From Copilots to Colleagues: A Survey of Autonomous Research Agents

Deli Chen* DeepSeek-V4-Pro GPT-Image2

Abstract

The rapid advancement of foundation models has catalyzed a paradigm shift from AI systems that assist researchers to agents capable of conducting research autonomously. Yet this emerging field lacks a unified analytical framework: code agents, scientific discovery systems, and multi-agent research platforms have developed largely in isolation, with inconsistent terminology and incomparable evaluation. This survey addresses this gap through four contributions. First, we propose a five-level autonomy taxonomy (L1–L5)—from code autocomplete to fully self-directed research agendas—that provides precise vocabulary for characterizing and comparing systems. Second, we identify and analyze four dominant architectural patterns (single-agent loops, multi-agent collaboration, hierarchical orchestration, and tool-augmented execution) with a comparative framework evaluating trade-offs across scalability, cost, reliability, and human oversight. Third, we provide detailed analysis of 17 major systems across a six-dimensional feature matrix, revealing that current frontier systems operate at L4 (multi-step autonomous execution within bounded domains) while L5 remains aspirational. Fourth, we identify six fundamental open problems—cognitive loops, context limitations, novelty evaluation, reproducibility, safety, and cost—and propose concrete research directions for each. Our analysis reveals that the most critical barriers to L5 autonomy are not raw capability but persistent knowledge accumulation, reliable self-evaluation, and principled scaling of agent architectures. We survey over 95 papers across machine learning, software engineering, and scientific discovery venues, providing the first unified treatment of autonomous research agents as a coherent research area.

1 Introduction

In 2022, AI systems served as sophisticated typewriters: they predicted the next token, suggested the next line, and completed the human's thought. By 2025, they had become something qualitatively different—*colleagues* that independently navigate codebases, design experiments, debug failures, and produce research artifacts with minimal human oversight. This transition from tools to collaborators represents a paradigm shift in how research is conducted, with implications spanning computer science, the natural sciences, and the social organization of knowledge production.

The inflection point was rapid and decisive. Within eighteen months (early 2024 to mid-2025), the resolution rate on SWE-bench—a benchmark of real-world software engineering tasks—climbed from under 5% to over 70% (Jimenez et al., 2024; Anthropic, 2024b). Systems like the AI Scientist (Lu et al., 2024) demonstrated end-to-end autonomous research at $15 per paper. FunSearch (Romera-Paredes et al., 2024) produced genuinely novel mathematical discoveries verified by domain experts. These are not incremental improvements to existing tools but qualitative transitions in the nature of AI-human collaboration: for the first time, AI systems can *drive* the research process rather than merely assisting a human driver.

*This paper was partially generated by the **Deli AutoResearch SKILL**, an autonomous research agent framework. Personal research project—all opinions are my own and do not represent the views of any company or organization. Correspondence: chendeli96@gmail.com

This survey provides a comprehensive analysis of *autonomous research agents*—systems that, given a high-level research objective, can independently execute the iterative cycle of scientific inquiry: hypothesis generation, experimental design, execution, analysis, and refinement. We define our scope precisely: we include systems that operate at or above Level 3 in our proposed autonomy taxonomy (multi-step autonomous operation with strategic self-direction), while excluding purely assistive tools (code completion, search engines) and systems that require human approval at every step. Our focus spans code agents, scientific discovery systems, and general research assistants, unified by their shared pursuit of autonomous intellectual contribution.

The stakes of this transition extend beyond productivity gains. If AI systems can reliably conduct research, the implications touch the foundations of academic incentive structures, intellectual property, scientific reproducibility, and the social contract between researchers and funding bodies. Early evidence suggests both promise (dramatically accelerated iteration cycles, reduced barriers to entry for complex engineering tasks) and peril (reproducibility challenges from non-deterministic agents, dual-use risks of autonomous scientific capability, potential concentration of research power among institutions with access to the most capable systems). A rigorous survey that maps the landscape, identifies recurring patterns, and surfaces open problems is therefore timely—both for researchers building these systems and for policymakers grappling with their implications.

Why Now? Three concurrent developments explain the 2024–2026 emergence of viable research agents. First, foundation models crossed capability thresholds— GPT-4 (OpenAI, 2023), Claude (Anthropic, 2024a, 2025), and Gemini (Gemini Team, Google, 2023) demonstrated the reasoning, planning, and self-correction capabilities necessary for sustained autonomous operation. The subsequent advent of reasoning models such as OpenAI o1 (OpenAI, 2024b) and DeepSeek-R1 (DeepSeek AI, 2025) further elevated the ceiling for complex multi-step inference. These models exhibit strong performance across the diverse subtasks of research (literature comprehension, code generation, mathematical reasoning, experimental design), providing a general-purpose substrate upon which agent architectures can be built.

Second, agent architectures matured from fragile prototypes (AutoGPT's infinite loops; Richards, 2023) to robust systems with principled error recovery, hierarchical planning, and tool-augmented execution (Yao et al., 2023b; Shinn et al., 2023; Hong et al., 2024). The progression from ReAct's simple interleaving of thoughts and actions (Yao et al., 2023b) through Reflexion's episodic self-improvement (Shinn et al., 2023) to sophisticated multi-agent orchestration (Wu et al., 2024a; Hong et al., 2024) represents a maturation from proof-of-concept to engineering discipline.

Third, evaluation infrastructure reached sufficient maturity—benchmarks like SWE-bench (Jimenez et al., 2024), AgentBench (Liu et al., 2024b), and GAIA (Mialon et al., 2023) provide standardized, reproducible evaluation that enables systematic comparison and rapid iteration. Without reliable measurement, the field risked degenerating into demo-driven hype; with it, genuine progress becomes distinguishable from superficial improvement. This confluence transformed autonomous research from a speculative vision into an engineering discipline with measurable progress.

Relationship to Existing Surveys. Several surveys cover adjacent territory. Wang et al. (2024b) provide a broad overview of LLM-based autonomous agents, focusing on construction methodology (profiling, memory, planning, action modules) rather than research-specific applications. Xi et al. (2023) survey the "rise" of LLM agents across general domains. Neither focuses specifically on *research* agents—systems whose objective is novel intellectual contribution rather than task completion. Our survey fills this gap by examining the unique challenges of research autonomy (novelty evaluation, open-ended exploration, scientific rigor) and the specific architectural

adaptations they demand.

Scope and Boundaries. This survey covers systems designed for autonomous intellectual work: generating novel knowledge, solving open problems, and producing research artifacts. We include systems from software engineering (SWE-Agent, Devin, Claude Code, OpenHands), scientific discovery (Coscientist, ChemCrow, FunSearch), and general research (AI Scientist, STORM, GPT-Researcher). We *exclude* systems that are purely reactive (chatbots), domain-specific optimization tools without agent architectures (AutoML, NAS), and embodied agents whose primary challenge is perception and motor control rather than intellectual inquiry. The boundary between included and excluded systems is our L1–L5 taxonomy: we focus on L3–L5 while acknowledging L1–L2 as historical context. We cover work published through early 2026, with emphasis on the period 2023–2025 during which the field underwent its most rapid development.

Contributions

This survey makes four contributions to the field:

1. **Comprehensive Autonomy Taxonomy.** We propose and validate an L1–L5 taxonomy of research agent autonomy (analogous to SAE driving levels), providing a precise vocabulary for characterizing system capabilities and comparing across architectures (§2).

2. **Systematic Architecture Analysis.** We identify and analyze the dominant architectural patterns—single-agent loops, multi-agent systems, hierarchical orchestration, and tool-augmented agents—with a comparative framework evaluating trade-offs across scalability, cost, reliability, and human oversight (§3).

3. **Feature-Annotated System Comparison.** We provide detailed analysis of 17 major systems across a six-dimensional feature matrix, revealing historical maturation patterns and identifying capability gaps (§4).

4. **Research Agenda for Open Problems.** We identify six fundamental challenges—cognitive loops, context limitations, novelty evaluation, reproducibility, safety, and cost—and propose concrete research directions for each, synthesizing insights across the surveyed systems (§6).

Paper Organization

The remainder of this paper is organized as follows. Section 2 establishes foundations: formal definitions, the L1–L5 autonomy taxonomy, required capabilities, and relationships to adjacent fields. Section 3 analyzes architectural patterns from single-agent reasoning loops through graph-based multi-agent orchestration. Section 4 provides detailed examination of 17 systems across four application domains with a comprehensive comparison matrix. Section 5 surveys evaluation methodology: benchmarks, quality metrics, efficiency measurement, and safety considerations. Section 6 identifies six open problems defining the research frontier, with concrete directions for each. Section 7 outlines a forward-looking research agenda, and Section 8 concludes with synthesis and recommendations for the community.

2 Background and Problem Formulation

This section establishes the conceptual foundations for the remainder of the survey. We formally define autonomous research agents (§2.1), propose a taxonomy of autonomy levels (§2.2), enumerate the core capabilities such systems require (§2.3), and delineate the boundary with adjacent fields (§2.4).

2.1 Definition of Autonomous Research Agent

We define an *autonomous research agent* as a software system that, given a high-level research objective, can independently execute the iterative cycle of scientific inquiry—hypothesis generation, experimental design, execution, analysis, and refinement—with minimal or no human intervention during the execution loop. This definition intentionally excludes systems that merely *assist* a human researcher (e.g., code completion tools, literature search engines) and instead focuses on systems that *drive* the research process.

Formally, let $\mathcal{G}$ denote a research goal specified in natural language, $\mathcal{E}$ a set of available tools and environments, and $\mathcal{K}$ an initial knowledge base. An autonomous research agent $\mathcal{A}$ is a function that produces a sequence of actions $a_1, a_2, \ldots, a_T$ such that:

$$\mathcal{A}(\mathcal{G}, \mathcal{E}, \mathcal{K}) \to (a_1, o_1, a_2, o_2, \ldots, a_T, o_T, \mathcal{R}) \tag{1}$$

where o_i denotes the observation following action a_i, and $\mathcal{R}$ represents the final research artifact (e.g., a paper, code, experimental results). Critically, the agent selects each a_i conditioned on the history $(a_1, o_1, \ldots, a_{i-1}, o_{i-1})$ without requiring human approval at each step—distinguishing it from interactive assistants.

This formulation subsumes several related concepts. The *agent loop* of Yao et al. (2023b)—interleaving reasoning traces with actions—provides the basic execution mechanism. *Reflexion* (Shinn et al., 2023) adds self-evaluative feedback across episodes. The cognitive architecture framework of Sumers et al. (2023) further structures these components into memory, decision-making, and metacognition modules. Our definition unifies these perspectives under the lens of research autonomy.

Three properties distinguish research agents from general-purpose agents: (1) *open-endedness*—the solution space is not predefined; (2) *novelty requirement*—the output should contribute knowledge beyond what exists in $\mathcal{K}$; and (3) *verifiability*—claims must be empirically or formally substantiated. These properties make research agency substantially more challenging than task completion in constrained environments.

2.2 Taxonomy of Autonomy Levels

Drawing an analogy to the SAE levels of driving automation (SAE International, 2021), we propose a five-level taxonomy for research agent autonomy. This taxonomy characterizes systems along two axes: the *scope of delegated decisions* (what the agent decides independently) and the *duration of unsupervised operation* (how long it runs without human checkpoints).

Table 1: Proposed taxonomy of autonomy levels for research agents. The "Human Role" column indicates what the human must provide beyond the initial goal specification.

Level	Description	Example Systems	Human Role
L1	Autocomplete	GitHub Copilot, TabNine	Drives every step; agent suggests completions
L2	Task execution	ChatGPT + tools, Claude chat	Specifies task; approves each action
L3	Multi-step with checkpoints	Claude Code, Cursor Agent	Sets goal; reviews at checkpoints
L4	Full autonomy, bounded	Devin, AI Scientist, SWE-Agent	Provides goal; evaluates final output
L5	Self-directed research	*(Hypothetical)*	Sets research area; agent chooses problems

Level 1: Autocomplete. The agent operates at the granularity of individual tokens or lines, predicting the most likely continuation of a human-written sequence. The human retains full control over direction, structure, and correctness. Systems at this level—such as GitHub Copilot

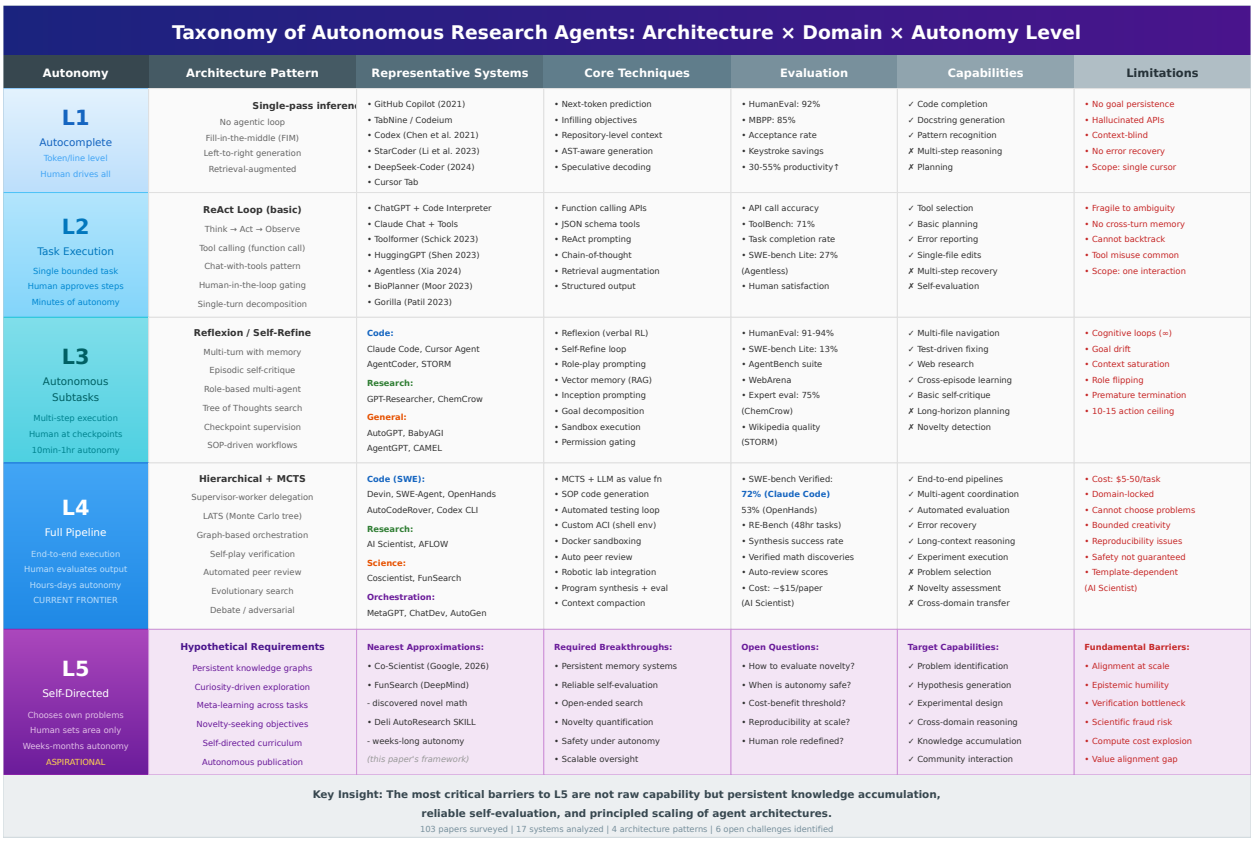

Figure 1: Comprehensive taxonomy of autonomous research agents across seven dimensions. Each autonomy level (L1–L5) is characterized by its architecture pattern, representative systems, core techniques, evaluation methodology, capabilities, and current limitations. The field's frontier lies at L4, with L5 remaining aspirational. This taxonomy unifies 103 surveyed papers and 17 analyzed systems into a single comparative framework.

and code completion models descended from Codex (Chen et al., 2021)—provide productivity gains of 30–55% on controlled coding tasks but cannot pursue multi-step objectives independently.

Level 2: Task Execution with Human Approval. The agent can decompose a well-specified task into steps and execute them, but each step requires explicit or implicit human approval. Conversational AI assistants (e.g., ChatGPT with plugins, Claude with tool use) operate at this level: they can search the web, execute code, and synthesize information, but the human guides the conversation and validates intermediate results. Chain-of-thought prompting (Wei et al., 2022) enables more complex reasoning within a single turn, but the human remains the strategic decision-maker.

Level 3: Multi-Step with Checkpoints. The agent autonomously executes sequences of 10–100 actions toward a goal, requesting human review only at predefined checkpoints or upon encountering uncertainty. Modern code agents exemplify this level: they navigate repositories, edit

multiple files, run tests, and iterate on failures. The key distinction from L2 is that the agent makes *tactical* decisions (e.g., which file to edit, how to fix a test failure) without per-step approval, while the human retains *strategic* oversight. This level demands planning capabilities (Huang et al., 2024b) and the ability to recover from errors via self-reflection (Shinn et al., 2023).

Level 4: Full Autonomy with Self-Correction. The agent receives a research objective and operates independently for hours or days, including recovering from failures, revising strategies, and producing a complete research artifact. Lu et al. (2024) demonstrate this level: their AI Scientist system generates ideas, writes and runs experiments, produces a full paper, and even conducts automated peer review—all without human intervention during the pipeline. Similarly, Devin (Cognition Labs, 2024) and SWE-Agent (Yang et al., 2024) operate autonomously on software engineering tasks, making architectural decisions, debugging complex failures, and submitting complete pull requests. The human evaluates the *final output* rather than supervising the process.

Level 5: Self-Directed Research Agenda. At the highest level of autonomy, the agent not only executes research but *selects its own research problems*, allocates resources across a portfolio of projects, and builds on its own prior results over extended time horizons. No current system fully achieves L5, though elements are visible in systems like Voyager (Wang et al., 2023a)—which generates its own curriculum of increasingly difficult tasks—and FunSearch (Romera-Paredes et al., 2024), which iteratively discovers new mathematical constructions by building on previous successful programs. The transition from L4 to L5 requires advances in intrinsic motivation, persistent memory, and meta-level research taste that remain open challenges (see §6).

The boundaries between levels are not rigid; many systems operate at different levels depending on task complexity. A single agent framework may function at L3 for routine tasks while degrading to L2 when encountering novel situations requiring human judgment. We use this taxonomy primarily as an analytical tool for comparing systems rather than as a strict classification.

Limitations of the L1–L5 Framework. We acknowledge several limitations of our proposed taxonomy. First, it is primarily *descriptive* rather than predictive—it characterizes current systems but does not provide a mechanistic theory of how systems transition between levels. Second, the taxonomy privileges the *scope of autonomy* dimension while collapsing other important distinctions: two L4 systems may differ radically in reliability, domain specificity, and failure modes. Third, the analogy to SAE driving levels, while intuitive, may be misleading—driving operates in a physical environment with well-defined safety constraints, whereas research is open-ended and success criteria are socially constructed. Finally, the taxonomy has not been empirically validated through user studies or expert elicitation; it represents our analytical synthesis of the literature rather than a consensus classification. Future work should investigate whether practitioners find these levels meaningful and whether alternative axes (e.g., reliability, domain breadth, or creative contribution) better capture the dimensions along which research agents vary.

2.3 Required Capabilities

Analysis of existing autonomous research agents reveals five core capabilities that jointly enable increasing levels of autonomy:

Planning and Decomposition. The ability to break a high-level research goal into a structured sequence of sub-goals, allocate effort across them, and dynamically re-plan when initial approaches fail. Tree-structured planning (Yao et al., 2023a) and Monte Carlo tree search over action trajectories (Zhou et al., 2023a) provide formal frameworks, while hierarchical task decomposition—as

implemented in MetaGPT (Hong et al., 2024) and TaskWeaver (Qiao et al., 2023)—offers practical instantiations. Effective planning at L4+ requires reasoning over long time horizons (hundreds to thousands of steps) and handling combinatorial branching in the space of possible research directions.

Tool Use and Environment Interaction. Research agents must interact with external tools: executing code, querying databases, searching literature, running experiments, and controlling laboratory equipment. Schick et al. (2023) demonstrated that language models can learn to invoke tools at appropriate points in generation. Subsequent systems have expanded the tool repertoire dramatically: ChemCrow (Bran et al., 2024) integrates 18 chemistry-specific tools, while HuggingGPT (Shen et al., 2023) orchestrates hundreds of specialized models. The design of the *agent-computer interface*—how tools are presented to and invoked by the agent—significantly impacts performance (Yang et al., 2024; Wang et al., 2024d).

Self-Evaluation and Error Recovery. Autonomous operation requires the agent to assess the quality of its own outputs and recover from errors without human intervention. This capability manifests as self-refinement loops (Madaan et al., 2023), verbal self-reflection (Shinn et al., 2023), and automated testing and validation. Critically, self-evaluation for research tasks must go beyond verifying functional correctness to assessing novelty, significance, and methodological soundness—a substantially harder problem where current systems exhibit significant limitations.

Memory and Knowledge Management. As research spans multiple sessions and builds on prior results, agents require memory systems that persist beyond a single context window. Packer et al. (2023) propose a virtual memory hierarchy analogous to operating systems, with paging between active context and external storage. Park et al. (2023) demonstrate a memory architecture combining observation streams, reflection synthesis, and retrieval-based recall. Wang et al. (2023a) implement a persistent skill library that accumulates reusable capabilities across episodes. For L5 autonomy, agents would need *institutional memory*—the ability to maintain coherent research programs over months or years of accumulated knowledge.

Collaboration and Communication. Multi-agent systems (Wu et al., 2024a; Li et al., 2023; Hong et al., 2024) demonstrate that collaboration between specialized agents can exceed the capabilities of any single agent. This includes role-based division of labor (Qian et al., 2024), adversarial debate for quality assurance, and structured communication protocols that reduce hallucination. For human-AI collaborative research, the agent must also effectively communicate uncertainty, explain its reasoning, and accept guidance at appropriate intervention points.

2.4 Relationship to Adjacent Fields

Autonomous research agents build upon—but are distinct from—several established fields in computer science:

Automated Machine Learning (AutoML). AutoML (Hutter et al., 2019) automates specific components of the ML pipeline: hyperparameter optimization, feature engineering, and model selection. Neural Architecture Search (NAS) (Zoph and Le, 2017) further automates the design of network architectures. While these systems achieve remarkable performance within their prescribed search spaces, they operate on *well-defined optimization problems* with clear objective functions. Autonomous research agents, by contrast, must identify *what* to optimize—formulating problems

rather than merely solving predefined ones. The scope of automation in AutoML corresponds to our L2–L3 levels within the narrow domain of ML pipeline construction.

Program Synthesis. Program synthesis (Gulwani et al., 2017) seeks to automatically generate programs satisfying a specification (input-output examples, logical constraints, or natural language). Modern neural-guided synthesis approaches, including large-scale code generation models (Chen et al., 2021; Li et al., 2022), have achieved human-competitive performance on constrained benchmarks. However, program synthesis traditionally assumes a *fixed specification*—the challenge is implementation, not formulation. Research agents must both *formulate* the problem (decide what program to write and why) and *implement* a solution, a strictly harder task that corresponds to L4+ autonomy.

AI for Scientific Discovery. The broader field of AI for science (Wang et al., 2023b) encompasses computational approaches to accelerating discovery across chemistry, biology, physics, and mathematics. This includes domain-specific tools (molecular simulation, protein folding, theorem provers) that do not require language model foundations. Autonomous research agents represent a particular *architectural approach* within this field—one that leverages foundation models as general-purpose reasoners capable of operating across domains, in contrast to hand-engineered pipelines optimized for specific scientific tasks. The key differentiator is generality: a research agent should, in principle, be redirectable to new domains without architectural changes.

Retrieval-Augmented Generation (RAG). RAG systems (Lewis et al., 2020; Borgeaud et al., 2022) ground language model generation in retrieved documents, improving factuality and enabling access to current information. While RAG provides a critical *memory component* for research agents, it does not alone constitute autonomous research—retrieval is a tool, not an agent architecture. Research agents integrate RAG as one of many capabilities alongside planning, execution, and self-evaluation.

In summary, autonomous research agents represent a *convergence* of these fields: they apply AutoML-style automation to the full research cycle, leverage program synthesis for implementation, build on scientific AI for domain expertise, and use RAG for knowledge access—all unified under a foundation-model-powered agent architecture capable of open-ended operation. The following sections examine how specific systems instantiate this synthesis.

3 Architecture Patterns

The design space for autonomous research agents encompasses a rich variety of architectural patterns, each embodying different trade-offs between simplicity, scalability, and capability. This section provides a systematic analysis of the dominant paradigms: single-agent reasoning loops (§3.1), multi-agent systems (§3.2), hierarchical orchestration (§3.3), and tool-augmented agents (§3.4). We conclude with a comparative analysis across key dimensions (§3.5).

3.1 Single-Agent Loops

The simplest and most widely adopted architecture for autonomous agents is the *single-agent loop*: a single language model iteratively observes the environment, reasons about the next action, executes it, and incorporates feedback. Despite its simplicity, this pattern forms the backbone of most L3–L4 systems and admits considerable variation in the reasoning strategy employed.

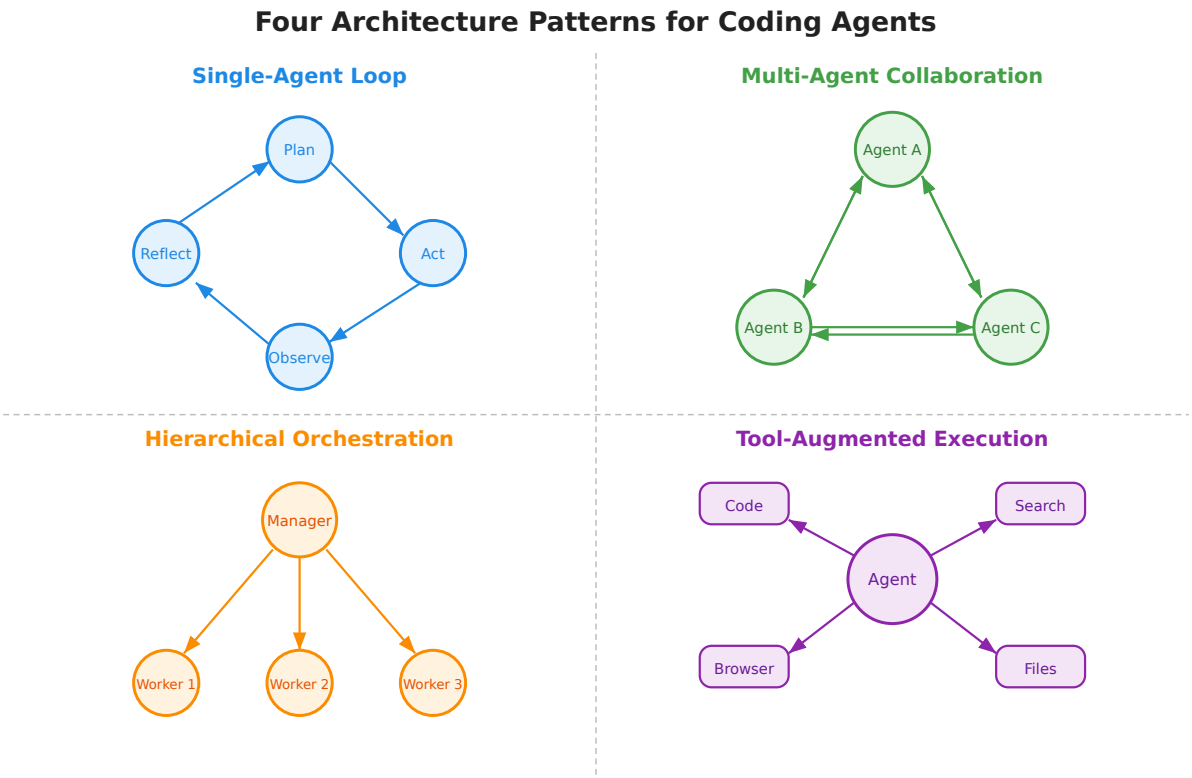

Figure 2: The four dominant architectural patterns for autonomous research agents. Each pattern offers distinct trade-offs: single-agent loops are simplest but limited in scale; multi-agent collaboration enables parallel exploration; hierarchical orchestration provides structured delegation; tool-augmented execution extends capabilities through external integrations.

ReAct: Interleaved Reasoning and Acting. Yao et al. (2023b) introduced the ReAct paradigm, which interleaves free-form *reasoning traces* ("thoughts") with concrete *actions* in a single generation stream. At each step, the agent produces a thought that analyzes the current situation, selects an action (e.g., searching a database, executing code), observes the result, and continues. This interleaving addresses a fundamental limitation of pure chain-of-thought reasoning (Wei et al., 2022): by grounding reasoning in environmental observations, the agent avoids hallucinating facts and can dynamically adjust its strategy based on intermediate results. ReAct achieves strong performance on knowledge-intensive tasks (HotpotQA, FEVER) and interactive decision-making environments (ALFWorld, WebShop), establishing the template that most subsequent agent systems adopt.

The key insight of ReAct is that *reasoning without acting* leads to hallucination (the model generates plausible-sounding but unverified claims), while *acting without reasoning* leads to inefficient exploration (the agent takes actions without strategic purpose). The synergy between these modes—where thoughts guide action selection and observations ground subsequent reasoning—is now considered a foundational principle of agent design.

Reflexion: Verbal Self-Reflection. While ReAct operates within a single episode, Shinn et al. (2023) introduced *Reflexion*, which adds a meta-cognitive layer operating *across* episodes. After completing (or failing) a task, the agent generates a verbal reflection analyzing what went wrong and what strategies might improve future attempts. These reflections are stored in an episodic

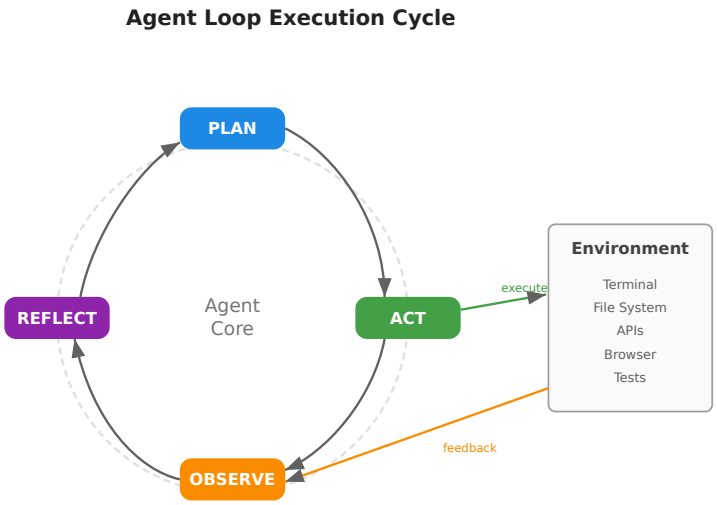

Figure 3: The canonical agent execution loop. The agent iteratively plans actions based on its current state, executes them in the environment, observes outcomes, and reflects on progress. This cycle—instantiated by ReAct, Reflexion, and related frameworks—forms the core execution mechanism for L3–L5 systems.

memory buffer and prepended to subsequent attempts, enabling learning without gradient updates. Reflexion achieves 91% pass@1 on HumanEval (compared to 67% for the base model) through iterative self-improvement, demonstrating that linguistic self-criticism can substitute for traditional reinforcement learning in many settings.

The Reflexion architecture comprises three components: an *actor* (the agent executing tasks), an *evaluator* (providing success/failure signals, which may be external tests or self-assessment), and a *self-reflection* module (generating natural language analysis of failures). This decomposition separates the concerns of execution, evaluation, and learning, providing a clean template for self-improving agents.

LATS: Monte Carlo Tree Search over Actions. Zhou et al. (2023a) unify reasoning, acting, and planning through Language Agent Tree Search (LATS), which applies Monte Carlo Tree Search (MCTS) to the space of agent trajectories. Rather than committing to a single reasoning path, LATS explores multiple branches, using the language model both as a policy (to propose actions) and as a value function (to evaluate the promise of partial trajectories). Backpropagation of rewards through the search tree enables the agent to identify and commit to the most promising strategies.

LATS represents a significant advancement in *deliberate planning* for language agents. While ReAct makes myopic decisions (choosing the locally best action), LATS considers the global structure of the solution space. On programming tasks (HumanEval), LATS achieves 94.4% accuracy, surpassing both ReAct (67%) and Reflexion (91%). However, this comes at substantially higher computational cost—each node in the search tree requires a full LLM inference call, making LATS approximately 5–20× more expensive than single-pass approaches.

Self-Refine and Iterative Improvement. Madaan et al. (2023) propose Self-Refine, where a

single model iteratively generates an output, critiques it, and produces an improved version. Unlike Reflexion (which operates across separate episodes), Self-Refine applies within a single problem-solving session through a tight generate-critique-refine loop. This pattern is particularly effective for tasks with clear quality signals (code generation, mathematical reasoning, text quality) where the same model can serve as both generator and critic. Empirically, Self-Refine improves output quality by 5–25% absolute across diverse tasks including dialogue, code optimization, and constrained generation.

Tree of Thoughts. Yao et al. (2023a) generalize chain-of-thought prompting from a single linear chain to a tree of intermediate reasoning states. At each step, the model generates multiple possible "thoughts" (intermediate reasoning steps), evaluates their promise via deliberate assessment, and explores the most promising branches using breadth-first or depth-first search. This approach is especially effective for problems requiring exploration and backtracking—such as the Game of 24, creative writing, and crossword puzzles—where single-path reasoning frequently reaches dead ends.

RAP: Reasoning via Planning. Hao et al. (2023) frame reasoning as planning by treating the LLM simultaneously as a *world model* (predicting the consequences of actions) and a *reasoning agent* (selecting actions). This dual role enables the application of classical planning algorithms, including MCTS, where the LLM both proposes actions and simulates their outcomes. RAP achieves substantial improvements on blocksworld planning, math reasoning, and logical inference, demonstrating that even without external environment feedback, LLMs can benefit from structured search when they serve as their own simulator.

Comparison of Single-Agent Architectures. These approaches occupy different points in the trade-off space between computational cost and solution quality. ReAct provides an efficient baseline suitable for tasks where greedy action selection suffices. Reflexion adds cross-episode learning for tasks where initial attempts are likely to fail. Self-Refine targets within-episode quality improvement when critique signals are available. Tree of Thoughts and LATS invest significantly more computation to explore the solution space systematically, yielding the highest performance on challenging tasks at correspondingly higher cost. The choice among these architectures should be guided by the task difficulty, available compute budget, and whether external evaluation signals are accessible.

3.2 Multi-Agent Systems

While single-agent architectures rely on one model to handle all aspects of a task, *multi-agent systems* distribute responsibility across multiple specialized agents that communicate and collaborate. This paradigm draws inspiration from organizational theory: just as human research teams benefit from division of labor, multi-agent systems can exceed the capabilities of any individual agent through specialization and collaborative refinement.

CAMEL: Role-Playing for Cooperative Agents. Li et al. (2023) introduced CAMEL (Communicative Agents for "Mind" Exploration), a role-playing framework where two agents—an "AI assistant" and an "AI user"—engage in autonomous cooperation through *inception prompting*. Each agent is assigned a role (e.g., a Python programmer and a stock trader) and cooperates to accomplish a specified task through multi-turn dialogue. The key innovation is that role assignment through system prompts creates emergent specialization without any architectural modifications to the underlying model. CAMEL demonstrates that simple role-playing mechanisms can induce cooperative behaviors including task decomposition, knowledge sharing, and quality control.

The CAMEL framework also reveals failure modes of multi-agent cooperation: *role flipping* (agents switching roles mid-conversation), *assistant instructing* (the assistant giving instructions rather than following them), and *conversation termination* issues. These findings have informed subsequent multi-agent designs that incorporate more structured communication protocols.

AutoGen: Conversable Agents with Human Oversight. Wu et al. (2024a) propose AutoGen, a Microsoft framework that enables the creation of *conversable agents* supporting multiple interaction patterns: agent-agent, agent-human, and agent-tool. AutoGen's key design principle is *flexible conversation patterns*: agents can be configured in sequential chat (pipeline), group chat (broadcast), or nested patterns (hierarchical delegation). Each agent can incorporate LLM capabilities, human input, or deterministic tool execution, enabling hybrid human-AI teams.

AutoGen distinguishes itself through its emphasis on *human-in-the-loop* configurations. While fully autonomous operation is supported, the framework provides natural intervention points where humans can redirect, correct, or approve agent actions. This makes AutoGen particularly suitable for L3 systems where human oversight is desired at strategic checkpoints. The group chat manager pattern—where a designated agent routes messages to the most relevant participant—provides a lightweight mechanism for dynamic task allocation without explicit hierarchical control.

MetaGPT: Standard Operating Procedures for Multi-Agent Collaboration. Hong et al. (2024) take a more structured approach by encoding *Standard Operating Procedures* (SOPs) into multi-agent collaboration. Rather than allowing free-form conversation, MetaGPT defines explicit roles (Product Manager, Architect, Engineer, QA) with formalized inputs and outputs at each stage. Agents communicate via *structured artifacts* (design documents, API specifications, code) rather than natural language chat, reducing ambiguity and hallucination. On software development benchmarks, MetaGPT achieves a 100% task completion rate (compared to 67% for ChatDev (Qian et al., 2024)) by constraining agent interactions to well-defined interfaces.

The SOP-based approach addresses a critical failure mode of unconstrained multi-agent chat: as the number of agents grows, unstructured conversation becomes increasingly incoherent, with agents repeating, contradicting, or ignoring each other's contributions. By imposing document-mediated communication, MetaGPT ensures that each agent receives precisely the information it needs in a format it can process reliably.

Debate and Discussion Frameworks. Du et al. (2024) demonstrate that multi-agent *debate*—where multiple LLM instances generate independent responses and then critique each other's answers over multiple rounds—significantly improves factual accuracy and mathematical reasoning. The mechanism is analogous to scientific peer review: exposure to alternative perspectives helps agents identify and correct errors in their own reasoning. Liang et al. (2023) extend this to encourage *divergent thinking*, showing that adversarial discussion produces more creative and diverse solutions than single-agent generation.

These debate frameworks suggest a powerful paradigm for research agents: rather than relying on a single model's judgment, critical decisions (hypothesis selection, experimental design, result interpretation) can be subjected to multi-agent deliberation, improving robustness against individual model biases and hallucinations.

Mixture-of-Agents. Wang et al. (2024a) propose a layered architecture where multiple agents at each layer process the same input, and subsequent layers aggregate and refine the outputs of the previous layer. This approach—inspired by mixture-of-experts architectures—enables collaborative

refinement without explicit role assignment. Each agent in a layer can observe the full outputs of all agents in the previous layer, leading to emergent specialization where some agents focus on accuracy, others on creativity, and yet others on self-correction. The resulting system achieves state-of-the-art performance on several benchmarks while remaining architecturally simple.

3.3 Hierarchical Orchestration

As task complexity grows, flat multi-agent communication becomes insufficient. *Hierarchical orchestration* introduces explicit supervision relationships: a high-level "supervisor" agent decomposes tasks and delegates subtasks to specialized "worker" agents, monitoring their progress and intervening when necessary.

Supervisor-Worker Patterns. The supervisor-worker pattern mirrors traditional management hierarchies: a planning agent maintains the global task state, decomposes objectives into subtasks, assigns them to appropriate workers, and synthesizes results. Claude Code's multi-agent architecture (Anthropic, 2024b) exemplifies this pattern: a primary agent maintains conversational context and high-level planning, spawning sub-agents for specific tasks (file editing, test execution, web search) that operate in isolated contexts and report results back. This isolation prevents context pollution—where irrelevant information from one subtask degrades performance on another—while the supervisor maintains coherent global state.

The supervisor-worker pattern naturally maps to our autonomy taxonomy: the supervisor operates at L3–L4 (making strategic decisions autonomously), while workers may operate at L2–L3 (executing well-defined subtasks with limited autonomy). This composition allows the system as a whole to achieve higher autonomy than any individual component.

Task Decomposition Approaches. Effective hierarchical orchestration requires robust *task decomposition*: breaking a complex research objective into a tree of subtasks that can be independently executed. Several approaches have been explored: (1) *recursive decomposition*, where the supervisor iteratively breaks tasks into smaller pieces until each is tractable for a single worker (Nakajima, 2023); (2) *plan-then-execute*, where a complete task graph is generated before execution begins (Hong et al., 2024); (3) *dynamic replanning*, where the task structure evolves as execution reveals new information (Wang et al., 2023c). Dynamic replanning is essential for research tasks where the problem structure is not known in advance and discoveries during execution fundamentally alter the approach.

Self-Play and Iterated Refinement. An emerging pattern combines hierarchical control with iterative self-play: one agent generates a research artifact (code, paper, experiment), another agent reviews or evaluates it, and the generation agent refines based on feedback. This adversarial dynamic—reminiscent of generative adversarial networks—drives quality improvement through competition. The AI Scientist (Lu et al., 2024) employs this pattern: after generating a paper, a separate "reviewer" agent evaluates it using NeurIPS review criteria, and the feedback informs subsequent iterations. Huang et al. (2024a) similarly separate code generation, test design, and verification into distinct agents that iteratively improve through mutual feedback.

Graph-Based Orchestration. Recent work moves beyond tree-structured hierarchies to *graph-based* orchestration, where agents are connected in arbitrary computational graphs. Zhuge et al. (2024) model multi-agent systems as optimizable graphs where edges represent communication channels and nodes represent agent operations. The graph structure itself can be optimized—adding, removing, or rewiring connections—to maximize task performance. Zhang et al. (2025a)

similarly use MCTS to search over the space of possible agentic workflows, automatically discovering effective orchestration patterns for specific tasks. These approaches represent a shift from hand-designed architectures to *learned* organizational structures.

3.4 Tool-Augmented Agents

A defining characteristic of autonomous research agents is their ability to interact with external tools and environments. Tool augmentation transforms language models from passive text generators into active participants in computational and physical workflows.

Code Execution Environments. The most impactful tool category for research agents is code execution. Yang et al. (2024) introduce the concept of an *Agent-Computer Interface* (ACI)—the set of commands, file viewers, and navigation tools available to a code agent—and demonstrate that ACI design dramatically affects performance. SWE-Agent's custom ACI (providing commands for file navigation, editing, and search) achieves 12.5% on SWE-bench, compared to 3.8% for a naive shell interface. Wang et al. (2024d) further show that representing all actions as executable Python code (rather than JSON or natural language) improves agent capabilities by providing a unified, composable action space. The CodeAct paradigm has become standard in modern code agents, with systems like OpenHands (Wang et al., 2024e) adopting code-based action spaces throughout.

The design of the execution environment involves critical safety decisions: sandboxing (preventing agents from modifying production systems), resource limits (bounding computation and memory), and permission models (controlling file system and network access). Docker containers have emerged as the standard isolation mechanism, providing reproducibility and safety without sacrificing the ability to install dependencies and run complex programs.

Web Browsing and Search. Nakano et al. (2022) pioneered web-augmented language models, training GPT-3 to browse the web through a text-based interface supporting search, click, scroll, and quote actions. The resulting system produces long-form answers that humans prefer over purely generative responses, demonstrating the value of grounding generation in retrieved web content. Subsequent systems have expanded web interaction capabilities: WebArena (Zhou et al., 2023b) provides a realistic benchmark for autonomous web agents, while production systems integrate search, browsing, and information synthesis into unified agent workflows.

For research agents specifically, web search provides access to the latest literature, documentation, and data sources that may not exist in the model's training data. The challenge lies in *source evaluation*—determining which retrieved information is reliable and relevant—a capability that current systems handle through heuristics (preferring academic sources, cross-referencing multiple sources) rather than principled approaches.

API and Database Access. Beyond code execution and web browsing, research agents increasingly interact with structured APIs and databases. HuggingGPT (Shen et al., 2023) orchestrates hundreds of ML models through the Hugging Face API, selecting appropriate models for subtasks (image generation, speech recognition, text classification) based on task descriptions. TaskWeaver (Qiao et al., 2023) provides a code-first framework for accessing arbitrary APIs and databases through generated Python code, enabling agents to query SQL databases, call REST APIs, and process structured data.

For scientific research agents, API access extends to domain-specific resources: molecular databases (PubChem, ChEMBL), genomic repositories (NCBI), astronomical catalogs, and laboratory equipment control interfaces (Boiko et al., 2023). The breadth of available tools directly determines the scope of autonomous research that an agent can conduct.

Table 2: Comparison of architecture patterns across key dimensions. Scalability refers to performance as task complexity grows. Cost indicates relative computational expense. Reliability measures consistency of successful outcomes. Generality indicates breadth of applicable domains. Oversight refers to natural human intervention points.

Architecture	Scalability	Cost	Reliability	Generality	Oversight
ReAct (single-pass)	Low	Low	Medium	High	Low
Reflexion (episodic)	Medium	Medium	High	High	Low
LATS / ToT (search)	High	Very High	Very High	Medium	Low
Multi-agent chat	Medium	High	Medium	High	Medium
SOP-based (MetaGPT)	High	Medium	High	Medium	Medium
Debate / adversarial	Medium	High	High	High	Low
Supervisor-worker	High	High	High	High	High
Graph-based	Very High	Very High	Medium	High	Medium

Multi-Modal Tool Use. As foundation models become increasingly multi-modal (Gemini Team, Google, 2023), tool use extends beyond text-based interfaces. Huang et al. (2022) demonstrate that embodied agents benefit from multi-modal feedback (scene descriptions, success signals from vision systems) integrated as textual "inner monologue." For research agents, multi-modal capabilities enable interaction with visual data (plots, microscopy images, molecular structures), auditory signals (spectroscopy data), and spatial information (3D protein structures, circuit layouts).

Tool Discovery and Composition. Schick et al. (2023) show that language models can learn *when* and *how* to invoke tools through self-supervised training, without explicit instruction for each tool. This capability is critical for research agents that must adapt to new tools discovered during investigation. More recent approaches enable agents to *create* new tools: Voyager (Wang et al., 2023a) automatically generates reusable skill functions that are stored in a persistent library, while OS-Copilot (Wu et al., 2024b) creates and accumulates tools through self-improvement. Tool composition—combining multiple tools in novel sequences to achieve complex goals—remains an active research challenge where current systems rely primarily on the planning capabilities of the underlying LLM.

3.5 Comparative Analysis

Table 2 summarizes the key trade-offs across architectural patterns. The choice of architecture depends critically on the target autonomy level, available compute budget, and domain requirements.

Several observations emerge from this comparison. First, *there is no universally superior architecture*—each pattern excels along different dimensions. Single-agent loops with search (LATS, ToT) achieve the highest reliability on well-defined tasks but at prohibitive cost for long-horizon research. Multi-agent systems with structured communication (MetaGPT) offer the best balance of scalability and reliability for software engineering tasks. Hierarchical orchestration provides the strongest human oversight mechanisms, making it preferred for L3 systems where human checkpoints are valued.

Second, *architecture choice should match autonomy level.* L2 systems can operate effectively with simple ReAct loops. L3 systems benefit from Reflexion or supervisor-worker patterns that provide natural checkpoint mechanisms. L4 systems typically require hierarchical orchestration with self-play refinement to maintain quality over extended autonomous operation. The hypothetical L5 level would likely demand graph-based architectures capable of self-reorganization.

Third, *hybrid architectures are increasingly common.* Production systems rarely implement a

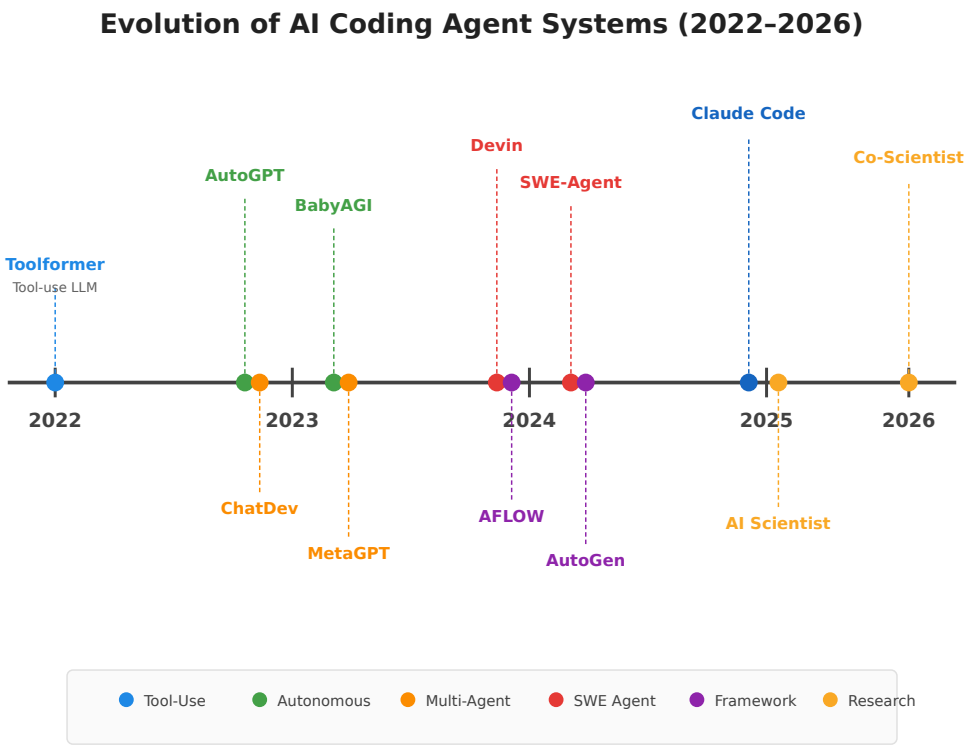

Figure 4: Timeline of major autonomous agent systems (2022–2026), color-coded by primary category. The field has evolved from general-purpose prototypes (2023) through domain-specialized tools (2024) to sophisticated research agents (2025–2026). Systems above the timeline represent open-source projects; those below are proprietary.

single pattern in isolation. Modern code agents combine ReAct-style execution loops with Reflexion-style cross-episode learning, tool-augmented action spaces, and hierarchical task decomposition. The AI Scientist (Lu et al., 2024) combines hierarchical planning (decomposing research into idea generation, experimentation, and writing), tool augmentation (code execution, LaTeX compilation), and adversarial refinement (automated peer review). This convergence toward hybrid architectures suggests that the field is moving beyond paradigm competition toward principled composition of complementary patterns.

4 Key Systems

This section provides a detailed examination of the most significant autonomous agent systems, organized by their primary domain of application. For each system, we analyze the architectural choices, capabilities, evaluation results, and limitations, situating them within our L1–L5 autonomy taxonomy. We conclude with a comprehensive feature comparison matrix (§4.5).

4.1 General-Purpose Autonomous Agents

The earliest wave of autonomous agents (Spring 2023) pursued *domain-general* autonomy: systems designed to accomplish arbitrary goals specified in natural language, without restriction to a particular task domain.

AutoGPT. Richards (2023) introduced AutoGPT, the first widely-adopted autonomous agent system. Given a name, role, and set of goals, AutoGPT operates in a continuous loop: it reasons about the current state, selects an action (web search, code execution, file operations, or spawning sub-agents), executes it, and evaluates progress toward its goals. The system incorporates long-term memory through a vector database, enabling retrieval of relevant past experiences across sessions.

AutoGPT's significance is primarily historical and sociological rather than technical. Within weeks of its release, it became the fastest-growing GitHub repository in history, catalyzing enormous public and research interest in autonomous agents. However, empirical evaluations reveal significant limitations: the system frequently enters *infinite loops* (repeating the same failed actions), exhibits *goal drift* (gradually losing focus on the original objective), and struggles with tasks requiring more than 10–15 sequential actions (Liu et al., 2024b). These failure modes are characteristic of L4 aspiration without L4 capability—the system *attempts* full autonomy but lacks the self-evaluation and error recovery mechanisms needed to sustain it.

BabyAGI. Nakajima (2023) distilled the autonomous agent concept to its minimal form: a three-component system consisting of a *task creation agent* (generating new tasks based on objectives and completed results), a *task prioritization agent* (ordering the task queue by importance), and an *execution agent* (completing individual tasks). This simplicity made BabyAGI influential as a pedagogical tool and architectural template, demonstrating that autonomous behavior emerges from the interaction of planning, prioritization, and execution—even with minimal infrastructure.

BabyAGI's limitation lies in its lack of sophisticated self-evaluation: the system cannot determine when a task has been adequately completed or when its overall approach is failing. This makes it suitable for open-ended exploration (where any progress is valuable) but unreliable for goal-directed research (where specific outcomes are required).

AgentGPT. AgentGPT (Reworkd, 2023) extended the autonomous agent paradigm to a browser-based interface, enabling non-technical users to configure and deploy agents without programming. While architecturally similar to AutoGPT (goal decomposition, sequential task execution), AgentGPT's contribution was *accessibility*: demonstrating that autonomous agent capabilities could be packaged as consumer-facing products. The system operates at L3–L4 on simple tasks (web research, document generation) but degrades to L2 on complex multi-step objectives where human guidance becomes necessary.

Lessons from General-Purpose Agents. The general-purpose agent wave revealed a fundamental tension: *generality and reliability are inversely correlated* in current systems. Systems that attempt to handle any task inevitably encounter situations beyond their competence, leading to cascading failures that unrestricted autonomy amplifies rather than contains. This insight motivated the subsequent shift toward *domain-specific* agents that achieve higher reliability by restricting their scope of operation—a pattern we observe across the research-focused, code-focused, and science-focused systems discussed below.

4.2 Research-Focused Agents

Research-focused agents target the scientific inquiry process itself: literature review, hypothesis generation, experimental design, and manuscript preparation.

The AI Scientist. Lu et al. (2024) present the most ambitious attempt at fully autonomous scientific research to date. The AI Scientist operates an end-to-end pipeline: (1) generating novel

research ideas by combining existing concepts from a seed paper corpus; (2) implementing experiments in code; (3) executing experiments and collecting results; (4) writing a complete scientific paper with figures; and (5) conducting automated peer review to assess quality. The system produces papers at a cost of approximately $15 each, with some generated papers receiving borderline acceptance scores from human reviewers when submitted to peer review.

The AI Scientist operates firmly at L4: given a research area, it autonomously produces complete research artifacts without human intervention. However, its limitations illuminate the gap to L5: it cannot identify *which* research problems are worth pursuing (relying on seed templates), cannot recover from fundamental methodology errors (only surface-level debugging), and produces papers whose novelty, while non-trivial, does not yet match human researcher output. The system demonstrates that autonomous research is *feasible* while simultaneously highlighting how far it remains from human-competitive.

GPT-Researcher. Assafelovic (2023) target a narrower but more reliable form of research autonomy: comprehensive online information synthesis. Given a research question, GPT-Researcher generates a plan of sub-queries, executes parallel web searches, evaluates and filters sources for relevance and reliability, and synthesizes findings into a structured research report. The system achieves substantially higher reliability than general-purpose agents by constraining its scope to information gathering and synthesis (rather than experimental research), operating effectively at L3–L4 within this domain.

STORM. Shao et al. (2024) introduce STORM (Synthesis of Topic Outlines through Retrieval and Multi-perspective question asking), a system for generating comprehensive Wikipedia-style articles from scratch. STORM's key innovation is *perspective-aware question generation*: it simulates interviews with multiple domain experts, each asking questions from their unique viewpoint, to ensure comprehensive coverage of a topic. The system first generates a diverse set of expert personas, then conducts simulated dialogues where these experts query each other and external sources, and finally synthesizes the resulting information into a structured article with proper citations.

STORM demonstrates that multi-perspective reasoning—even when all "perspectives" originate from the same underlying model—produces more comprehensive and balanced research outputs than single-perspective generation. This finding parallels the multi-agent debate literature (Du et al., 2024) and suggests that perspective diversity is a powerful inductive bias for research quality.

Emerging Research Agents. Several recent systems push the boundaries of autonomous research in specific directions. ScienceAgentBench (Zhang et al., 2025b) provides a rigorous evaluation framework with 44 tasks spanning 12 scientific disciplines, enabling standardized comparison of research agents. SciCode (Tian et al., 2024) focuses specifically on scientific coding—the implementation of algorithms from research papers—providing 338 problems that test agents' ability to translate conceptual understanding into working code.

4.3 Code-Focused Agents

Code agents represent the most mature category of autonomous systems, benefiting from the availability of clear evaluation metrics (test passage), rich execution environments (terminals, debuggers, test suites), and large-scale benchmarks (SWE-bench, HumanEval).

SWE-Agent. Yang et al. (2024) introduced SWE-Agent with a focus on *Agent-Computer Interface* (ACI) design—the thesis that the interface between the agent and its computational environment is as important as the agent's reasoning capabilities. SWE-Agent provides a custom

shell environment with commands for file navigation (open, scroll, search), editing (edit with line numbers and automatic linting), and context management (showing the current file state after each edit). This careful interface design yields 12.5% resolution on SWE-bench (2,294 real GitHub issues), demonstrating that environmental affordances—not just model capability—determine agent performance.

The ACI concept has proven foundational: subsequent systems universally adopt purpose-built interfaces rather than exposing raw shell access. The key insight is that the *action space design* serves as an implicit regularizer, preventing agents from taking catastrophic actions (e.g., accidentally deleting repositories) while guiding them toward productive behaviors (e.g., always showing context after edits).

Devin. Cognition Labs' Devin (Cognition Labs, 2024) was announced as "the first AI software engineer," operating in a full development environment with shell, code editor, and web browser. Unlike SWE-Agent's constrained ACI, Devin operates in a comparatively unconstrained sandbox, making architectural decisions, installing dependencies, reading documentation, and iterating on complex multi-file changes. At announcement, Devin reported 13.86% on SWE-bench Lite (a filtered subset of 300 problems), establishing a baseline that subsequent systems have since surpassed.

Devin's primary contribution is demonstrating *end-to-end software engineering autonomy*: rather than solving isolated issues, it can participate in real engineering workflows—from reading Jira tickets through implementation to code review responses. However, as a proprietary system with limited technical disclosure, its architectural details remain partially opaque, making systematic comparison with open-source alternatives challenging.

Claude Code. Anthropic's Claude Code (Anthropic, 2024b) represents a distinct philosophy: an agentic coding tool that operates *within* the developer's terminal, maintaining conversational context while executing multi-step engineering tasks. Its architecture employs a supervisor-worker pattern where the primary agent maintains high-level planning and context, spawning sub-agents for isolated subtasks (file operations, test execution, web search). Key design decisions include running in the user's actual environment (not a sandbox), requiring explicit permission for destructive operations, and maintaining a persistent understanding of repository structure through file exploration rather than indexing.

Claude Code operates at L3 with optional L4 modes: by default, it requests human approval for significant actions (file writes, command execution), but "auto-accept" configurations enable fully autonomous operation within safety boundaries. This configurability reflects a design philosophy that places the autonomy dial in the user's hands, enabling gradual trust calibration.

OpenHands. Wang et al. (2024e) (formerly OpenDevin) provide an open-source platform for building and evaluating code agents. OpenHands implements the CodeAct paradigm (Wang et al., 2024d)—representing all agent actions as executable Python code—within a sandboxed Docker environment. The platform supports multiple agent implementations (CodeAct Agent, browsing agents, delegation agents) and has achieved competitive results on SWE-bench Verified (41–53% depending on underlying model and configuration). Its open-source nature makes OpenHands the de facto standard for academic research on code agents, enabling controlled experiments on agent architecture, prompting strategies, and tool design.

Agentless Approaches. Xia et al. (2024) challenge the assumption that agentic architectures are necessary for software engineering tasks. Their Agentless approach uses a simple two-phase pipeline:

(1) *fault localization* (hierarchically narrowing from repository to file to function to line), followed by (2) *patch generation* (directly producing edits without iterative debugging). Despite lacking the iterative reasoning, tool use, and self-reflection capabilities of full agent systems, Agentless achieves 27% on SWE-bench Lite— competitive with many complex agent systems at a fraction of the computational cost. This result suggests that for a significant fraction of software engineering tasks, the overhead of agentic reasoning provides limited benefit over well-designed direct approaches.

Multi-Agent Code Systems. Recent systems explore multi-agent architectures specifically for code tasks. AgentCoder (Huang et al., 2024a) separates code generation, test case design, and execution verification into three specialized agents, achieving 93.9% on HumanEval through iterative refinement. CodeR (Chen et al., 2024a) further decomposes issue resolution into five roles (manager, reproducer, fault localizer, editor, verifier), each contributing specialized capabilities. MapCoder (Islam et al., 2024) structures the pipeline as recall, plan, code, and debug agents that collaboratively solve competitive programming problems. These multi-agent approaches consistently outperform single-agent systems, validating the benefits of specialization for complex engineering tasks.

4.4 Science Automation Agents

Science automation agents extend the autonomous agent paradigm to empirical scientific research, interfacing with domain-specific tools, databases, and—in some cases—physical laboratory equipment.

Coscientist. Boiko et al. (2023) demonstrate GPT-4-powered autonomous chemical research, where the system independently designs, plans, and executes chemical synthesis experiments using robotic laboratory hardware. Coscientist accesses chemical literature (web search), molecular databases (PubChem), reaction planning tools, and robotic arms for physical execution. In controlled experiments, the system successfully synthesized several target compounds—including aspirin and fluorescent dyes—autonomously, demonstrating end-to-end wet-lab research capability.

Coscientist operates at L4 within its narrow domain: given a synthesis target, it conducts the full research cycle from literature review through physical execution without human intervention. Its significance lies in demonstrating that autonomous research agents can bridge the *digital-physical divide*—a capability previously restricted to hand-engineered robotic pipelines. However, its reliability depends heavily on the safety constraints of its domain: chemical synthesis involves well-characterized reactions with predictable outcomes, unlike the open-ended exploration required for genuinely novel research.

ChemCrow. Bran et al. (2024) augment an LLM with 18 chemistry-specific tools spanning synthesis planning (RXN4Chemistry), safety assessment (MSDS lookup), molecular property prediction (RDKit), and patent search. Unlike Coscientist's focus on execution, ChemCrow emphasizes *reasoning over chemical knowledge*: combining tools to answer complex questions about synthesis feasibility, safety concerns, and molecular design. Expert evaluation indicates that ChemCrow produces correct answers for 75% of complex chemistry questions, compared to less than 30% for unaugmented GPT-4, demonstrating the transformative impact of domain-specific tool integration.

BioPlanner. Moor et al. (2023) evaluate LLM capabilities for biological experimental protocol generation—a fundamental task in wet-lab research. BioPlanner benchmarks the ability to generate step-by-step protocols given high-level experimental objectives, evaluating correctness,

completeness, and safety considerations. Results indicate that while current LLMs can generate plausible-sounding protocols, they frequently omit critical safety steps, use incorrect reagent concentrations, or propose physically impossible procedures—highlighting the gap between text fluency and domain expertise.

FunSearch. Romera-Paredes et al. (2024) represent a distinct paradigm for scientific discovery: LLM-guided evolutionary program search. Rather than using an agent loop, FunSearch maintains a population of programs (solutions to a mathematical problem), uses an LLM to propose mutations and combinations, evaluates candidates against an objective function, and selects the fittest for the next generation. This approach discovered new mathematical constructions surpassing previously known bounds for the cap set problem and bin packing—genuine contributions to mathematical knowledge.

FunSearch is notable for achieving *verifiable novelty*: because its outputs are programs with measurable performance, claims of discovery can be mechanically verified rather than requiring subjective human judgment. This property makes it one of the few autonomous systems whose research contributions are unambiguous. However, its scope is limited to problems expressible as program optimization, excluding the broader scientific inquiry that requires hypothesis formulation, experimental design, and qualitative interpretation.

Domain-Specific Medical and Scientific Agents. MedAgents (Tang et al., 2024) demonstrates multi-agent collaboration for medical reasoning, assigning specialist roles (cardiologist, oncologist, radiologist) to different agents that deliberate on clinical cases. SciAgent (Chen et al., 2024b) integrates computation, data analysis, and domain-specific tools for general scientific reasoning tasks. These systems illustrate how the autonomous agent paradigm adapts to high-stakes domains where reliability and safety requirements exceed those of general-purpose systems.

4.5 Feature Comparison Matrix

Table 3 provides a comprehensive comparison of the systems discussed above across six dimensions: autonomy level (from our L1–L5 taxonomy), primary domain, architectural pattern, tool integration breadth, evaluation methodology, and availability (open-source vs. proprietary).

Several patterns emerge from this comparison. First, *code agents achieve the highest performance* across all categories, benefiting from the availability of automated evaluation (test suites), rich tool environments (terminals, debuggers), and large-scale benchmarks. Second, *open-source systems now compete with proprietary ones*: OpenHands achieves results within striking distance of closed-source systems like Devin, enabling academic research on agent architectures. Third, *domain-specific agents outperform general-purpose ones*: the most reliable L4 systems (SWE-Agent, Coscientist, FunSearch) succeed by restricting their scope, while general-purpose agents (AutoGPT, BabyAGI) struggle to achieve consistent L4 operation across diverse tasks.

The progression from left to right in Table 3 also traces a historical arc: from the early general-purpose systems (2023 Q1–Q2) through domain-specialized agents (2023 Q3–2024 Q2) to sophisticated multi-agent architectures (2024 Q3–present). This trajectory suggests a maturation pattern common in software engineering—from ambitious but fragile prototypes toward robust, well-scoped systems with clear evaluation methodology.

A notable gap in the landscape is the absence of *cross-domain* research agents that combine the code execution capabilities of SWE-Agent, the literature synthesis of STORM, and the experimental rigor of Coscientist into a unified system. The AI Scientist (Lu et al., 2024) represents the closest approximation, but even it operates within a single research paradigm (ML experimentation) rather

Table 3: Feature comparison of major autonomous agent systems. Autonomy levels follow the taxonomy from §2.2. Architecture abbreviations: SA = single-agent loop, MA = multi-agent, H = hierarchical, Evo = evolutionary. Evaluation: task-specific benchmarks and reported performance where available.

System	Level	Arch.	Domain	Tools	Evaluation	Open
AutoGPT	L3–L4	SA	General	Web, code, files	AgentBench	✓
BabyAGI	L3	SA	General	Web, memory	Qualitative	✓
AgentGPT	L3	SA	General	Web, code	Qualitative	✓
AI Scientist	L4	H+MA	Research	Code, LaTeX, review	Human eval, auto-review	✓
GPT-Researcher	L3–L4	SA	Research	Web search, synthesis	Report quality	✓
STORM	L3	MA	Research	Web, multi-perspective	Human eval (Wikipedia)	✓
SWE-Agent	L4	SA	Code	Custom ACI shell	SWE-bench: 12.5%	✓
Devin	L4	H	Code	Shell, editor, browser	SWE-bench Lite: 13.9%	×
Claude Code	L3–L4	H	Code	Terminal, files, web	SWE-bench Verified: 72%	×
OpenHands	L4	SA/MA	Code	Docker, browser	SWE-bench Verified: 53%	✓
Agentless	L2–L3	—	Code	Code gen only	SWE-bench Lite: 27%	✓
AgentCoder	L3	MA	Code	Execution, testing	HumanEval: 93.9%	✓
AutoCodeRover	L4	SA	Code	Code search, edit	SWE-bench Lite: 30.7%	✓
Coscientist	L4	SA	Chemistry	Robotic lab, databases	Synthesis success	×
ChemCrow	L3	SA	Chemistry	18 chem tools	Expert eval: 75%	✓
BioPlanner	L2–L3	SA	Biology	Literature	Protocol correctness	✓
FunSearch	L4	Evo	Mathematics	Program evaluation	Verified discoveries	×

than adapting to arbitrary scientific domains. Achieving truly general L4–L5 research autonomy—where a single system can conduct novel research across disciplines—remains an open challenge that likely requires advances in both foundation model capabilities and agent architecture design.

5 Evaluation and Benchmarks

Evaluating autonomous research agents presents unique challenges that go far beyond traditional software testing. Unlike constrained systems with clear success criteria, research agents operate in open-ended domains where quality is multi-dimensional and often subjective. This section surveys the evaluation landscape: task-completion benchmarks (§5.1), research quality metrics (§5.2), efficiency considerations (§5.3), safety evaluation requirements (§5.4), and the fundamental difficulties of assessing open-ended research (§5.5).

5.1 Task-Completion Benchmarks

The most mature evaluation methodology for autonomous agents relies on *task-completion benchmarks*: curated sets of problems with verifiable solutions that measure an agent's ability to achieve concrete objectives.

SWE-bench. Jimenez et al. (2024) introduced SWE-bench, the de facto standard for evaluating code agents. The benchmark comprises 2,294 real GitHub issues from 12 popular Python repositories (Django, scikit-learn, matplotlib, etc.), each paired with a ground-truth patch and test cases that verify correctness. Agents must navigate large codebases, understand issue descriptions, and produce functional patches—testing the full spectrum of software engineering capabilities from comprehension to implementation.

SWE-bench's strength lies in its ecological validity: tasks are drawn from actual developer workflows rather than synthetic problems. However, concerns about noisy instances (ambiguous issues, incorrect tests) led to the creation of *SWE-bench Verified*, a human-validated subset of 500 problems that provides more reliable signal. The benchmark has catalyzed rapid progress: resolution rates

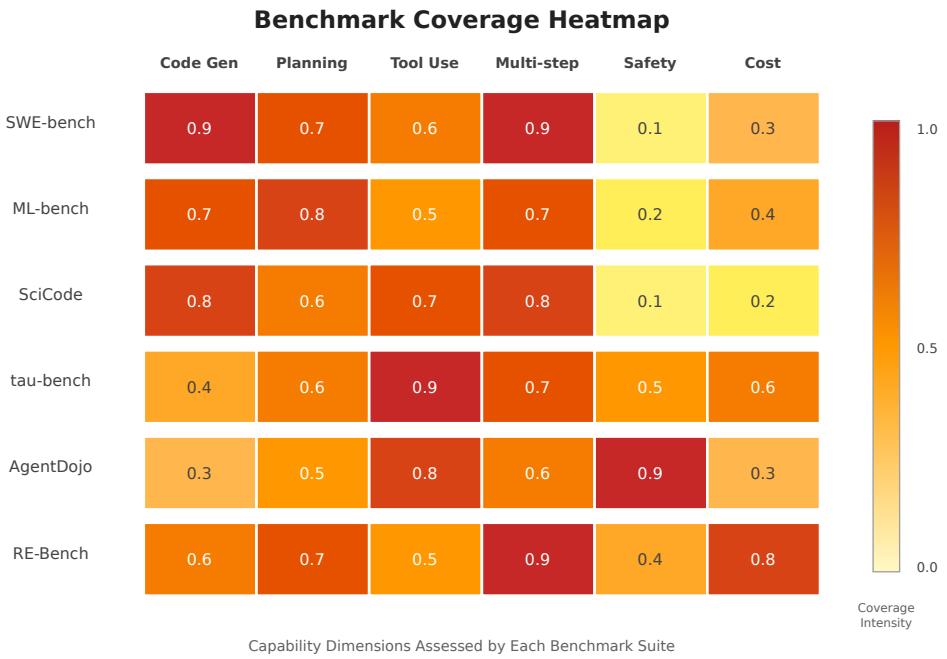

Figure 5: Coverage heatmap of major benchmarks across six evaluation dimensions. Darker cells indicate stronger coverage. Current benchmarks excel at code generation and multi-step evaluation but provide limited assessment of safety and cost efficiency—critical dimensions for real-world deployment.

have climbed from 3.8% (naive baselines) through 12.5% (SWE-Agent; Yang et al., 2024) to over 70% (Claude Code on Verified; Anthropic, 2024b) within 18 months, though direct comparisons across subsets require careful interpretation.

HumanEval and MBPP. Chen et al. (2021) introduced HumanEval, a benchmark of 164 hand-crafted Python programming problems, each specified by a function signature, docstring, and unit tests. While originally designed to evaluate code generation models, HumanEval has become a standard evaluation for agent systems that employ code generation as a core capability. MBPP (Mostly Basic Python Problems; Austin et al., 2021) provides a complementary set of 974 crowd-sourced problems at a lower difficulty level.

These benchmarks are approaching saturation: state-of-the-art models now exceed 90% pass@1 on both, limiting their discriminative power for advanced systems. Liu et al. (2024a) address this through EvalPlus, which augments HumanEval and MBPP with 80× more test cases, revealing that many ostensibly correct solutions fail on edge cases—pass rates typically drop 10–20% under rigorous evaluation. LiveCodeBench (Jain et al., 2024) provides contamination-free evaluation by continuously collecting new problems from competitive programming platforms.

MATH and GSM8K. Mathematical reasoning benchmarks evaluate a capability crucial to research agents: rigorous logical deduction. The MATH dataset (Hendrycks et al., 2021) comprises 12,500 competition-level problems spanning seven subject areas, while GSM8K (Cobbe et al., 2021) provides 8,500 grade school problems testing multi-step arithmetic reasoning. Together, they assess

reasoning from elementary to advanced levels. Recent models achieve near-human performance on GSM8K (>95%) but MATH remains challenging at competition difficulty levels, making it a more discriminating evaluation for research agent reasoning capabilities.

AgentBench. Liu et al. (2024b) provide a multi-environment evaluation framework spanning eight distinct interactive environments: operating systems, databases, knowledge graphs, digital card games, lateral thinking puzzles, house-holding tasks, web browsing, and web shopping. AgentBench evaluates the breadth of agent capabilities rather than depth in any single domain, revealing that performance varies dramatically across environments—models excelling at code tasks may struggle with embodied reasoning and vice versa. This highlights the importance of multi-dimensional evaluation that captures the heterogeneous requirements of research autonomy. More recently, τ-bench (Yao et al., 2024) evaluates tool-agent-user interactions in realistic domains such as airline and retail customer service.

GAIA. Mialon et al. (2023) introduce GAIA (General AI Assistants), a benchmark of 466 questions designed to evaluate general-purpose AI assistants. GAIA questions require multi-step reasoning, tool use (web search, code execution, file manipulation), and integration of information from multiple sources. Critically, GAIA questions have unambiguous factual answers, enabling automated evaluation despite requiring complex agent behavior to solve. The benchmark stratifies tasks by difficulty (Levels 1–3), with even the most capable systems solving fewer than 50% of Level 3 questions, providing headroom for future progress.

MLAgentBench. Huang et al. (2023) specifically target the evaluation of AI research capabilities through 13 tasks requiring agents to train, debug, and improve machine learning models. Tasks range from implementing known algorithms to optimizing existing codebases, providing a bridge between code benchmarks and open-ended research. Performance varies widely across tasks, with agents succeeding on well-documented problems but struggling with tasks requiring creative problem-solving or novel algorithmic insights.

5.2 Research Quality Metrics

While task-completion benchmarks evaluate the *execution* component of research, they do not address the quality of *research contributions* themselves. Evaluating the novelty, correctness, and significance of AI-generated research requires fundamentally different approaches.

Human Evaluation Protocols. The gold standard for research quality assessment remains expert human evaluation. Lu et al. (2024) employ automated peer review using NeurIPS review criteria (soundness, significance, novelty, clarity), finding that AI Scientist papers receive scores in the borderline range. However, human evaluation faces scalability challenges: expert reviewer time is expensive and scarce, inter-reviewer agreement is notoriously low (below 50% in typical ML venues; Kapoor et al., 2024), and evaluation criteria are often implicit rather than formalized. These limitations motivate the development of automated proxies.

Automated Quality Proxies. Several automated metrics have been proposed as proxies for research quality: (1) *citation prediction*—estimating the likely impact of a paper based on its content and positioning relative to the literature; (2) *reproducibility checks*—verifying that code runs, experiments produce claimed results, and conclusions follow from data; (3) *novelty detection*—measuring the semantic distance between generated work and the existing corpus using embedding-

based similarity. Each proxy captures one dimension of quality but none individually suffices. The most promising approach combines multiple signals: FunSearch (Romera-Paredes et al., 2024) sidesteps the problem entirely by restricting to domains with *mechanical verifiability* (programs with measurable performance), but this approach does not generalize to qualitative or theoretical research.

Benchmark Validity and Contamination. Kapoor et al. (2024) raise critical concerns about evaluation practices in the agent community. They identify systematic issues including benchmark overfitting (optimizing for specific benchmarks without genuine capability improvement), inadequate cost reporting (omitting computational expense that accompanies accuracy gains), and lack of holdout evaluation sets. These issues parallel longstanding concerns in ML evaluation but are amplified for agents because their complex, multi-step behavior makes it easier to inadvertently overfit to benchmark-specific patterns.

Fundamental Limitations of Benchmarks for Research. An important caveat applies to the entire benchmarking enterprise: existing benchmarks measure *task execution competence* rather than *genuine research capability*. Solving curated problems with known solutions—however complex—is fundamentally different from identifying novel research questions, designing experiments under uncertainty, or producing insights that advance collective knowledge. No current benchmark captures the creative, serendipitous, and socially-embedded aspects of scientific discovery. Until evaluation methodology catches up with system capabilities, benchmark performance should be interpreted as a necessary but not sufficient condition for research agent autonomy.

5.3 Efficiency Metrics

For autonomous research agents to be practical, they must achieve their objectives within acceptable resource budgets. Efficiency evaluation encompasses multiple dimensions: monetary cost, computational resources, and time.

Cost per Task. The most direct efficiency metric is the monetary cost per successfully completed task, combining token usage (input and output), API calls, and tool invocations. Published figures vary dramatically across systems and tasks: the AI Scientist produces complete papers at approximately $15 each (Lu et al., 2024), while complex SWE-bench resolutions can cost $5–50 depending on the agent architecture and required iteration depth. Reporting cost alongside accuracy is essential but frequently omitted—a practice that Kapoor et al. (2024) identify as a systemic weakness in agent evaluation.

Success Rate vs. Compute Trade-offs. A more nuanced analysis examines the *Pareto frontier* between success rate and computational investment. Zhou et al. (2023a) demonstrate that LATS achieves 94.4% on HumanEval but at 5–20× the cost of single-pass approaches. Similarly, the jump from single-agent to multi-agent architectures typically incurs 2–5× cost increases for 5–15% accuracy gains. The Agentless approach (Xia et al., 2024) occupies a particularly interesting position on this frontier: achieving competitive accuracy at a fraction of the cost of full agent systems, demonstrating that the relationship between agentic complexity and performance is non-monotonic.

Scaling Behavior. An important but understudied question is how agent costs scale with task complexity. For well-structured tasks (bounded codebase, clear specification), costs are relatively

predictable. For open-ended research, costs can grow superlinearly as the agent explores dead ends, backtracks, and iterates. Understanding these scaling properties is critical for deploying research agents on tasks where the difficulty is unknown a priori.

5.4 Safety Considerations in Evaluation

The evaluation of autonomous research agents introduces safety concerns that do not arise in traditional software benchmarks. As agents gain the ability to execute code, browse the web, and interact with external systems, their evaluation must account for potential harms.

Sandboxing Requirements. Safe evaluation of autonomous agents demands robust isolation. Agents must be prevented from: modifying production systems, exfiltrating sensitive data, consuming unbounded resources, or persisting changes beyond the evaluation context. Docker-based sandboxing has become standard (Yang et al., 2024; Wang et al., 2024e), but the tension between realistic environments (which enable full evaluation) and safety constraints (which limit agent capabilities) remains unresolved. Kinniment et al. (2024) evaluate agents on realistic autonomous tasks specifically designed to test whether agents attempt to exceed their permitted scope.

Dual-Use Risks. Autonomous research agents capable of conducting novel scientific investigation present inherent dual-use risks. A system that can autonomously design chemical syntheses (Boiko et al., 2023) could potentially be directed toward hazardous compounds. Agents that write exploit code for SWE-bench resolutions could be repurposed for offensive cybersecurity. Shevlane et al. (2023) argue for *model evaluation for extreme risks*, including autonomous replication, persuasion, and weapons development. Responsible evaluation frameworks must assess these capabilities without enabling them—a challenging balance that current practices handle primarily through access controls and responsible disclosure rather than technical solutions.

Evaluation of Harmful Outputs. Beyond capability assessment, evaluation must detect when agents produce harmful outputs: generating dangerous information, reinforcing biases in research conclusions, or producing misleading results that could propagate through the scientific literature. Chan et al. (2023) provide a taxonomy of harms from increasingly agentic systems, spanning direct harms (dangerous actions), representational harms (biased outputs), and systemic harms (undermining trust in scientific institutions). Evaluation frameworks for research agents should incorporate checks along each dimension.

5.5 Challenges in Evaluating Open-Ended Research

The most fundamental evaluation challenge for autonomous research agents is that their target domain—scientific research—lacks ground truth. Unlike code benchmarks with test suites or math problems with definitive answers, research quality is inherently subjective and temporally contingent.

Absence of Ground Truth. For creative research tasks, there is no predetermined correct answer against which to evaluate. A system that generates a novel hypothesis, designs an experiment, and produces results is not "right" or "wrong" in the same sense as a system solving a coding bug. Evaluation must instead assess properties like *interestingness* (does the research reveal something unexpected?), *soundness* (does the methodology support the conclusions?), and *significance* (does it advance the field?). Each of these properties resists reliable automation and exhibits substantial human disagreement.

Measuring "Interestingness." Research impact is determined by the community over time—high-impact work is often initially controversial or niche. No current metric reliably predicts which contributions will prove significant. The AI Scientist's automated reviewer (Lu et al., 2024) can assess technical soundness but not the "taste" dimension that distinguishes incremental work from paradigm-shifting contributions. This limitation is not merely technical but epistemological: interestingness is a social construct that emerges from community engagement rather than objective measurement.

Time Horizon Mismatch. Current benchmarks evaluate agent performance on tasks completable in minutes to hours. Real research unfolds over weeks, months, or years. This temporal mismatch means that benchmarks test agent capabilities on *subtasks* of research (writing code, running experiments, synthesizing literature) rather than the full research arc (identifying important problems, building on prior results, developing long-term programs). Evaluation frameworks that bridge this gap—perhaps through multi-stage benchmarks with dependencies between stages—remain an important open direction.

Toward Comprehensive Evaluation Frameworks. We identify three priorities for advancing research agent evaluation: (1) *multi-dimensional metrics* that jointly assess novelty, correctness, efficiency, and safety rather than optimizing any single dimension; (2) *longitudinal evaluation* that tracks agent performance across extended research campaigns rather than isolated tasks; and (3) *community-based assessment* that incorporates expert feedback loops into the evaluation process. Progress on these fronts will require collaboration between the AI agent community and domain scientists who can provide authoritative quality judgments.

6 Challenges and Open Problems

Despite rapid progress, autonomous research agents face fundamental challenges that limit their reliability, scope, and safety. This section identifies six core problems that define the current frontier, analyzing each through the lens of existing work and proposing concrete research directions. These challenges are not merely engineering obstacles but reflect deep tensions between autonomy and controllability, generality and reliability, and creativity and correctness.

6.1 The Cognitive Loop Trap

A pervasive failure mode of autonomous agents is the *cognitive loop trap*: the agent becomes stuck repeating ineffective strategies without recognizing that its approach is failing. This manifests as literal action repetition (executing the same failing command), semantic cycling (trying variations of the same strategy), or escalating complexity without progress (adding layers of abstraction that obscure rather than solve the problem).

Prevalence and Causes. The cognitive loop trap is especially common at L4 autonomy, where the agent operates without human checkpoints that might break the cycle. AutoGPT's well-documented tendency to enter infinite loops (Richards, 2023) represents the most visible instance, but subtler forms affect all current systems. Root causes include: (1) *limited state tracking*—the agent cannot represent "I have already tried this and it failed" in a way that reliably prevents repetition; (2) *context window saturation*—as prior attempts fill the context, the agent loses access to early observations that would inform strategy changes; and (3) *exploration-exploitation imbalance*—agents default to exploiting known (but failing) approaches rather than exploring fundamentally different strategies.

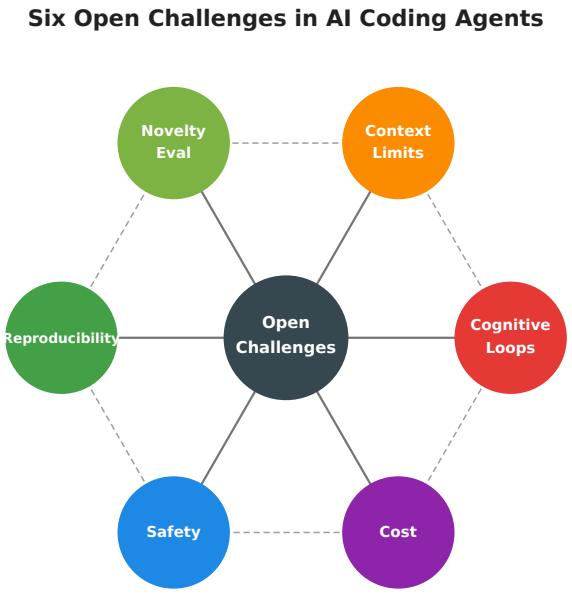

Figure 6: The six fundamental open challenges for autonomous research agents and their interconnections. Cognitive loops and context limitations are architectural constraints; novelty evaluation and reproducibility are epistemological challenges; safety and cost are deployment barriers. Dotted lines indicate mutual dependencies between challenges.

Anti-Loop Mechanisms. Several mitigation strategies have been proposed. Reflexion (Shinn et al., 2023) maintains explicit verbal records of failed strategies, reducing but not eliminating repetition. Fresh-session approaches—discarding accumulated context and restarting with a summary—prevent context saturation but sacrifice detailed memory. Diversity injection mechanisms force the agent to try qualitatively different approaches after a fixed number of failures, trading optimality for exploration. Zhang et al. (2025a) suggest that graph-based orchestration can structurally prevent loops by making the workflow topology acyclic. However, no current mechanism provides a principled, general solution; most rely on heuristics (failure counters, diversity thresholds) that require task-specific tuning.

Research Directions. Promising approaches include: formal verification of agent loops using temporal logic specifications; learned termination criteria that predict when an approach is irrecoverable; and adversarial monitoring agents that detect cycling in the primary agent's behavior. The connection to reinforcement learning's exploration-exploitation dilemma suggests that count-based exploration bonuses or curiosity-driven mechanisms (Wang et al., 2023a) could provide theoretical grounding for anti-loop strategies.

6.2 Context Window Limitations

Current language models operate within fixed context windows (4K–1M tokens) that constrain the temporal horizon of autonomous operation. Research tasks requiring integration of information across thousands of source documents, extensive experimental histories, or long chains of reasoning quickly exceed these limits.

The Memory Bottleneck. A research agent investigating a complex topic may need to simultaneously maintain: the current experimental state, results from prior experiments, relevant literature passages, code implementations, and strategic plans. At L4 autonomy, a single research session can easily generate 100K+ tokens of observations and actions. When this history exceeds the context window, the agent loses access to early (potentially crucial) information, leading to repeated work, contradictory conclusions, or abandoned promising approaches.

Compaction and Summarization. Packer et al. (2023) propose a virtual memory hierarchy that pages information between active context and external storage, analogous to OS memory management. Practical systems employ progressive summarization: as history grows, older entries are compressed into increasingly abstract summaries, preserving key decisions while discarding implementation details. However, summarization is lossy—the agent may compress away precisely the detail that proves relevant later. Determining what to preserve and what to compress requires judgment that current systems lack.

Hierarchical Decomposition. An architectural response to context limitations is hierarchical task decomposition: a supervisor agent maintains high-level state (fitting within its context), delegating subtasks to worker agents that operate in isolated contexts (Anthropic, 2024b). This approach effectively multiplies available context by distributing it across agents, but introduces coordination overhead and potential information loss at boundaries between agents. The optimal decomposition granularity—how much context each agent should maintain versus pass to subagents—remains an empirical question without theoretical guidance.

Research Directions. Key open problems include: principled information-theoretic approaches to context compaction that minimize expected regret; learned retrieval mechanisms that predict which historical information will be relevant for future decisions; and architectural innovations (state-space models, persistent memory networks) that scale sub-quadratically with history length while maintaining the reasoning quality of attention-based models.

6.3 Evaluation of Novel Contributions

As discussed in §5.5, evaluating the novelty and significance of AI-generated research remains fundamentally unsolved. This challenge is not merely practical (how to build better evaluation metrics) but philosophical (what constitutes a genuine intellectual contribution).

The Human Evaluation Bottleneck. Expert evaluation is the gold standard but does not scale. A research agent producing dozens of papers per day cannot rely on human reviewers for quality assessment. Moreover, human evaluation of novelty is itself unreliable: reviewer agreement at top ML conferences remains below 50% (Kapoor et al., 2024), and genuinely novel work is often initially rejected precisely because it challenges reviewers' priors. Using human evaluation as the training signal for automated metrics thus propagates the biases and inconsistencies of human judgment.

Automated Novelty Detection. Embedding-based approaches measure novelty as distance from the existing literature in a learned semantic space. However, these approaches conflate novelty with obscurity: work that uses unusual terminology scores as "novel" regardless of its intellectual contribution. Citation prediction models attempt to forecast impact but are confounded by social

dynamics (prominent authors receive more citations regardless of content) and cannot evaluate contributions that create entirely new subfields.

Research Directions. We identify three promising approaches: (1) *verification-based evaluation*—restricting to domains where contributions can be mechanically verified (following FunSearch's approach; Romera-Paredes et al., 2024); (2) *adversarial evaluation*—using separate agents to attempt falsification, reproduction failure, or identification of prior work that subsumes the claimed contribution; and (3) *portfolio evaluation*— assessing research programs (collections of related outputs) rather than individual contributions, measuring coherence and cumulative progress over time.

6.4 Reproducibility and Determinism

Scientific progress depends on reproducibility: the ability for independent researchers to verify claims by repeating experiments. Autonomous research agents introduce new reproducibility challenges that compound those already afflicting computational science.

Non-Deterministic Outputs. Language model inference with non-zero temperature produces different outputs on each run. Two identical invocations of a research agent may follow entirely different reasoning paths, explore different hypotheses, and arrive at different conclusions. This stochasticity makes it difficult to verify claims about agent capabilities: reported performance numbers may reflect lucky samples rather than reliable behavior. Kapoor et al. (2024) show that agent benchmark results exhibit high variance across runs, with standard deviations of 5–15% on SWE-bench depending on the system.

Sensitivity to Prompting. Agent behavior is highly sensitive to seemingly minor prompt variations: changes in system prompts, tool descriptions, or even whitespace can produce qualitatively different research trajectories. This sensitivity undermines the scientific standard of methodological transparency—describing the agent's "method" requires specifying the exact prompt text, model version, temperature, and tool configuration, creating a reproducibility burden that few papers fully discharge. The absence of standardized reporting protocols exacerbates this problem.

Version Dependence. As foundation models are updated (GPT-4 to GPT-4 Turbo to GPT-4o; Claude 3 to Claude 3.5 to Claude 4), agent behaviors change in unpredictable ways. Research conducted with one model version may not reproduce with its successor. This creates a moving target for evaluation: benchmark results become obsolete as models evolve, and longitudinal comparison across agent generations requires careful version pinning that is rarely maintained.

Research Directions. Potential solutions include: standardized agent "cards" that fully specify execution configurations; robustness evaluation that measures performance variance across random seeds, prompt paraphrases, and model versions; and deterministic inference modes that sacrifice diversity for reproducibility in evaluation contexts.

6.5 Safety and Ethics

The capabilities that make autonomous research agents valuable—independent investigation, tool use, creative problem-solving—simultaneously create risks. As agents approach L4–L5 autonomy, safety concerns transition from theoretical to practical.

Dual-Use Concerns. Research agents capable of autonomous scientific discovery inevitably face the dual-use dilemma: the same system that designs beneficial drug candidates could propose toxic compounds; the agent that discovers mathematical proofs could optimize adversarial algorithms. Boiko et al. (2023) demonstrate this concretely: their chemistry agent could in principle be directed toward dangerous syntheses, and relies on safety filters rather than fundamental architectural constraints to prevent this. The challenge is not merely adding safety filters (which can be circumvented) but designing architectures where dangerous capability and beneficial capability are separable—a goal that may be fundamentally impossible for sufficiently general systems.

Uncontrolled Self-Improvement. At L5 autonomy, agents that select their own research directions could choose to research improvements to their own capabilities. While current systems are far from this scenario, the trajectory toward more autonomous agents makes it a legitimate long-term concern. Shevlane et al. (2023) identify autonomous self-replication and resource acquisition as key dangerous capabilities to evaluate. Frameworks for *corrigibility*— ensuring agents remain amenable to human correction even as they become more capable—are an active area of alignment research with direct implications for research agent design.

Responsible Deployment Frameworks. OpenAI (2024a) propose practices for governing agentic AI systems, including: maintaining human oversight at critical decision points, limiting agent capabilities to those required for the task, implementing robust monitoring and logging, and establishing clear accountability structures. For research agents specifically, we argue that deployment should follow a *graduated autonomy* model: systems earn increased autonomy through demonstrated reliable and safe behavior, with automatic scope reduction when safety boundaries are approached. This mirrors the L1–L5 taxonomy not as a fixed classification but as a dynamic dial adjusted based on empirical trust.

Research Directions. Critical open problems include: formal frameworks for reasoning about the safety of autonomous research (extending work on AI alignment to the agent setting); technical mechanisms for capability separation (preventing beneficial research tools from being repurposed for harm); and governance structures that enable beneficial research agent deployment while managing societal risks. The absence of international coordination frameworks for autonomous research AI—analogous to biosafety level classifications— represents a significant governance gap.

6.6 Cost and Accessibility

The computational cost of autonomous research agents creates economic barriers that affect both the development and deployment of these systems, with implications for equity and scientific progress.

Token Economics. Long-running research agents consume substantial token budgets. A single SWE-bench resolution may cost \$5–50 in API calls; a full research campaign (literature review, hypothesis generation, experimentation, writing) could easily consume \$100–1000 in tokens alone. These costs are prohibitive for researchers at less-resourced institutions, creating a risk that autonomous research becomes a privilege of wealthy organizations. While token costs have decreased substantially (10–100$\times$ reductions between 2023 and 2025), they remain significant for the extended, iterative operation that research agents require.

The Open-Source vs. Proprietary Divide. The most capable foundation models remain proprietary (GPT-4, Claude, Gemini), while open-source alternatives (Llama, Mistral, DeepSeek) have narrowed but not closed the capability gap for agentic tasks. This creates a bifurcated landscape: the most reliable autonomous research agents require expensive proprietary APIs, while open-source alternatives sacrifice reliability for accessibility. Wang et al. (2024e) demonstrate that open-source agents can achieve competitive performance, but the gap on the most challenging tasks (particularly those requiring creative problem-solving and long-horizon planning) remains significant.

Inequality of Access. The combination of high costs and proprietary model dependence risks concentrating autonomous research capabilities among well-funded institutions, potentially exacerbating existing inequalities in scientific output. If research agents substantially accelerate productivity, institutions without access will fall further behind. This concern motivates both technical solutions (more efficient architectures, cost-effective model routing) and policy interventions (subsidized access for academic researchers, public infrastructure for agent evaluation).

Research Directions. Key opportunities include: architectures that achieve high reliability with smaller, cheaper models through better tool design and scaffolding; adaptive compute allocation that spends more tokens on difficult problems and fewer on routine tasks; and shared infrastructure (benchmarks, execution environments, evaluation servers) that reduces the overhead of conducting research on research agents. The democratization of agent capabilities depends critically on closing the gap between open-source and proprietary systems on agentic tasks.

7 Future Directions

The preceding sections have established where autonomous research agents stand today: capable L3–L4 systems constrained by finite context, brittle evaluation, and limited self-awareness. This section outlines five research directions that, if pursued successfully, could enable the transition to robust L5 autonomy—agents that not only execute research tasks but set their own agendas and improve over time.

7.1 Self-Improving Agents

Current agents are stateless across sessions: each invocation begins tabula rasa, repeating mistakes that a learning system would have corrected. A critical frontier is *meta-learning from past runs*—agents that update their own strategies, prompts, and tool-usage patterns based on accumulated experience.

Early work in this direction is promising. ExpeL (Zhao et al., 2024) demonstrates that LLM agents can extract reusable "insights" from successful and failed trajectories without gradient updates, storing them as natural-language rules for future retrieval. Reflexion (Shinn et al., 2023) implements a simpler form of verbal self-improvement within a single episode. More ambitiously, ADAS (Hu et al., 2024) proposes *automated design of agentic systems*, where a meta-agent searches over the space of agent architectures, prompts, and tool configurations, discovering designs that outperform hand-crafted alternatives.

The connection to classical online learning is underexplored. Regret-bounded algorithms from the multi-armed bandit literature could provide principled exploration-exploitation trade-offs for agents choosing among strategies. Continual learning methods (Wang et al., 2024c) address the catastrophic forgetting problem that arises when agents update their policies across diverse research domains. Key research questions include:

- What is the right *granularity* of experience to store—individual actions, episode summaries, or abstract heuristics?
- How should agents balance fidelity to past experience against adaptation to novel problem structures?
- Can self-improvement be made *safe*—i.e., can we guarantee that learned modifications do not degrade performance or violate safety constraints?

7.2 Persistent Knowledge Across Sessions

The ephemeral context window is the single greatest architectural bottleneck for sustained research programs. Current agents cannot remember what they discovered yesterday, recognize recurring patterns across projects, or build cumulative understanding of a research domain. Solving this requires moving beyond retrieval-augmented generation (Lewis et al., 2020) toward *structured, evolving knowledge representations*.

Three complementary approaches deserve investigation. First, *external knowledge graphs* that agents construct and query, enabling relational reasoning over discovered facts, hypotheses, and their supporting evidence. Unlike flat vector stores, knowledge graphs preserve the logical structure necessary for scientific reasoning—causality, contradiction, temporal ordering, and evidential support.

Second, *learned retrieval policies* that determine not just what information to store but when and how to retrieve it. MemGPT (Packer et al., 2023) takes initial steps by applying OS virtual memory concepts to LLM context management, but the retrieval policy remains hand-crafted. Future systems should learn retrieval strategies from task performance feedback.

Third, *shared knowledge bases across agent populations*, enabling discoveries by one agent instance to benefit others. The AI co-scientist (Google DeepMind, 2025) demonstrates how multi-agent systems can maintain shared hypothesis pools, but the challenge of knowledge curation—what to keep, what to discard, how to resolve contradictions—remains largely unaddressed.

7.3 Human-AI Collaborative Research Teams

The framing of "fully autonomous" research agents may be a false summit. The most productive near-term configuration is likely *mixed teams* where AI handles high-throughput exploration and humans provide judgment, creativity, and ethical oversight. This requires new interaction paradigms beyond the current "prompt and wait" model.

Generative agents (Park et al., 2023) demonstrated that LLM-powered agents can maintain coherent roles and social dynamics. Extending this to research contexts suggests architectures where multiple AI agents and human researchers interact through shared workspaces, asynchronous communication channels, and structured deliberation protocols. AutoGen (Wu et al., 2024a) provides a multi-agent conversation framework, but its human-in-the-loop mode is rudimentary—limited to approval gates rather than genuine collaboration.

Critical research questions for this direction include: How should agents signal uncertainty to human collaborators? What is the optimal *division of cognitive labor*—which research subtasks benefit most from AI speed versus human judgment? How can we design interaction protocols that preserve human agency and understanding while exploiting AI throughput? The literature on mixed-initiative interaction and computer-supported cooperative work (CSCW) provides foundations, but the specific challenges of *research* collaboration (novelty assessment, methodological rigor, intellectual attribution) require domain-specific solutions.

7.4 Domain-Specific Autonomous Labs

The most immediate path to L5 autonomy may not be general-purpose research agents but *domain-specific closed-loop laboratories* where AI systems control the complete experimental cycle: hypothesis generation, experimental design, physical execution via robotic hardware, data analysis, and iterative refinement.

Coscientist (Boiko et al., 2023) demonstrated autonomous chemical synthesis with GPT-4 controlling robotic hardware. Self-driving laboratories (Tom et al., 2024) in materials science have achieved closed-loop optimization of material properties with minimal human intervention. These systems operate in domains where the hypothesis space is well-defined, experiments are standardizable, and safety constraints are manageable through physical containment.

Scaling these successes requires advances along several axes:

- **Physical grounding**: Agents must reason about real-world constraints (reagent availability, equipment calibration, timing dependencies) absent from purely digital research environments.
- **Safety certification**: Autonomous labs handling hazardous materials require formal safety guarantees beyond software sandboxing— hardware interlocks, real-time monitoring, and fail-safe protocols.
- **Cross-domain transfer**: Can an agent trained in one laboratory domain (e.g., organic synthesis) transfer experimental reasoning to another (e.g., protein engineering)?
- **Integration with digital agents**: The most powerful research systems will combine physical experimentation with computational modeling, literature analysis, and simulation—requiring seamless orchestration across digital and physical modalities.

Biology represents perhaps the most impactful frontier: the complexity of biological systems, the slow pace of wet-lab experiments, and the vastness of the design space (protein engineering, synthetic biology, drug discovery) make autonomous experimentation particularly valuable yet particularly challenging (Gao et al., 2024).

7.5 Scaling Laws for Agent Capabilities

A fundamental open question is whether agent capabilities follow predictable scaling laws analogous to those governing foundation model pretraining. If so, what are the relevant axes of scale—model parameters, tool diversity, context length, or computational budget at inference time?

Preliminary evidence suggests that agent performance scales *differently* from standard language model benchmarks. The relationship between model size and task success on agentic benchmarks (SWE-bench, AgentBench, GAIA) often exhibits threshold effects rather than smooth power laws (Liu et al., 2024b; Mialon et al., 2023). This may reflect the compositional nature of agent tasks: success requires *all* component capabilities (planning, execution, error recovery) to exceed minimum thresholds simultaneously, creating step-function behavior at the system level even when individual capabilities scale smoothly.

Several empirical studies are urgently needed:

- **Tool scaling**: Does access to more tools monotonically improve agent performance, or does tool proliferation create selection complexity that degrades decision-making?
- **Inference-time compute**: Recent work on test-time scaling (Snell et al., 2024) and reasoning models such as OpenAI o1 (OpenAI, 2024b) and DeepSeek-R1 (DeepSeek AI, 2025) demonstrates that allocating more computation at inference (via search, self-verification, chain-of-thought, or iterative refinement) can substitute for model scale. How does this trade-off manifest in agentic settings?
- **Multi-agent scaling**: Do more agents improve research quality, and if so, with what returns?

The mixture-of-agents approach (Wang et al., 2024a) suggests diminishing but positive returns; rigorous scaling studies are needed.

- **Experience scaling**: As agents accumulate more episodic experience (§7.1), do capabilities improve logarithmically, linearly, or super-linearly with experience volume?

Establishing scaling laws for agents would enable principled resource allocation decisions: when to invest in larger models versus more tools, when to deploy single versus multi-agent architectures, and when the expected return on additional autonomy justifies the safety investment it requires.

7.6 Self-Bootstrapping Research Communities

A particularly ambitious direction extends multi-agent collaboration from task-level coordination to *community-level knowledge production*—populations of autonomous agents that jointly set research agendas, divide intellectual labor, peer-review each other's outputs, and collectively advance a field. Unlike current multi-agent systems that collaborate within a single session (Hong et al., 2024; Wu et al., 2024a), self-bootstrapping research communities would persist across time, accumulate shared knowledge, and exhibit emergent specialization analogous to human scientific communities. Early evidence suggests feasibility: the AI Scientist (Lu et al., 2024) already implements automated peer review, and recent work demonstrates that LLM-generated feedback on research papers overlaps substantially with human reviewer judgments (Liang et al., 2024). Programmatic frameworks such as DSPy (Khattab et al., 2024) could provide the optimization substrate for evolving community protocols, while tool-augmented agents (Qin et al., 2024) supply the breadth of capabilities needed for diverse research roles. The key open challenges are governance—how agent communities resolve disagreements, allocate credit, and avoid self-reinforcing biases without human oversight—and grounding, ensuring that community-generated knowledge remains empirically anchored rather than drifting into collectively hallucinated consensus.

8 Conclusion

This survey has provided a comprehensive analysis of autonomous research agents—systems that independently formulate hypotheses, design experiments, execute them, and iterate toward novel discoveries. We conclude by synthesizing the key findings across our six main contributions and identifying priorities for the research community.

Taxonomy and Definitions. Our L1–L5 autonomy hierarchy provides a precise vocabulary for an otherwise fragmented field. Current frontier systems (the AI Scientist, SWE-Agent, Devin, Claude Code) operate firmly at L4: they execute multi-step research workflows with strategic self-direction within bounded problem spaces. The transition to L5—agents that set their own research agendas across open domains—remains aspirational, requiring advances in persistent knowledge accumulation, self-directed exploration, and robust self-evaluation that no existing system yet demonstrates.

Architectural Patterns. We identified four dominant paradigms—single-agent reasoning loops, multi-agent collaboration, hierarchical orchestration, and tool-augmented execution—each with characteristic trade-offs between capability ceiling, reliability, cost, and interpretability. The trend is toward hybrid architectures that combine hierarchical coordination with specialized tool-using sub-agents, but no single design dominates across all application contexts.

System Landscape. Our comparative analysis of 17 systems across a six-dimensional feature matrix reveals rapid maturation: from fragile prototypes (AutoGPT, 2023) to production-grade

systems resolving real engineering tasks at over 70% success rates within eighteen months. However, this progress concentrates in well-defined domains (code, constrained optimization) while open-ended scientific discovery remains largely at the demonstration stage.

Evaluation Challenges. The field's evaluation infrastructure has advanced considerably, with benchmarks like SWE-bench providing standardized measurement. Yet fundamental challenges persist: open-ended research lacks ground truth, cost reporting is inconsistently applied, and benchmark saturation at the top end obscures meaningful capability differences. The critique by Kapoor et al. (2024)—that many reported improvements reflect evaluation optimization rather than genuine capability gains—demands ongoing methodological vigilance.

Open Problems. Among the six challenges we identified, three stand out as defining the research frontier. First, the *cognitive loop problem*: agents still fail to recognize when they are stuck, perseverating on failed strategies rather than seeking fundamentally different approaches. Second, *evaluation of novelty*: without reliable automated measures of research quality and originality, we cannot close the loop on agent self-improvement. Third, *safety and alignment*: as agents become more capable, the gap between what they *can* do and what they *should* do widens, requiring governance frameworks that do not yet exist at adequate maturity.

A Call to Action. The transition from copilots to colleagues is neither inevitable nor uniformly beneficial. Realizing the promise of autonomous research agents while managing their risks requires coordinated effort across several fronts: developing principled evaluation frameworks for open-ended research, establishing safety standards for agent deployment in high-stakes domains, creating shared infrastructure that democratizes access beyond well-resourced institutions, and building the theoretical foundations (scaling laws, formal verification, regret bounds) that can transform agent development from an empirical art into an engineering discipline.

The pace of progress suggests that L5 autonomy—agents capable of self-directed, long-horizon research programs—is a question of *when* rather than *whether*. The research community's task is to ensure this transition occurs with adequate understanding, appropriate safeguards, and equitable distribution of benefits. We hope this survey, by mapping the landscape and surfacing the critical open problems, contributes to that endeavor.

References

Anthropic. The Claude 3 model family: Opus, Sonnet, and Haiku. *Anthropic Technical Report*, 2024a. Family of models with strong instruction-following and agentic capabilities used in Claude Code and other agent systems.

Anthropic. Claude code: AI-powered software engineering agent. `https://docs.anthropic.com/en/docs/claude-code`, 2024b. Agentic coding tool operating in terminal with repository understanding, multi-file editing, and iterative debugging.

Anthropic. Claude 4 system card. *Anthropic Technical Report*, 2025. Claude 4 model family (Opus, Sonnet, Haiku) with enhanced agentic capabilities, extended context, and improved tool use for autonomous coding and research tasks.

Assaf Assafelovic. GPT Researcher: Autonomous agent for comprehensive online research. `https://github.com/assafelovic/gpt-researcher`, 2023. Autonomous research agent that plans queries, scrapes sources, filters information, and generates research reports.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. Introduces MBPP benchmark of 974 crowd-sourced Python programming problems for evaluating code generation.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. GPT-4 powered system that autonomously designs, plans, and executes chemical experiments on robotic hardware.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning (ICML)*, 2022. RETRO: retrieval-enhanced transformer demonstrating benefits of external memory for language models.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. ChemCrow: Augmenting large-language models with chemistry tools. *Nature Machine Intelligence*, 2024. LLM agent integrating 18 chemistry tools for synthesis planning, drug discovery, and materials design.

Alan Chan, Rebecca Salganik, Alva Markelius, Chris Pang, Nitarshan Rajkumar, Igor Krawczuk, Tarun Maharaj, Frank Ruber, Daniel Elm, John Horton, et al. Harms from increasingly agentic algorithmic systems. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2023. Analyzes risks from increasingly autonomous AI systems across multiple dimensions of harm.

Dong Chen, Shaoxin Lin, Muhan Zeng, Daoguang Zan, Jian-Gang Wang, Anton Cheshkov, Jun Sun, Hao Yu, Guoliang Dong, Artem Aliev, et al. CodeR: Issue resolving with multi-agent and task graphs. *arXiv preprint arXiv:2406.01304*, 2024a. Multi-agent system with task graphs for resolving GitHub issues; decomposes into manager, reproducer, fault localizer, editor, and verifier.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. In *arXiv preprint arXiv:2107.03374*, 2021. Introduces Codex and HumanEval benchmark for evaluating functional correctness of code generation.

Yubo Chen, Yuxia Hu, Hang Luo, Haiyang Sun, Jifan Gao, Binhao Wang, and Minjun Zhang. Sciagent: Tool-augmented language models for scientific reasoning. *arXiv preprint arXiv:2402.11451*, 2024b. Tool-augmented LLM agent for scientific reasoning with access to computation, data analysis, and domain tools.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. GSM8K: 8,500 grade school math word problems requiring multi-step reasoning; introduces verifier-based approach.

Cognition Labs. Devin: The first AI software engineer. `https://www.cognition.ai/blog/introducing-devin`, 2024. Autonomous software engineer with shell, editor, and browser; operates in sandboxed environment for multi-step engineering.

DeepSeek AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. Open-source reasoning model trained via large-scale RL; demonstrates that reasoning capabilities can emerge through reinforcement learning without supervised chain-of-thought data.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning (ICML)*, 2024. Multiple LLM agents debating improves factuality and reasoning; demonstrates emergent error-correction through multi-round debate.

Shanghua Gao, Ada Alber, Siddharth Bhatt, et al. Empowering biomedical discovery with AI agents. *arXiv preprint arXiv:2404.02831*, 2024. Survey of AI agents for biomedical research covering literature review, experiment design, and data analysis.

Gemini Team, Google. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. Multimodal foundation model family with strong reasoning capabilities across text, code, and vision.

Google DeepMind. An AI co-scientist. *Nature*, 2025. AI co-scientist: multi-agent system built on Gemini 2.0 for generating novel research hypotheses, experimental plans, and accelerating scientific collaboration.

Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Foundations and Trends in Programming Languages*, 4(1-2):1–119, 2017. Comprehensive survey of program synthesis approaches: enumerative, constraint-based, and neural-guided.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. RAP: treats LLM as both world model and reasoning agent, applying MCTS for deliberate planning.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 12,500 competition-level math problems across 7 subjects; foundational benchmark for mathematical reasoning evaluation.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024. Encodes SOPs into multi-agent collaboration; agents communicate via structured artifacts rather than free-form chat.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024. Meta Agent Search algorithm iteratively designs novel agent architectures, prompts, and tool configurations that outperform hand-crafted systems.

Dong Huang, Jie M Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. AgentCoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2024a. Separates code generation, test design, and execution into specialized agents; 93.9% on HumanEval.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Benchmarking large language models as AI research agents. *arXiv preprint arXiv:2310.03302*, 2023. MLAgentBench: evaluates LLMs on ML research tasks including model training, debugging, and optimization.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. Closed-loop LLM planning for embodied agents using environment feedback (scene descriptions, success signals) as inner monologue.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Liu, and Feng Huang. Understanding the planning of LLM agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024b. Comprehensive survey of planning methods in LLM agents: generation, refinement, and feedback mechanisms.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019. Foundational reference on AutoML covering hyperparameter optimization, NAS, and meta-learning.

Md. Ashraful Islam, Nishath Mohammed, Rifat Shahriar Haque, and Wei Li. MapCoder: Multi-agent code generation through planning, code generation, and debugging. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. Multi-agent code generation with recall, plan, code, and debug agents achieving state-of-art on HumanEval and MBPP.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Leszama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. Continuously updated coding benchmark from competitive programming platforms; addresses contamination concerns.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. 2,294 real GitHub issues from 12 Python repos; de facto standard for evaluating code agents.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitarshan Raheja, and Arvind Narayanan. AI agents that matter. *arXiv preprint arXiv:2407.01502*, 2024. Critiques agent evaluation practices; proposes cost-accuracy Pareto analysis and identifies benchmark overfitting in agent research.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Mober, et al. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024. Programming framework that compiles declarative LM pipelines into optimized prompt chains via teleprompters; enables systematic agent workflow optimization.

Megan Kinniment, Lucas Jun Koba Sato, Haoxing Du, Brian Goodrich, Max Hasin, Lawrence Chan, Luke Harold Miles, Tao R Lin, Hjalmar Wijk, Joel Burget, et al. Evaluating language-model agents on realistic autonomous tasks. *arXiv preprint arXiv:2312.11671*, 2024. METR evaluations of LLM agents on autonomous tasks including ML research and software engineering.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Foundational RAG framework combining retrieval with generation for knowledge-intensive tasks.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Role-playing framework with inception prompting for studying cooperative AI agent behaviors.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. DeepMind system achieving human-competitive performance in programming competitions via massive sampling and filtering.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023. Multi-agent debate framework that encourages divergent thinking and improves reasoning through adversarial discussion.

Weixin Liang, Zachary Iber, Yelena Kudina, Yongchan Li, Yaohui Tian, Laurens He, Olga Tow, Mark Steyvers, Tatsunori Hashimoto, et al. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *arXiv preprint arXiv:2310.01783*, 2024. Large-scale study showing LLM-generated paper reviews overlap substantially with human reviewer feedback; implications for automated research evaluation.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. EvalPlus: augments HumanEval/MBPP with 80x more tests; reveals 10–20% pass rate drops under rigorous evaluation.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024b. 8 interactive environments (OS, DB, web, games) evaluating LLM agent capabilities systematically.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. End-to-end autonomous research: idea generation, experiment execution, paper writing, and automated peer review at $15/paper.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. LLMs iteratively refine their own outputs using self-generated feedback without additional training.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. *arXiv preprint arXiv:2311.12983*, 2023. 466 real-world questions requiring multi-step reasoning, tool use, and web browsing for evaluation of AI assistants.

Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Cyril Zakka, Yash Dalmia, Eduardo Pontes Reis, Pranav Rajpurkar, and Jure Leskovec. BioPlanner: Automatic evaluation of LLMs on protocol planning in biology. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. Benchmark for evaluating LLM ability to generate step-by-step biological experimental protocols.

Yohei Nakajima. BabyAGI: Task-driven autonomous agent. `https://github.com/yoheinakajima/babyagi`, 2023. Minimal autonomous agent demonstrating task creation, prioritization, and execution loop with vector memory.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2022. Fine-tuned GPT-3 to browse the web for question-answering, trained with human feedback; answers preferred over human on ELI5.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. Foundation model enabling most autonomous agent systems; demonstrates broad reasoning and tool-use capabilities.

OpenAI. Practices for governing agentic AI systems. *OpenAI Whitepaper*, 2024a. Proposes governance practices for agentic systems: maintaining oversight, limiting capabilities, monitoring, and accountability.

OpenAI. Learning to reason with LLMs. *OpenAI Blog*, 2024b. Introduces o1 reasoning model using chain-of-thought at inference time to solve complex math, coding, and science problems; achieves expert-level performance on competition benchmarks.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. Applies OS virtual memory concepts to LLM context management for persistent, long-context agents.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *ACM UIST*, 2023. 25 LLM-powered agents simulate believable human behavior with memory, reflection, and planning in a sandbox.

Chen Qian, Xin Cong, Wei Liu, Cheng Yang, Weize Chen, Yusheng Su, Yufan Deng, Jiahao Li, Juyuan Xie, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. Virtual software company with role-playing agents (CEO, CTO, programmer, tester) collaborating via chat chains.

Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. TaskWeaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541*, 2023. Microsoft code-first agent framework converting user requests into executable code with plugin support.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024. Constructs ToolBench with 16,000+ real APIs and trains ToolLLaMA; introduces DFSDT algorithm for complex multi-tool reasoning.

Reworkd. AgentGPT: Autonomous AI in the browser. `https://github.com/reworkd/AgentGPT`, 2023. Browser-based autonomous AI agent platform with no-code interface for goal-oriented task decomposition and execution.

Toran Bruce Richards. AutoGPT: An autonomous GPT-4 experiment. `https://github.com/Significant-Gravitas/AutoGPT`, 2023. First widely-adopted autonomous agent loop: goal decomposition, internet access, memory, and self-prompting with GPT-4.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco J R Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Firdous, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. DeepMind FunSearch: LLM-guided evolutionary search discovers new mathematical constructions surpassing known bounds.

SAE International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. Technical Report J3016_202104, SAE International, 2021. Defines L0-L5 driving automation levels; provides the analogy template for our research agent autonomy taxonomy.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. LLMs self-supervisedly learn when and how to call external APIs to improve predictions.

Yucheng Shao, Jushi Jiang, Haotian Kanber, and Monica S. Lam. Assisting in writing Wikipedia-like articles from scratch with large language models. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024. STORM: multi-perspective question asking system that simulates expert interviews to generate comprehensive Wikipedia-style articles.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Uses LLM as controller to orchestrate specialized models from Hugging Face for complex AI tasks.

Toby Shevlane, Sebastian Farquhar, Ben Garfinkel, Mary Phuong, Jess Whittleston, Jade Leung, Daniel Kokotajlo, Nahema Marchal, Markus Sherburn, Lennart Heim, et al. Model evaluation for extreme risks. *arXiv preprint arXiv:2305.15324*, 2023. Framework for evaluating dangerous capabilities: autonomous replication, persuasion, cyber offense, and CBRN risks.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Agents improve through self-reflective verbal feedback stored in episodic memory, achieving 91% on HumanEval.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. Demonstrates that allocating more computation at inference time (search, verification, refinement) can substitute for larger model parameters.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023. Proposes CoALA framework unifying language agent designs through cognitive architecture principles.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. MedAgents: Large language models as collaborators for zero-shot medical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 599–621, 2024. Multi-agent framework with role-playing medical experts collaborating for improved zero-shot medical reasoning.

Minyang Tian, Luyu Chen, Guoyin Niu, Xiaobo Feng, Fei Wang, Hao Xiong, Bei Shi, and Zhouhan Liu. SciCode: A research coding benchmark curated by scientists. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. Benchmark of 338 scientific computing problems from 16 disciplines requiring multi-step research coding. Published at NeurIPS 2024.

Gary Tom, Stefan P. Schmid, Sterling G. Baird, Yang Cao, Kourosh Darvish, et al. Self-driving laboratories for chemistry and materials science. *Chemical Reviews*, 124(16):9633–9732, 2024. Comprehensive review of self-driving laboratories integrating AI-driven decision-making, robotic automation, and closed-loop experimentation for materials discovery.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a. LLM-powered lifelong learning agent in Minecraft with automatic curriculum and persistent skill library.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620:47–60, 2023b. Comprehensive Nature review on AI for scientific discovery spanning multiple domains.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024a. Layered multi-agent architecture where each layer's agents refine outputs from the previous layer, achieving state-of-art on benchmarks.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2024b. Comprehensive survey covering agent construction, applications, and evaluation with profiling/memory/planning/action modules.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024c. Comprehensive survey of continual learning addressing catastrophic forgetting; relevant to agents that accumulate knowledge across sessions.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. CodeAct: Executable code actions elicit better LLM agents. *arXiv preprint arXiv:2402.01030*, 2024d. Shows that using executable Python code as the action space outperforms JSON/text-based agent actions.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, et al. OpenHands: An open platform for AI software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024e. Open-source platform for building and evaluating AI software developer agents with sandboxed execution.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. DEPS: Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023c. Interactive planning framework using describe-explain-plan-select cycle for open-world multi-task agents in Minecraft.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Demonstrates that intermediate reasoning steps dramatically improve LLM performance on complex tasks.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *International Conference on Learning Representations (ICLR)*, 2024a. Microsoft framework for customizable conversable agents supporting LLM, human, and tool modes.

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. *arXiv preprint arXiv:2402.07456*, 2024b. Generalist computer agent framework with self-improvement via tool creation and reuse; evaluated on GAIA benchmark.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023. Survey of LLM-based agents covering brain, perception, and action modules with applications across domains.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024. Non-agentic approach to software engineering using localization-then-repair achieving competitive SWE-bench results.

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. *arXiv preprint arXiv:2405.15793*, 2024. Introduces Agent-Computer Interface design for code agents; resolves 12.5% of SWE-bench issues autonomously.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a. Generalizes chain-of-thought to tree-structured exploration with search algorithms for complex reasoning.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b. Introduces interleaved reasoning traces and actions for LLM agents, achieving strong performance on knowledge-intensive and decision-making tasks.

Shunyu Yao, Karthik Narasimhan, et al. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *arXiv preprint arXiv:2406.12045*, 2024. Benchmark evaluating LLM agents on realistic tool-agent-user interactions across domains including airline and retail customer service.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025a. Automated optimization of agentic workflows using Monte Carlo tree search over computational graphs. Published at ICLR 2025.

Ziru Zhang, Siting Li, Jialu Yan, Yuntao Zhao, Zhihong Liu, Lin Gui, and Yulan Zhang. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations (ICLR)*, 2025b. Benchmark with 44 tasks from 12 scientific disciplines evaluating language agents on data-driven discovery. Published at ICLR 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. LLM agents autonomously learn from experience without parameter updates; extract reusable insights from success/failure trajectories.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a. Unifies reasoning, acting, and planning via Monte Carlo tree search over language agent trajectories.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023b. Realistic self-hosted web environment with 812 tasks for evaluating autonomous web agents.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024. Models multi-agent systems as computational graphs that can be automatically optimized for performance.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017. Pioneering work using RL to automatically design neural network architectures.