Reflexion and Beyond: Verbal Self-Reflection in Large Language Models

Adam Belkassmi, Jakub Naruszko and Louisa Bahr

Abstract—Large Language Models (LLMs) have demonstrated impressive capabilities as autonomous agents in diverse tasks. However, enabling these agents to efficiently learn from trial-and-error experiences remains a core challenge. Traditional reinforcement learning approaches require extensive training samples and costly fine-tuning, which is impractical for large models. This paper explores an alternative paradigm: verbal self-reflection. We review the Reflexion framework, which provides an LLM agent with the ability to generate and utilize natural language feedback (self-critique) to iteratively improve its performance without updating model weights. We detail Reflexion's architecture and algorithm, evaluate its empirical performance on decision-making, reasoning and coding benchmarks and situate it within a broader taxonomy of iterative improvement frameworks, including Self-Refine, ReAct, Toolformer and Voyager. A comprehensive comparative analysis is presented along key dimensions such as performance, efficiency, cost, interpretability and generality. Our analysis reveals that verbal reinforcement significantly enhances task success and highlights a nuanced trade-off space among competing agent architectures. We also discuss the limitations of self-generated feedback, alongside the ethical and governance considerations for deploying such self-improving systems. The paper concludes by synthesizing these findings and outlining future research directions, including recursive self-improvement, scalable memory and the application of reflection for AI safety and alignment.

1 Introduction

1.1 The Challenge of Efficient Learning in LLM Agents

To evolve from simple predictors to dynamic agents, Large Language Models (LLMs) must learn from the consequences of their actions. Reinforcement Learning (RL) has emerged as a powerful method for enabling agents to learn from interactions within their environments. At its core, RL is concerned with how an agent ought to take actions in an environment to maximize cumulative reward over time. The central elements of RL include state spaces, action spaces, rewards, policies and value functions, which collectively guide the learning process of the agent. A policy, the primary decision-making component, dictates how the agent selects actions based on perceived states, while the value functions estimate the expected returns [1, pp. 1-14].

Despite its strengths, traditional gradient-based RL faces substantial challenges when applied to LLMs primarily due to their extensive computational demands, large datasets and lengthy training periods, complicating scalability and iterative refinement. Moreover, these methods often lack interpretability, hindering efficient troubleshooting and improvement. Recent approaches like Reflexion [2] propose verbal self-reflection to address these issues by providing explicit textual feedback, enabling faster, more transparent and targeted model enhancements without extensive retraining.

1.2 Research Focus: Evaluating Verbal Self-Reflection

Recent research on LLMs increasingly emphasizes the use of verbal self-reflection as a method to enhance learning efficiency and interpretability. Verbal self-reflection involves enabling LLMs to articulate insights about their own reasoning processes, transforming simple feedback signals into explicit textual descriptions. This approach not only aids in pinpointing specific errors but also accelerates targeted model improvements without necessitating extensive retraining procedures [2].

The Reflexion framework, proposed by Shinn et al. [2], demonstrates how verbal self-reflection substantially improves model performance across various tasks such as coding (HumanEval benchmark), reasoning (HotpotQA [3]) and decision-making (ALFWorld [4]). Their findings show that explicit, language-based reflection helps models rapidly refine their strategies by directly identifying errors and potential improvements from past attempts [2].

Complementing these insights, recent analyses further underscore the benefits of verbal reflection, particularly highlighting how reflection significantly enhances problem-solving performance through more nuanced feedback compared to scalar rewards alone. Their research indicates that verbal reflections promote a deeper internalization of task-specific strategies, resulting in robust and generalizable performance improvements in subsequent trials [5].

Furthermore, broader perspectives provided by the Turing Post [6] elaborate on reflection as a pivotal capability for autonomous agents, enabling them to learn effectively from their mistakes. According to their analysis, reflection provides crucial introspective capabilities that allow agents to self-diagnose and adapt their strategies dynamically, significantly outperforming baseline agents without reflective capabilities in complex, iterative scenarios [6].

In conclusion, evaluating verbal self-reflection within LLMs highlights its potential to enhance both efficiency and interpretability significantly. This research direction suggests substantial practical implications, fostering adaptive, robust and highly interpretable autonomous agents capable of iterative self-improvement.

1.3 Contribution and Paper Structure

This paper makes three primary contributions to the study of iterative LLM agents. First, it provides a detailed synthesis and analysis of key iterative improvement frameworks, beginning with an in-depth exploration of the primary Reflexion [2] framework and expanding to a broader taxonomy that includes Self-Refine [7], ReAct [8], Toolformer [9] and Voyager [10]. Second, it establishes a structured evaluation rubric consisting of six key metrics: Task Performance, Efficiency, Operational Cost, Interpretability, Complexity and Generalization, designed to systematically assess these complex agent architectures. Third, it presents a detailed comparative analysis that applies this rubric, culminating in a synthesis of the distinct trade-offs, strengths and optimal use cases for each framework.

The paper is organized to guide the reader from foundational concepts to this nuanced analytical synthesis. *Chapter 2* provides a deep dive into the Reflexion framework, detailing its architecture, algorithm and empirical performance. *Chapter 3* broadens this context by presenting a taxonomy of the four other prominent frameworks, outlining their core methodologies. *Chapter 4* contains the core analytical contribution of this paper: it first defines our *six evaluation* metrics *(4.1)* and then applies them in a metric-by-metric comparison of the agent architectures *(4.2)*. Finally, *Chapter 5* discusses the practical limitations and implications of these systems regarding reliability, ethics and cost, before *Chapter 6* concludes with a summary of findings and an outlook on future research.

2 The Reflexion Framework: A Deep Dive into Verbal Reinforcement

The Reflexion framework, introduced by Shinn et al. [2], represents an innovative approach to reinforcement learning in LLMs that leverages verbal self-reflection to enhance model performance and interpretability. Unlike traditional gradient-based methods, Reflexion utilizes explicit textual feedback, allowing models to iteratively refine their strategies based on reflective insights derived from previous trials. This chapter provides an in-depth exploration of Reflexion, detailing its unique *Actor-Evaluator-Self-Reflection architecture*, the iterative algorithmic process and empirical validations demonstrating its effectiveness across key benchmarks such as HumanEval, HotpotQA and ALFWorld.

2.1 The Actor-Evaluator-Self-Reflection Architecture

The *Actor-Evaluator-Self-Reflection* architecture is central to the Reflexion framework [2], structured around three interdependent modules designed to facilitate iterative learning and refinement in LLMs.

The Actor component is responsible for generating initial responses or actions based on the given prompts or environmental states. It leverages pre-trained language models to propose solutions or strategies dynamically during task execution [11].

The Evaluator module assesses the Actor's outputs, typically through defined success criteria or feedback mechanisms. It identifies errors or shortcomings in the actions or responses produced by the Actor, enabling the model to distinguish between successful and unsuccessful outcomes [11].

The Self-Reflection module synthesizes feedback from the Evaluator into explicit textual reflections. These reflections clearly articulate the errors made and provide actionable insights for future improvement. The reflections are subsequently integrated back into the Actor's reasoning process for the next iteration, significantly enhancing the quality and efficiency of decision-making [12].

By explicitly verbalizing reflections, models can better internalize feedback, rapidly adapt strategies and enhance interpretability [2].

Shinn et al. [2] emphasize that the explicit linguistic feedback loop helps LLMs systematically refine their behaviors and substantially improve their performance across diverse and complex tasks.

2.2 The Algorithmic Cycle: From Trajectory to Iterative Refinement

The algorithmic cycle central to the Reflexion framework [2] involves iteratively refining the strategies of LLMs based on their generated trajectories, evaluative feedback and subsequent reflections. This cycle mirrors foundational concepts found in dynamic programming (DP), specifically through methods such as policy iteration and value iteration, as discussed by Sutton and Barto [1, pp. 73-89].

In the Reflexion framework proposed by Shinn et al. [2], a trajectory refers to the sequence of actions or responses generated by the Actor module during each attempt to solve a given task. Following each trajectory, the Evaluator assesses performance, providing explicit textual feedback that identifies errors or successes. The Self-Reflection module then synthesizes this evaluative feedback into clear, actionable reflections that articulate insights and propose improvements for future iterations [2].

This iterative refinement approach aligns closely with DP's generalized policy iteration (GPI) concept, where evaluation and improvement processes cyclically interact. Policy iteration involves repeatedly performing two key processes: policy evaluation, which assesses the effectiveness of a given policy and policy improvement, which enhances the policy based on the evaluation outcomes. Value iteration similarly iteratively updates value estimates to progressively refine policies until optimal or near-optimal solutions are identified [1 pp. 73-89].

Both DP methods emphasize iterative improvement based on past experiences or trajectories, resembling the Reflexion architecture's integration of self-generated textual feedback. By embedding evaluative insights explicitly within each iteration, Reflexion enhances learning efficiency and accelerates the refinement of decision-making strategies in LLMs. This structured loop of generating, evaluating, reflecting and refining policies underscores the cycle's effectiveness, significantly improving performance and interpretability across diverse tasks and domains.

2.3 Core Mechanisms: Transforming Rewards and Prompt Engineering

The effectiveness of the Reflexion framework [2] in LLMs critically depends on two core mechanisms: transforming sparse rewards into explicit textual reflections and employing sophisticated prompt engineering strategies.

Transforming sparse rewards involves converting basic evaluative feedback, typically numerical or binary, into detailed textual reflections. These textual reflections explicitly articulate the agent's mistakes and outline potential corrective measures, providing agents with actionable insights. This explicit reflection accelerates improvement compared to traditional scalar rewards alone, enhancing learning transparency and promoting clearer understanding of performance outcomes [2].

Additionally, recent research highlights iterative refinement through self-feedback, underscoring how generated textual reflections facilitate deeper internalization of corrections and improvements, significantly enhancing subsequent outputs [7].

Prompt engineering further complements this reward transformation process by shaping the initial conditions for generating reflective outputs. Systematic prompt design directly influences LLM behaviors, optimizing response accuracy and interpretability through targeted instructions and iterative prompt refinement [13].

Structuring prompts explicitly to encourage introspective behavior guides LLMs in systematically generating solutions, evaluating their accuracy and clearly articulating reflections [14].

Furthermore, iterative refinement approaches, as outlined by recent studies, emphasize adjusting prompts dynamically based on previous outcomes to continually improve the quality and relevance of model-generated reflections [7].

Together, these mechanisms, transforming rewards into explicit textual reflections and meticulous prompt engineering, strengthen the Reflexion framework, significantly enhancing the efficiency, transparency and effectiveness of iterative learning cycles in LLMs.

2.4 Empirical Performance on Core Benchmarks

The Reflexion framework [2] was evaluated empirically across three core benchmark categories designed to test its generality and effectiveness: sequential decision-making, knowledge-intensive reasoning and code generation tasks. Empirical evaluations demonstrated substantial performance improvements over baseline models that did not incorporate reflective mechanisms.

In sequential decision-making tasks, specifically using the ALFWorld benchmark [4], a text-based game environment simulating real-world scenarios, Reflexion-enhanced agents exhibited a significant performance improvement. Reflexion achieved a success rate of *130 / 134, which equals ~97%*, marking an increase of *22* percentage points over the non-reflective baseline *(~75%/derived)* [2]. This indicates that explicit textual reflection significantly boosts an agent's capacity for goal-directed decision-making and adaptive problem-solving in dynamic environments. For knowledge-intensive reasoning tasks, evaluated via the HotpotQA benchmark [3], Reflexion demonstrated a notable improvement in accurately solving multi-hop reasoning questions. Performance increased from a baseline success rate of ~33% to *~53%*, reflecting a *20 percentage-point* improvement [2]. This advancement underscores the ability of reflective agents to better

handle complex reasoning chains and multi-step information retrieval tasks.

In code generation and debugging tasks evaluated using the HumanEval benchmark, Reflexion-enabled agents achieved outcomes with a *91% pass@1* accuracy, outperforming previous top models such as GPT-4, which previously recorded around an *80% pass@1* accuracy [2].

Additionally, Reflexion introduced challenging subsets like LeetcodeHardGym to demonstrate further robustness and generalizability in coding contexts, highlighting the advantages of iterative reflective feedback loops [2].

These benchmark results align with related empirical findings from advanced LLM evaluations. DeepSeek-R1's integration of reinforcement learning similarly highlighted iterative refinement capabilities to achieve improved reasoning and coding performance [15].

Moreover, OpenAI's GPT-4o system further validates the critical role of iterative self-improvement and fine-tuned feedback in achieving enhanced accuracy across multiple complex tasks, reinforcing the findings seen in Reflexion [16].

Overall, empirical evidence consistently demonstrates that incorporating reflective textual feedback into reinforcement learning frameworks yields substantial performance enhancements across various benchmarks, emphasizing the value and effectiveness of Reflexion's iterative reflective approach.

Table 1: Performance of Reflexion on benchmarks versus baseline agents. Results adapted from [2].

Task	Baseline Success	Reflexion Success
AlfWorld@10 (text game)	~ 75 %	~ 97**%** (+22 pp)
HotpotQA (multi-hop QA)	~ 33 %	~ 53 **%** (+20 pp)
HumanEval (code pass@1)	80 %	91 % (+11 pp)

Table 1. compares the final success rates of the Reflexion agent against a non-reflective baseline agent across three distinct and challenging domains: sequential decision-making (AlfWorld @Trial Number 10), knowledge-intensive reasoning (HotpotQA) and code generation (HumanEval). The data (where the absolute AlfWorld and HotpotQA values are visually estimated from Fig. 1./Fig. 2.) demonstrate that incorporating the verbal self-reflection mechanism leads to performance improvements, with gains ranging from *11* to *22 percentage points*, underscoring the effectiveness and generality of the approach [2].

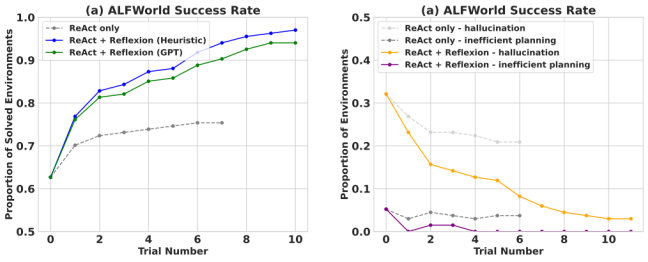

Fig. 1. (a) Performance of 134 AlfWorld tasks showing cumulative proportions of successfully solved tasks using techniques of (Heuristic) and (GPT) for binary classification. (b) Categorizing AlfWorld trajectories by reason of failure. Reproduced unaltered from [2].

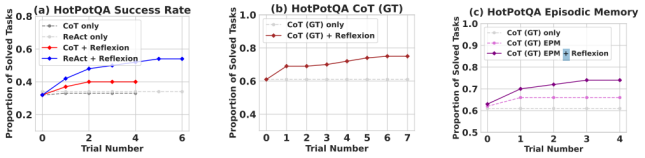

Fig. 2. Performance of Chain-of-Thought (CoT), ReAct and Reflexion on 100 HotPotQA questions. (a) Reflexion ReAct vs Reflexion CoT (b) Reflexion CoT (GT) only for reasoning (c) Reflexion vs episodic memory ablation. Reproduced unaltered from [2].

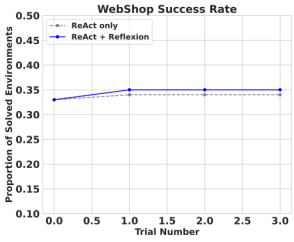

Fig. 3. Shows Reflexion vs ReAct performance on WebShop across 100 customer shopping requests. The addition of Reflexion fails to outperform the ReAct baseline, illustrating that Reflexion doesn't help in tasks requiring diversity and exploration. Reproduced and unaltered from [2].

3 A Taxonomy of Iterative Improvement Frameworks

The development of LLMs into capable autonomous agents hinges on their ability to learn and adapt from experience. While traditional training methods are often resource-intensive, a class of methodologies known as iterative improvement frameworks has emerged to enhance LLM capabilities more efficiently. These frameworks, such as Self-Refine [7], ReAct [8], Toolformer [9] and Voyager [10], empower models to systematically refine their outputs by leveraging internal or external feedback. By employing mechanisms ranging from self-generated critiques and explicit reasoning processes to the integration of external tools, these approaches guide models toward progressively better performance. This chapter provides a structured categorization of these key frameworks, analyzing their distinct methodologies and their applicability across various domains to highlight the evolving landscape of autonomous agent development.

Table 2: Overview of selected iterative LLM Agent Frameworks

Framework	Brief Description
Reflexion	Multi-trial agent that uses self-generated language critiques to improve over episodes; remembers past mistakes via a memory of reflections [2].
Self-Refine	Post-hoc refinement of a single output through self-critique and regeneration (no external environment) [7].
ReAct	Interleaves chain-of-thought reasoning with actions (e.g., API calls or environment steps) in a single trial to solve tasks [8].
Toolformer	Self-supervised fine-tuning that teaches an LLM to insert tool API calls into its generation to improve accuracy [9].
Voyager	Embodied lifelong-learning agent in a sandbox world (Minecraft [17]), using an LLM (GPT-4) to continually acquire and refine skills via feedback loops [10].

3.1 Self-Refine: Iteration via Self-Generated Feedback

Self-Refine [7] is a framework designed for scenarios where an LLM's initial output might be suboptimal, enabling the model to enhance it by generating its own feedback. It typically addresses single-turn tasks, such as responding to prompts or solving mathematical problems, through a clear iterative process. Initially, the LLM produces an answer, subsequently critiques its output by identifying errors or suggesting improvements and then generates a revised answer. This feedback-refinement cycle can be repeated multiple times, though in practice, it often involves only one or two refinement rounds [7].

Importantly, no external reward signals or environmental feedback are needed; the model leverages internal evaluative criteria explicitly provided within the prompt (e.g., "make the response more polite" or "correct reasoning errors").

Empirical evaluations demonstrate that Self-Refine significantly improves output quality across diverse tasks. Across seven varied benchmarks Self-Refine raises performance by about *20 percentage points* on average versus the same model's single-step generation baseline, and human judges consistently prefer its refined dialogue answers [7].

Similarly, in mathematical reasoning tasks, incorporating self-generated critiques notably reduced errors. A compelling advantage of Self-Refine is its inherent simplicity, as it requires no additional models or supplementary training data, depending exclusively on the introspective capability of the base LLM. However, Self-Refine focuses strictly on individual prompts without accumulating memory or supporting sequential decision-making across tasks. Feedback from each refinement cycle is utilized immediately and not retained, thus making Self-Refine primarily suitable as a test-time optimization approach for single-turn prompts.

3.2 ReAct: Synergizing Reasoning and Acting

ReAct [8] is a framework that unifies logical reasoning (via chain-of-thought prompting) with action-taking in an environment. In ReAct, the LLM is prompted to produce outputs alternating between Thoughts (natural language reasoning steps) and Actions (commands interacting with external tools or environments). For example, a ReAct agent solving a question-answering problem might reason, *"The question asks X; I should search the knowledge base,"* then execute the action *"SEARCH[X]"*. After observing results, the agent continues reasoning, *"The result suggests Y,"* before finally formulating its answer. This structured interleaving enables agents to dynamically gather additional information and adapt their plans based on new insights [8].

ReAct has been successfully applied to knowledge-intensive tasks using external tools such as Wikipedia APIs to reduce hallucinations, as well as interactive tasks like ALFWorld and WebShop (a virtual shopping environment). On the WebShop virtual-shopping benchmark, ReAct lifted the success rate to 40 %, an absolute gain of about 10 percentage points over the best prior baseline (30 %) [8].

Notably, Yao et al. reported that ReAct achieved a *34% absolute improvement* in task success compared to a pure imitation learning baseline on the ALFWorld benchmark, underscoring its effective integration of reasoning and action. Moreover, ReAct's explicit chain-of-thought reasoning enhances interpretability and debuggability by clearly justifying each action based on preceding reasoning steps [8].

Nevertheless, ReAct, as originally proposed, lacks mechanisms for learning from mistakes across episodes. Each reasoning cycle is isolated; failing once does not directly inform subsequent attempts unless externally guided or explicitly prompted again. This limitation, as noted by the authors, might naturally be addressed by incorporating previous reasoning traces into subsequent prompts, potentially enabling an LLM to leverage past reasoning experiences to improve future performance. Thus, ReAct significantly influenced subsequent iterative frameworks by demonstrating how an LLM can systematically integrate cognitive deliberation and practical execution within a unified prompting methodology [8].

3.3 Toolformer: Self-Supervised Tool Acquisition

Toolformer [9] is a framework designed to enhance the capabilities of LLMs by enabling them to autonomously learn and integrate external tools through self-supervised learning. Proposed by Schick et al., Toolformer allows the model itself to identify when and how to invoke external computational tools to improve its responses. The framework leverages a clear, effective procedure: initially, the LLM generates outputs suggesting possible tool interactions; then, through self-supervision, it evaluates these interactions for effectiveness, integrating successful ones into its workflow [9].

Toolformer operates by fine-tuning the LLM, first using a handful of demonstrations per tool and then on a much larger corpus that the model has automatically annotated with synthetic tool calls. Specifically, the model creates synthetic training examples demonstrating how calling external tools, such as a calculator, two different search engines (a general web search API and a Wikipedia search API), a question-answering system (Q&A System), a calendar or a machine translation system, can improve outputs. For instance, the model learns to insert an API call like *"<calc>3+5</calc>"* into its text, receive the result *"8"* and then incorporate that result into its final answer. This self-generated training set enables the LLM to learn when to invoke such tools, enhancing its practical utility and accuracy without requiring large-scale human annotation [9].

Empirical results demonstrate Toolformer's effectiveness across multiple domains. On tasks requiring precise factual knowledge or calculation, Toolformer (based on a 6.7B parameter model/GPT-J [18]) achieved performance competitive with models *more than an order of magnitude larger* (e. g., GPT-3 [19]) that lacked tool-use capabilities [9].

This approach also maintains interpretability, as the model explicitly signals when and why a tool is invoked. However, this offline training approach comes with a trade-off in flexibility; the model's capabilities are bounded by the specific tools provided during training. Unlike iterative methods, it has no mechanism to self-correct a new type of error at test time if that error pattern was not part of its training data. Nonetheless, Toolformer represents a significant advancement, showing how self-supervised learning can equip LLMs to overcome their innate limitations by acting to use external resources [9].

3.4 Voyager: An Embodied Agent for Open-Ended Lifelong Learning

Voyager [10] is an embodied agent framework enabling LLMs to engage in open-ended, lifelong learning within an interactive environment. The system uses GPT-4 as the controller for an autonomous agent that explores the world of Minecraft [17] indefinitely, learning skills and discovering new objectives without human intervention. Voyager integrates three key components: an automatic curriculum that proposes new goals to maximize exploration, a skill library where the agent stores learned behaviors as reusable Python code and an iterative prompting mechanism for self-correction [10].

The agent's learning cycle is driven by interaction. When attempting a skill, failure or errors prompt the LLM with the specific error message, which it then uses to refine its Python code. Over time, Voyager becomes highly proficient, learning to mine resources, craft tools and navigate the game's complex technology tree. Quantitatively, Voyager collected *3.3 × more* unique items and traveled *2.3 times* as far as previous state-of-the-art agents. Furthermore, its skill library allowed it to adapt to new, unseen worlds far more effectively than agents without such a long-term memory mechanism [10].

Voyager serves as a powerful proof-of-concept for continual learning, demonstrating that an LLM can effectively write and debug its own code through environmental feedback [10]. However, the framework's

effectiveness comes with significant trade-offs. Its reliance on a powerful proprietary model like GPT-4 and the constant interaction with the game environment make it computationally expensive [10]. Additionally, its design is highly specialized for the Minecraft API [10].

Nonetheless, Voyager highlights a path toward truly autonomous systems that continually learn and improve, blurring the line between static reasoning and active skill acquisition [10].

4 Comparative Analysis of Agent Architectures

In this chapter, we conduct an comparative analysis of various agent architectures, identifying and defining key evaluation metrics that facilitate a structured and detailed assessment. The analysis begins by examining core metrics including *task performance and efficacy*, *computational efficiency*, *operational cost*, *interpretability*, *architectural complexity and generalization capabilities*. Each metric is addressed in dedicated subsections, providing a thorough exploration of their implications and significance. The chapter concludes with a synthesis of these insights, offering a comprehensive understanding of how different architectures, including the Reflexion [2] framework, align with diverse application scenarios and performance requirements.

4.1 Defining the Metrics for Evaluation

4.1.1 Task Performance and Efficacy

Task Performance and Efficacy refers to the extent to which an agent successfully accomplishes its intended tasks, particularly when benchmarked against alternative methods or human-level performance. Evaluation typically includes metrics such as final success rates, accuracy levels and various quality indicators. For instance, metrics such as high pass@1 accuracy on coding tasks and notable improvements in QA and game environments serve as representative indicators of robust task performance [2]. Additionally, consistency is considered a crucial metric, assessing whether the framework reliably solves the majority of problems upon iterative retries and refinement attempts [2].

4.1.2 Learning and Computational Efficiency

Learning and Computational Efficiency assesses the computational resources and time required by an agent to perform tasks effectively. Important factors include the volume of data necessary for training, computational cost during inference and the speed of achieving optimal or near-optimal performance. Dong et al. [20] emphasize that in-context learning incurs substantial computational expenses as the cost increases significantly with the context length and number of in-context demonstrations. Their analysis highlights the trade-off between prompt length, computational resources and model performance, underscoring that extending the context often results in disproportionately higher computational costs relative to performance gains [20].

4.1.3 Operational Cost

Operational Cost evaluates the direct financial and computational resources required to deploy and operate an agent for a given task. This metric encompasses several factors, including the number of API calls made to an LLM, the total number of tokens processed per query, the resulting latency and any underlying infrastructure expenses. For iterative frameworks, this metric is particularly critical as multi-step reasoning or repeated trials can multiply these costs, directly impacting the system's scalability and viability in real-world applications. Ultimately, analyzing operational cost provides a practical lens for the core trade-off between the resources consumed and the performance gained by an agent architecture.

4.1.4 Interpretability and Diagnosability

Interpretability and Diagnosability evaluate how transparently an agent's decision-making processes and outcomes can be understood by human users or developers. This metric encompasses the agent's ability to clearly articulate reasoning steps, provide explanations for its decisions, facilitate straightforward identification and troubleshooting of errors or inefficiencies. High interpretability ensures users can trust and validate agent outputs, while robust diagnosability supports easier maintenance, refinement and debugging, significantly enhancing operational reliability and user confidence. For iterative and reflective frameworks, interpretability is particularly crucial, as it directly supports effective feedback loops and continuous improvement cycles, enhancing user confidence and operational reliability.

4.1.5 Architectural Complexity and Dependencies

Architectural Complexity and Dependencies assesses the intricacy of an agent’s design, including the number and interrelations of components, layers and external systems required for its operation. This metric addresses how complexity impacts deployment ease, maintenance efforts and scalability. Agents with lower complexity generally offer easier integration, faster deployment cycles and reduced maintenance overhead. Conversely, higher complexity and numerous dependencies can enhance functionality but might also introduce potential points of failure and increased resource demands. Evaluating architectural complexity is critical for balancing advanced capabilities against operational simplicity and reliability.

4.1.6 Generalization and Task Adaptability

Generalization and Task Adaptability evaluates an agent's ability to perform effectively across diverse tasks and in varied contexts, beyond the scenarios for which it was explicitly trained or optimized. This metric considers how readily and efficiently the agent can adapt its learned behaviors to novel situations or unforeseen task requirements. High generalization indicates robustness and flexibility, enabling the agent to maintain consistent performance without extensive retraining or manual adjustments. Effective adaptability further reflects the capacity to swiftly incorporate new information or task-specific refinements, underscoring the agent’s potential utility across a broad spectrum of practical applications.

4.2 Framework Comparison and Synthesis

When evaluating *Task Performance and Efficacy*, all frameworks exhibit robust capabilities, though their strengths manifest distinctly. Reflexion [2] and Self-Refine [7] demonstrate significant iterative improvement and consistent performance across tasks such as coding, QA and sequential decision-making, driven by structured textual reflections and self-feedback loops. ReAct [8] similarly achieves high efficacy, particularly excelling in tasks demanding intricate reasoning and sequential actions. Toolformer [9] differentiates itself notably in tasks benefiting from external tools, achieving exceptional performance when relevant tools are available but facing limitations otherwise. Voyager [10] uniquely stands out in dynamic, open-ended environments, consistently delivering high efficacy through continuous and adaptive lifelong learning.

Regarding *Learning and Computational Efficiency*, Reflexion [2] stands out for minimizing retraining by leveraging iterative textual reflections, demonstrating relatively higher efficiency. ReAct [8] and Self-Refine [7] maintain commendable efficiency, though the extended reasoning steps and iterative feedback loops respectively can moderately increase computational demands. Toolformer [9] presents variability in computational efficiency, highly dependent on the complexity of tool integration and retrieval processes, often incurring greater computational overhead. Voyager [10], with its lifelong learning cycles, achieves commendable efficiency considering its

continual adaptive processes, yet inherently involves substantial computational resources over prolonged interactions.

The frameworks display a clear gradient in *Operational Costs*, largely influenced by the frequency of model interactions and reliance on external systems. At the more moderate end, ReAct [8] and Self-Refine [7] utilize internal iterative loops or reasoning sequences, moderately increasing resource use. Reflexion [2], while still moderate, experiences costs directly proportional to the number of reflective iterations per task. Conversely, Toolformer [9] and Voyager [10] represent the upper spectrum of operational costs due to their external dependencies: Toolformer [9] relies significantly on external API interactions for tool invocation and Voyager [10] continuously interacts intensively with its operational environment, substantially increasing overall resource demands.

Interpretability and Diagnosability vary across frameworks primarily based on the clarity and transparency of their reasoning processes. Reflexion [2], Self-Refine [7] and ReAct [8] score notably high, explicitly articulating reasoning and decision-making steps, thus substantially aiding troubleshooting and validation. Conversely, Toolformer [9] demonstrates slightly lower interpretability due to the opaqueness introduced by external tool dependencies, potentially obscuring internal decision processes. Voyager [10] maintains moderate interpretability, balancing the complexity inherent in lifelong adaptive learning with a structured approach to decision transparency.

In assessing *Architectural Complexity and Dependencies*, frameworks differ significantly in their simplicity and external integration requirements. ReAct [8] and Reflexion [2] maintain relatively low to moderate complexity, favoring streamlined processes and minimal external dependencies. Self-Refine [7] similarly holds moderate complexity with minimal additional components. In contrast, Toolformer [9] exhibits substantially higher complexity due to extensive external tool dependencies. Voyager [10] represents the most architecturally complex, integrating advanced mechanisms for continual interaction and extensive memory management to support open-ended lifelong learning.

Generalization and Task Adaptability are areas where frameworks clearly differentiate based on their operational focus and design. Reflexion [2], Self-Refine [7] and ReAct [8] show consistently high generalization, adeptly applying iterative learning and reasoning strategies across diverse and complex tasks. Toolformer [9] presents conditional adaptability, excelling in environments where external tools are readily available but limited in contexts lacking suitable integrations. Voyager [10] surpasses all frameworks in adaptability, exceptionally suited for dynamic and evolving environments through its continuous lifelong learning paradigm, enabling it to swiftly adapt and generalize to new and unforeseen scenarios.

In summary, there is no single "best" framework: each excels in different aspects. If one needs a quick performance boost on static tasks and values ease of use, Self-Refine [7] is attractive. If the goal is an agent that can plan and act in a single trial with transparency, ReAct [8] is excellent. For agents that must truly improve over multiple attempts, Reflexion [2] offers a simple yet effective recipe. For leveraging external knowledge or tools to boost accuracy, Toolformer [9] shows what is possible (at the cost of extra training). And for ambitious open-world exploration, Voyager [10] demonstrates how far an LLM can go when equipped with memory and allowed to continuously learn.

Table 3: Snapshot of five LLM-agent frameworks rated on six key dimensions

Framework	1	2	3	4	5	6
Reflexion	✓	✓	△	✓	△	✓
Self-Refine	✓	✓	△	✓	△	✓
ReAct	✓	△	△	✓	✓	✓
Toolformer	△	△	✗	△	✗	△
Voyager	✓	△	✗	△	✗	✓

Legend: ✓ strong | △ moderate | ✗ weaker

1 - Task Performance & Efficacy
2 - Learning & Computational Efficiency
3 - Operational Costs
4 -Interpretability & Diagnosability
5 - Architectural Complexity & Dependencies
6 - Generalization & Task Adaptability

5 Limitations and Practical Implications

The advanced agent frameworks discussed previously, notably Reflexion [2] and Self-Refine [7], represent significant progress in enhancing LLM capabilities through iterative improvement. However, their successful transition to real-world use demands careful consideration of several practical limitations and implementation challenges. This chapter critically examines these challenges, focusing specifically on the reliability of self-generated critique, necessary frameworks for data governance and ethical responsibility and comprehensive cost-benefit analyses essential for enterprise adoption.

5.1 The Reliability of Self-Generated Critiques

The central premise of frameworks like Reflexion [2] and Self-Refine [7] is that an LLM can generate its own feedback to iteratively improve its performance. This capability is powerful, but its reliability is a significant concern. The core challenge is that if an LLM is flawed enough to produce an incorrect initial output, its capacity to generate a correct and useful critique is also questionable. This creates a scenario of the "blind leading the blind," potentially reinforcing initial errors or introducing subtler mistakes.

A critical distinction in evaluating feedback is between process-based and outcome-based supervision. Outcome-based feedback only assesses the correctness of the final answer, whereas process-based feedback evaluates the intermediate reasoning steps that led to it.

Research by Uesato et al. [21] and Lightman et al. [22] highlights that models rewarded only for correct outcomes can learn to produce right answers for the wrong reasons, a phenomenon that undermines their robustness. For example, a model might correctly guess the solution to a math problem despite using incorrect or flawed reasoning. When applied to self-reflection, this implies a model could generate a critique that leads to a correct final answer on a specific problem but is based on a logically unsound principle, making the "learned" skill unreliable.

The work of Lightman et al. on step-by-step verification suggests that a more robust approach involves verifying the reasoning process itself [22].

This principle directly applies to the self-critique mechanism. A reliable self-reflection framework must not only produce a better final result but must do so through a sound and verifiable critical process. Without this, there is a risk that the model learns to generate feedback that appears plausible but does not generalize or reflect genuine understanding.

The system cards for advanced models like GPT-4o acknowledge these challenges, noting that despite internal testing and red-teaming,

ensuring the consistent reliability and factuality of complex reasoning remains an ongoing research problem [16].

Ultimately, the trustworthiness of frameworks that rely on self-generated critiques is bounded by the inherent reliability of the base model, making it a crucial bottleneck for their application in high-stakes domains such as medicine, law, or finance.

5.2 Data Governance and Ethical Considerations

Beyond technical reliability, the deployment of autonomous and continuously learning agents raises profound data governance and ethical challenges. These systems operate on vast datasets and generate extensive logs of their "thoughts", actions and interactions, creating a complex web of responsibilities for developers and users.

A primary concern is data privacy and security. The extensive memory required by frameworks like Voyager [10] or the detailed reasoning traces generated by ReAct [8] and Reflexion [2] can inadvertently store or process sensitive information, creating a significant attack surface.

Research by Carlini et al. [23] has demonstrated that it is possible to extract verbatim training data from LLMs, posing a severe privacy risk if the model was trained on personal or proprietary information.

The self-improvement loop could further amplify this risk by enhancing the model's ability to access or manipulate data beyond its original design [24].

Strong data governance, which includes robust protocols for data handling, anonymization and access control, is therefore not an optional add-on but a foundational requirement for these systems [25].

Furthermore, ethical issues such as bias and fairness are critical. An LLM's biases, inherited from its training data, can be subtly amplified through self-reflection. If a model's underlying data reflects societal biases, its self-generated critiques may fail to identify and correct these biases, potentially even reinforcing them as "correct" reasoning.

As noted by Müller [26] in the Stanford Encyclopedia of Philosophy, the ethical implications of AI are vast and the autonomy of these agents places greater responsibility on developers to explicitly ensure alignment with human values.

This has led to an emerging regulatory landscape. The European Union's proposed AI Act, for instance, classifies certain autonomous systems as "high-risk," imposing strict requirements for transparency, oversight and data quality [27].

The governance frameworks described by Pahune et al. [25] are becoming legal necessities, requiring organizations to document and justify their AI's decision-making processes.

The proactive safety and ethics evaluations detailed in resources like the GPT-4o system card represent the industry's response to these impending legal and ethical obligations [16].

Consequently, deploying an iterative agent framework is not merely a technical task but an exercise in navigating a complex ethical and regulatory environment.

5.3 Cost-Benefit Analysis in Business Applications

The practical adoption of iterative LLM agents in commercial settings ultimately hinges on a favorable cost-benefit analysis. While the potential benefits (such as hyper-automating complex workflows, enhancing decision-making and creating novel user experiences) are significant, they must be weighed against substantial and often recurring costs.

The "cost" side of the equation extends beyond simple software licensing. A primary driver is computational expense.

Frameworks like Voyager [10], which can require *hundreds* of GPT-4 API calls during a single run, entail non-trivial usage costs; the authors estimate about US $50 for just 160 iterations [28].

Furthermore, the infrastructure required to run, monitor and maintain these agents, along with the highly specialized human expertise needed to develop and oversee them, represents a major investment [29, 30].

Steinberg notes that finance functions usually judge technology on short-term, efficiency-based ROI, which makes it hard to justify AI initiatives whose pay-offs lie in harder-to-quantify benefits such as agility, competitiveness and reduced risk [30].

The "benefit" side, while compelling, can be more difficult to quantify. As Elbashir et al. [31] discuss in their work on business intelligence systems, the value derived from advanced information systems often manifests as improvements to organizational processes, which can be challenging to measure with traditional ROI metrics.

The success of an agent like Voyager [10], for example, is clear within Minecraft [17], but translating that into measurable business outcomes requires careful strategic alignment. The industry attention garnered by such advancements signals a strong belief in their transformative potential, but this must be validated with tangible results.

As Mathews highlights, a key challenge for enterprises is determining whether generative AI is worth the investment given these dynamics [29].

The analysis must consider not only direct cost savings or revenue generation but also second-order effects like improved innovation capacity, employee upskilling and competitive advantage. A business might justify the high cost of a ReAct-style agent not for automating a single task, but for building a platform that enhances the problem-solving capabilities of its entire research and development team. Therefore, investing in these advanced frameworks requires a sophisticated, forward-looking analysis that treats the technology not merely as a tool but as a strategic organizational capability.

6 Recent Developments, Conclusion and Future Directions

6.1 Recent Developments

The landscape of iterative improvement for LLM agents is evolving at a rapid pace. While the foundational frameworks discussed previously established the core principles of self-correction and tool use, recent research has pushed these concepts in new directions, focusing on more sophisticated architectures, specialized applications and critical evaluations of cost-effectiveness. This chapter highlights several key recent developments that are shaping the future of this field.

One of the most significant recent trends is the evolution from single-model self-reflection towards dual-model architectures that decouple the "Reasoner" (the task-doer) from the "Critic" (the feedback provider). This represents a direct architectural departure from the original Reflexion framework [2], which uses a single LLM to perform all roles. The DARS framework [32], for instance, proposes a specialized, trained Critic model that provides detailed, trace-level verbal feedback to a Reasoner model. This separation allows for more precise and actionable critiques, outperforming single-model preference optimization methods and addressing the potential conflict of interest inherent in a single model critiquing its own work [32].

Building on this, the theoretical Socratic-RL framework [33] proposes a "Teacher-Student" architecture. In this model, the "Teacher" not only provides feedback but its ability to teach is itself improved over time through a meta-learning process. Unlike Reflexion's [2] static self-critique capability, this creates a system that learns how to learn more effectively, representing a conceptual leap towards more sophisticated, automated process supervision [33].

Recent work has also focused on adapting reflective frameworks to real-world domains where performance is paramount. One notable example is the application of a self-reflective agent to the task of explainable stock prediction. While this approach is inspired by Reflexion's [2] iterative loop, the SEP framework [34] makes a critical modification: it uses the outputs of the self-reflective process to generate training data for a subsequent fine-tuning stage using Proximal Policy Optimization (PPO). This marks a fundamental difference in methodology; whereas Reflexion [2] improves

performance at inference time by changing the prompt's context without altering the model, SEP uses the reflective loop to permanently update the model's weights. This successful application in the chaotic domain of finance highlights the practical utility of reflective methods for generating high-quality training data [34].

While much research has focused on increasing the sophistication of verbal feedback, an emerging counter-argument questions the necessity of this complexity. Recent work introduces the concept of "retrials without feedback" [35] a mechanism where an LLM is allowed to retry a problem upon failure without any explicit verbal critique. This directly challenges the central hypothesis of Reflexion [2], which posits that the content of the self-critique is what drives learning [35].

Empirical results from Potamitis et al. [35] show that for certain tasks, simpler methods like Chain-of-Thought augmented with this retrial mechanism can outperform more complex frameworks like Reflexion [2] in terms of cost-efficiency. This suggests that for some problems, the overhead of generating and processing verbal feedback may not justify the marginal performance gain over simply allowing the model another stochastic attempt. This research forces a critical re-evaluation of whether improvement comes from intelligent feedback, as in Reflexion [2], or simply from more opportunities for exploration [35].

These recent developments paint a picture of a field that is both maturing and diversifying, largely by evolving or challenging the initial paradigm set by Reflexion [2]. The move towards decoupled Reasoner-Critic architectures points to a future of more specialized learning processes. Concurrently, the use of reflective loops for fine-tuning data generation and the compelling results from simple retrials highlight that verbal feedback is just one of several paths to improvement. The path forward will likely involve a hybridization of these approaches, where the choice of agent architecture, from a simple retrial loop to a complex, self-improving Teacher-Student model, will be tailored to the specific task's requirements for performance, interpretability and operational cost.

6.2 Synthesis of Key Findings

This paper's analysis of iterative improvement frameworks reveals several key findings that illuminate the current state and future trajectory of LLM-based agents. First and foremost, the empirical success of the Reflexion [2] framework validates verbal self-reflection as a powerful and viable paradigm. Its ability to achieve significant performance gains across diverse domains: from sequential decision-making to complex reasoning and code generation, without costly model retraining confirms that language itself can serve as an effective, explicit feedback mechanism for learning. This marks a significant departure from traditional reinforcement learning, demonstrating a more flexible and interpretable path to iterative improvement.

Second, our comparative analysis makes it clear that there is no universally superior architecture. The landscape of iterative agents is defined by a series of nuanced and critical trade-offs. Frameworks like Self-Refine [7] offer simplicity and resource efficiency, making them highly attractive for improving single-turn generative tasks, but they lack the memory and environmental interaction needed for sequential decision-making. ReAct [8] provides exceptional interpretability through its interleaved thought-action process, but its learning is confined to a single episode. Toolformer [9], in contrast, achieves specialized, high performance by teaching an LLM to use external tools, but this comes at the cost of a significant upfront fine-tuning phase and limited adaptability beyond its pre-defined toolset. Finally, Voyager [10] represents the pinnacle of autonomous, lifelong learning within a specific domain (till now only Minecraft), yet its immense architectural complexity and high operational cost make it impractical for general application. The optimal choice is therefore not a settled question but is contingent on a project's specific priorities, balancing performance needs against constraints in cost, complexity and interpretability.

Third, the practical application of these frameworks is fundamentally constrained by two significant challenges. The first is the reliability of self-generated critiques. As explored in Chapter 5, the "blind leading the blind" problem: where a model flawed enough to make an error may also be too flawed to critique it effectively, remains a core bottleneck. The second is the critical need for robust data governance and ethical oversight. The extensive memory and reasoning traces generated by these agents create significant privacy risks and raise profound questions about accountability and bias amplification, necessitating a proactive approach to safety and alignment.

6.3 Open Research Questions and Outlook

The progress in iterative improvement frameworks opens several promising and critical avenues for future research. A primary direction is the exploration of more sophisticated agent architectures designed to improve the quality and reliability of feedback. The recent trend towards decoupled "Reasoner-Critic" systems, as seen in the DARS framework [32], represents a significant step forward. By assigning the roles of task execution and critique to separate, specialized models, these architectures can overcome the inherent conflict of interest in single-model self-correction. The theoretical Socratic-RL framework [33] pushes this even further, proposing a "Teacher" model that not only provides feedback but also "learns how to teach" through a meta-learning loop, suggesting a path toward more efficient and targeted process supervision.

Concurrently, a crucial area of research will be a more rigorous cost-benefit analysis of verbal feedback itself. Emerging work that champions the efficacy of simple, unguided "retrials" directly challenges the central premise of frameworks like Reflexion [2]. This research forces a critical re-evaluation of whether performance gains stem from the intelligent content of the feedback or merely from providing the model with more stochastic opportunities to find a solution. Future hybrid models might dynamically choose between costly, complex reflection and cheaper, simpler retrials based on the specific task and context.

Furthermore, future work must focus on increasing the sample efficiency and reducing the operational latency of these iterative loops. This will likely involve research into advanced knowledge distillation techniques to compress the procedural knowledge gained from reflection into a model's parameters, as well as the development of more efficient memory structures that can store and retrieve lessons learned without exceeding context window limitations.

Finally, ensuring the safety and alignment of agents that can self-improve is paramount. Future frameworks must move beyond simply enhancing performance and begin to integrate formal verification and ethical principles directly into the reflection process itself.

REFERENCES

[1] R. S. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge, Massachusetts, London, England: The MIT Press, 2020.

[2] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," Mar. 2023. [Online]. Available: http://arxiv.org/pdf/2303.11366v4

[3] Z. Yang *et al.,* "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," Sep. 2018. [Online]. Available: http://arxiv.org/pdf/1809.09600v1

[4] M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht, "ALFWorld: Aligning Text and Embodied Environments for Interactive Learning," Oct. 2020. [Online]. Available: http://arxiv.org/pdf/2010.03768v2

[5] M. Renze and E. Guven, "Self-Reflection in LLM Agents: Effects on Problem-Solving Performance," vol. 35, pp. 476–483, 2024, doi: 10.1109/FLLM63129.2024.10852493.

[6] K. Se, *#12: How Do Agents Learn from Their Own Mistakes? The Role of Reflection in AI.* [Online]. Available: https://www.turingpost.com/p/reflection (accessed: Jun. 17 2025).

[7] A. Madaan *et al.,* "Self-Refine: Iterative Refinement with Self-Feedback," Mar. 2023. [Online]. Available: http://arxiv.org/pdf/2303.17651v2

[8] S. Yao *et al.,* "ReAct: Synergizing Reasoning and Acting in Language Models," Oct. 2022. [Online]. Available: http://arxiv.org/pdf/2210.03629v3

[9] T. Schick *et al.,* "Toolformer: Language Models Can Teach Themselves to Use Tools," in *Advances in Neural Information Processing Systems*, vol. 36, *Advances in Neural Information Processing Systems 36 (NeurIPS 2023). Proceedings of the 37th Conference on Neural Information Processing Systems*, A. Oh and T. Naumann and A. Globerson and K. Saenko and M. Hardt and S. Levine, Ed., Red Hook, NY: Neural Information Processing Systems (NeurIPS) Foundation. Accessed: Jun. 17 2025. [Online]. Available: https://arxiv.org/abs/2302.04761

[10] G. Wang *et al.,* "Voyager: An Open-Ended Embodied Agent with Large Language Models," May. 2023. [Online]. Available: http://arxiv.org/pdf/2305.16291v2

[11] LangChain, *Reflection Agents.* [Online]. Available: https://blog.langchain.dev/reflection-agents/ (accessed: Jun. 17 2025).

[12] DAIR.AI, *Reflexion.* [Online]. Available: https://www.promptingguide.ai/techniques/reflexion (accessed: Jun. 17 2025).

[13] S. Schulhoff *et al.,* "The Prompt Report: A Systematic Survey of Prompt Engineering Techniques," Jun. 2024. [Online]. Available: http://arxiv.org/pdf/2406.06608v6

[14] J. Kaur, *Reflection Agent Prompting: Strategies for More Efficient Performance.* [Online]. Available: https://www.akira.ai/blog/reflection-agent-prompting (accessed: Jun. 17 2025).

[15] DeepSeek-AI *et al.,* "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," Jan. 2025. [Online]. Available: http://arxiv.org/pdf/2501.12948v1

[16] OpenAI, "GPT-4o System Card: Technical report," OpenAI, San Francisco, CA, May. 2024. Accessed: Jun. 17 2025. [Online]. Available: https://openai.com/research/gpt-4o-system-card

[17] Daniel Dominguez, "Minecraft Welcomes Its First LLM-Powered Agent," *InfoQ*, 2023. [Online]. Available: https://www.infoq.com/news/2023/05/minecraft-voyager-llm-agent/

[18] Ben Wang and Aran Komatsuzaki, *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*, 2021. [Online]. Available: https://github.com/kingoflolz/mesh-transformer-jax

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei, "Language models are few-shot learners," in vol. 33, *Advances in Neural Information Processing Systems*, 2020, pp. 1877–1901. Accessed: Jun. 19 2025. [Online]. Available: https://arxiv.org/pdf/2005.14165

[20] Q. Dong *et al.,* "A Survey on In-context Learning," Dec. 2022. [Online]. Available: https://arxiv.org/abs/2301.00234

[21] J. Uesato *et al.,* "Solving math word problems with process- and outcome-based feedback," Nov. 2022. [Online]. Available: http://arxiv.org/pdf/2211.14275v1

[22] H. Lightman *et al.,* "Let's Verify Step by Step," May. 2023. [Online]. Available: http://arxiv.org/pdf/2305.20050v1

[23] N. Carlini *et al.,* "Extracting Training Data from Large Language Models," Dec. 2020. [Online]. Available: http://arxiv.org/pdf/2012.07805v2

[24] T. Green, M. Gubri, H. Puerto, S. Yun, and S. J. Oh, "Leaky Thoughts: Large Reasoning Models Are Not Private Thinkers," Jun. 2025. [Online]. Available: http://arxiv.org/pdf/2506.15674v1

[25] S. Pahune, Z. Akhtar, V. Mandapati, and K. Siddique, "The Importance of AI Data Governance in Large Language Models," *BDCC*, vol. 9, no. 6, p. 147, 2025, doi: 10.3390/bdcc9060147.

[26] V. C. Müller, *Ethics of Artificial Intelligence and Robotics.* [Online]. Available: https://plato.stanford.edu/entries/ethics-ai/ (accessed: Jun. 17 2025).

[27] *Proposal for a Regulation laying down harmonised rules on Artificial Intelligence (Artificial Intelligence Act): AI Act proposal*, 2021. Accessed: Jun. 17 2025. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206

[28] MineDojo, *Voyager FAQ*: GitHub, Inc., 2023. Accessed: Jun. 19 2025. [Online]. Available: https://github.com/MineDojo/Voyager/blob/main/FAQ.md

[29] A. Mathews. [Online]. Available: https://aimresearch.co/conference-videos/cost-benefit-analysis-is-generative-ai-worth-the-investment-for-enterprises (accessed: Jun. 17 2025).

[30] M. Steinberg, *Cost-benefit analysis of AI technology investments in Finance.* [Online]. Available: https://www.mindbridge.ai/blog/cost-benefit-analysis-of-ai-technology-investments-in-finance/ (accessed: Jun. 17 2025).

[31] M. Z. Elbashir, P. A. Collier, and M. J. Davern, "Measuring the effects of business intelligence systems: The relationship between business process and organizational performance," *International Journal of Accounting Information Systems*, vol. 9, no. 3, pp. 135–153, 2008, doi: 10.1016/j.accinf.2008.03.001.

[32] J. Li *et al.,* "Two Heads Are Better Than One: Dual-Model Verbal Reflection at Inference-Time," Feb. 2025. [Online]. Available: http://arxiv.org/pdf/2502.19230v1

[33] X. Wu, "Socratic RL: A Novel Framework for Efficient Knowledge Acquisition through Iterative Reflection and Viewpoint Distillation," Jun. 2025. [Online]. Available: http://arxiv.org/pdf/2506.13358v1

[34] K. J. L. Koa, Y. Ma, R. Ng, and T.-S. Chua, "Learning to Generate Explainable Stock Predictions using Self-Reflective Large Language Models," vol. 12706, pp. 4304–4315, 2024, doi: 10.1145/3589334.3645611.

[35] N. Potamitis and A. Arora, "Are Retrials All You Need? Enhancing Large Language Model Reasoning Without Verbalized Feedback," Apr. 2025. [Online]. Available: http://arxiv.org/pdf/2504.12951v1