N	Population size.	Sec. 6.1.3
$\mathrm{Fit}(p)$	Prompt fitness function.	Sec. 6.1.3
p_{child}	Offspring prompt.	Sec. 6.1.3
$\mathrm{Crossover}(\cdot)$	Prompt crossover operator.	Sec. 6.1.3
$\mathrm{Mutate}(\cdot)$	Prompt mutation operator.	Sec. 6.1.3
$g(p_t)$	Textual gradient or update guidance.	Sec. 6.1.4
$\oplus$	Textual update operation.	Sec. 6.1.4
$\texttt{IMPROVE}_p$	Prompt improvement operator.	Sec. 6.1
$\texttt{IMPROVE}_m$	Memory improvement operator.	Sec. 6.2
object_t	Stored memory objects.	Sec. 6.2.1
$\mathrm{structure}_t$	Memory structure or index.	Sec. 6.2.2
Create	Memory write operation.	Sec. 6.2.3
Read	Memory retrieval operation.	Sec. 6.2.3
Update	Memory revision operation.	Sec. 6.2.3
Delete	Memory pruning operation.	Sec. 6.2.3
$\texttt{IMPROVE}_{\mathcal{T}}$	Tool improvement operator.	Sec. 6.3
$\mathcal{I}_{\Sigma_t}$	Scaffold improver.	Sec. 6.4
$\langle \Sigma_t \rangle$	Serializable scaffold encoding.	Sec. 6.4
$\mathrm{exec}(\cdot)$	Execution of a scaffold encoding.	Sec. 6.4
$\tilde{\Sigma}_{t+1}$	Proposed scaffold update.	Sec. 6.4
$\mathcal{V}(\cdot)$	Scaffold validation function.	Sec. 6.4
Evaluation and iteration		
t	Outer-loop iteration index.	Throughout
$t+1$	Next self-improvement iteration.	Throughout
k	Inner step or candidate index.	Sec. 5.1, Sec. 5.2
arg max	Maximizer of an objective.	Sec. 6.1.1
arg min	Minimizer of an objective.	Sec. 5.1
$\langle \cdot \rangle$	Serializable encoding notation.	Sec. 6.4