ML-Master: Towards AI-for-AI via Intergration of Exploration and Reasoning

Zexi Liu[*], **Yuzhu Cai**[*], **Xinyu Zhu**[*], **Yujie Zheng**[*], **Runkun Chen**[*], **Siheng Chen**
School of Artificial Intelligence, Shanghai Jiao Tong University

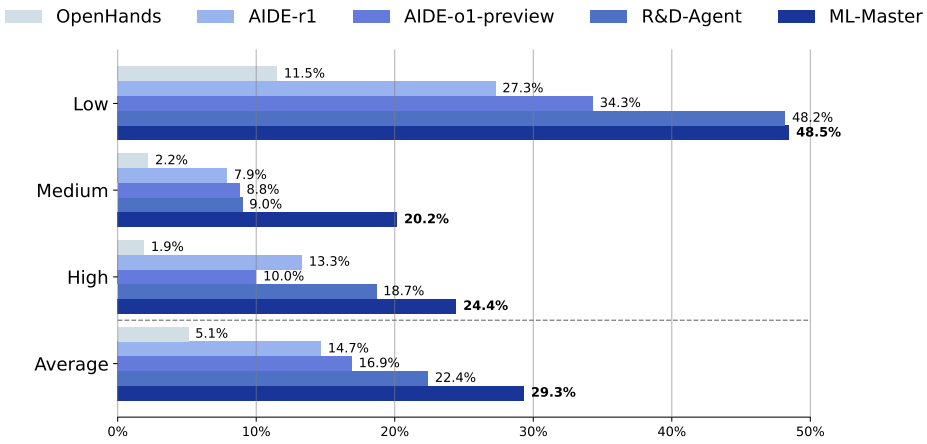

Figure 1: Performance of OpenHands [1], AIDE [2], R&D-Agent [3] and ML-Master on MLE-Bench [4].

Abstract

The pursuit of Artificial General Intelligence (AGI) requires overcoming significant limitations in current AI development, particularly the dependence AI practitioners for experimentation and iterative refinement. A promising pathway to accelerating the arrival of AGI is through AI-for-AI (AI4AI), where AI systems autonomously design, optimize, and evolve AI algorithms. While LLM-based agents have shown the potential, they are often unable to fully leverage the experience accumulated by agents during the exploration of solutions in the reasoning process, leading to inefficiencies and suboptimal performance. To address this limitation, we propose ML-Master, a novel AI4AI agent that seamlessly integrates exploration and reasoning by employing a selectively scoped memory mechanism. This approach allows ML-Master to efficiently combine diverse insights from parallel solution trajectories with analytical reasoning, guiding further exploration without overwhelming the agent with excessive context. We evaluate ML-Master on the MLE-Bench, where it achieves a 29.3% average medal rate, surpassing existing methods, particularly in medium-complexity tasks, while accomplishing this superior performance within a strict 12-hour time constraint—half the 24-hour limit used by previous baselines. These results demonstrate ML-Master's potential as a powerful tool for advancing AI4AI and accelerating the development of AGI.

[*]Equal contribution. Order randomized.

1 Introduction

"Recursive self-improvement — a process in which AI algorithms are able to learn on their own and increase their capabilities, is already underway." — Eric Schmidt, Former CEO of Google

Artificial Intelligence (AI) has profoundly reshaped human civilization, driving transformative advancements across diverse areas such as healthcare, finance, education, and industry. While contemporary AI systems have already demonstrated substantial societal and economic impacts [5, 6, 7, 8], these achievements merely represent the initial phases of a far more extensive and dynamic evolutionary trajectory. As AI continues its rapid development, this trajectory promises possibilities far beyond our current understanding, harboring immense potential that could fundamentally redefine human capability, productivity, and creativity.

Despite the rapid evolution of AI, the current development paradigm remains fundamentally constrained by its reliance on human expertise, involving extensive manual experimentation, iterative refinement, and parameter tuning, which are inherently labor-intensive, and time-consuming. This human-centric approach creates significant bottlenecks, severely limiting the scale, pace and depth of AI innovation. Moreover, as AI advances towards and potentially surpasses human-level intelligence, these human-driven methodologies will inevitably become insufficient, impeding AI's continuous evolution and hindering the realization of its full potential. To overcome these limitations, a novel paradigm, *AI-for-AI* (AI4AI), has emerged. AI4AI refers to AI systems capable of autonomously designing, optimizing, and iteratively evolving their own algorithms and cognitive strategies, effectively minimizing human intervention in the AI development. [9, 10] Central to this paradigm is the endowment of AI with intrinsic self-evolving capacities, enabling continuous autonomous exploration, experimentation, and refinement without dependence on persistent human oversight. So that AI4AI can initiate a self-sustaining cycle of continuous intellectual growth.

In practice, developing effective AI solutions is inherently an iterative and exploratory process. AI practitioners naturally integrate exploration and reasoning into a cohesive cognitive methodology. Specifically, exploration entails actively seeking new insights through various experimentation and discovery [2], while reasoning involves carefully analyzing existing knowledge and reflecting upon past experiences [11, 12]. Neither alone is sufficient: exploration without reasoning can lead to inefficiency and aimless trial-and-error, while reasoning without exploration risks stagnation. Instead, effective problem-solving emerges from a harmonious interplay between exploration and reasoning, where new insights gained through exploration continuously enrich and refine subsequent reasoning processes. This iterative cycle of purposeful exploration and thoughtful reasoning forms the foundation of continuous improvement and innovation in human-driven AI development, motivating the need for AI4AI frameworks that similarly integrate these complementary cognitive strategies.

Recent breakthroughs in Large Language Models (LLMs) [11, 12, 13] have provide convincible evidence supporting the feasibility of AI4AI. These powerful models have demonstrated the capability to serve as autonomous agents, independently performing tasks traditionally reserved for human experts. [14, 15, 1, 16] However, existing AI4AI methods have struggled to effectively integrate exploration and reasoning, primarily because exploration processes often fail to sufficiently distill past experiences to generate promising solutions. Additionally, reasoning models find it challenging to effectively utilize the extensive and unstructured experiences accumulated during exploration, as overly long contexts can overwhelm the reasoning process, leading to hallucinations and unreliable outputs. Most studies [17, 18, 19, 20, 21] like AI Scientist [17], SELA [18], and Dophin [20] primarily emphasize exploration strategies without sufficiently leveraging the analytical reasoning capabilities of advanced reasoning models, thus missing valuable insights and limiting their adaptability in complex scenarios. Conversely, other works such as AIDE [2] and Agent Laboratory [16] attempt to utilize the reasoning capabilities, but their exploration strategies are often inefficient or insufficiently comprehensive, leading to hallucinations, unreliable outputs, and suboptimal performance.

To address this critical limitation, we introduce **ML-Master**, a novel AI4AI agent inspired by the unified cognitive strategies of expert AI developers, which addresses the challenge of integrating exploration and reasoning by employing an adaptively scoped memory mechanism. ML-Master integrates exploration and reasoning into a coherent iterative methodology, where each component mutually reinforces the other without compromising either. Specifically, ML-Master simultaneously leverages the analytical learning capabilities of reasoning models and comprehensive, efficient exploration strategies, forming a virtuous cycle of continuous improvement. Within this unified cognition, ML-Master comprises two complementary and mutually supportive modules: (1) **Multi-**

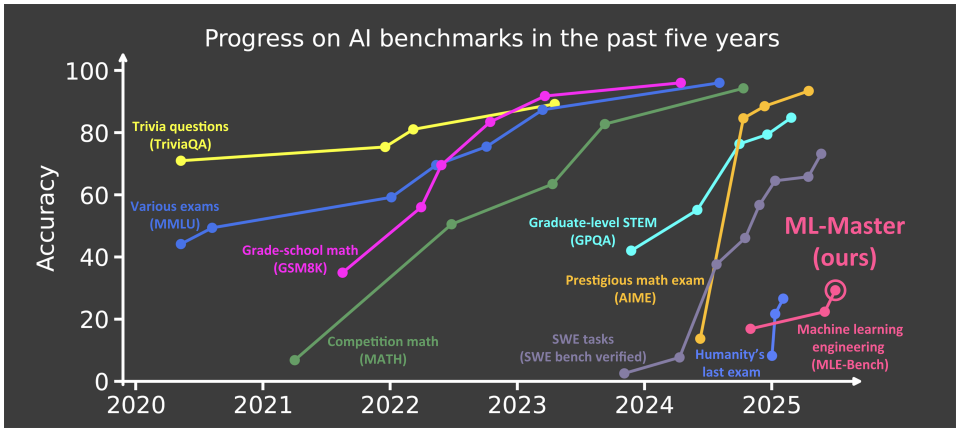

Figure 2: Positioned at the frontier of AI-for-AI progress, ML-Master demonstrates notable advances in autonomous machine learning engineering, contributing to the fast-evolving path toward AGI [22].

trace exploration empowers ML-Master to explore multiple solution trajectories step by step in parallel, actively generating diverse experiences and insights. These exploratory outcomes enrich the reasoning process by providing concrete execution feedback and new knowledge, enabling more informed and accurate analytical reasoning in subsequent iterations. This parallel exploration optimizes exploration strategies, efficiently manages contextual memory, and ensures comprehensive coverage of diverse solution paths. (2) **Steerable reasoning** enhances the reasoning capabilities of an advanced reasoning model (Deepseek-R1 [11]) by explicitly embedding contextual memory into the reasoning process. This integration ensures precise, reliable, and controlled analytical capabilities, significantly reducing hallucinations and erroneous interpretations commonly observed in LLM-based agents. The insights derived from this reasoning process directly inform and guide exploration steps. By organically combining multi-trace exploration and steerable reasoning within a unified iterative methodology, ML-Master achieves robust, efficient, and high-performing AI4AI.

We benched ML-Master on the widely recognized MLE-Bench [4], achieving state-of-the-art results that significantly surpass existing methods across multiple evaluation metrics. MLE-Bench, introduced by OpenAI, is a comprehensive benchmark designed to evaluate systems on challenging real-world machine learning tasks derived from Kaggle competitions. As shown in Figure 1, we measure performance using the average medal rate, defined as the percentage of tasks for which the method achieves Bronze, Silver, or Gold-level performance. ML-Master achieved an average medal rate of 29.3%, substantially outperforming the strongest baseline, R&D-Agent [23], which achieved a medal rate of 22.4%. Furthermore, ML-Master excels particularly in tasks categorized as medium difficulty, attaining an impressive medal rate of 20.2%, more than doubling the previous best result of 9.0%, clearly demonstrating its superior capability in handling complex and challenging AI development scenarios. Notably, ML-Master accomplished this superior performance within a strict time constraint of only 12 hours, merely half of the 24-hour limit previously employed by baselines. Results on MLE-Bench underscore ML-Master's consistent superiority across various evaluation dimensions. This substantial margin highlights ML-Master's ability to handle complex AI development tasks with remarkable efficiency and accuracy, further establishing its superiority in AI4AI.

In summary, our contributions are as follows:

- We propose ML-Master, a novel AI4AI agent that seamlessly integrates comprehensive exploration and analytical reasoning into a unified iterative methodology, inspired by the cognitive strategies of expert AI developers.
- We achieve SOTA performance on the MLE-Bench, attaining an average medal rate of 29.3% and excelling particularly in complex, medium-difficulty tasks, where we more than double the previous best result with a medal rate of 20.2%.
- Remarkably, ML-Master delivers this exceptional performance with less computational cost than previous methods, requiring only 12 hours—half the time limit set by earlier approaches.

Table 1: Comparison of ML-Master with existing AI4AI methods.

Method	Exploration Strategy	Reasoning Enhancement	Is Memory Adaptive?	Exploration-Reasoning Integration
MLAB [19]	Single-trace,	✗	✗	✗
OpenHands [1]	LLM-driven	✗	✗	✗
SELA [18]	MCTS	✗	✗	✗
AIDE [2]	Greedy	✔	✗	✗
Agent Laboratory [16]	LLM-driven	✔	✗	✗
R&D-Agent [23]	LLM-driven	✔	✗	✗
ML-Master (ours)	**Multi-trace**	✔	✔	✔

2 Related Work

Automated machine learning (AutoML). AutoML is the study of automating the tasks of machine learning engineering. By streamlining model development through both heuristic and learning-based approaches, AutoML serves as an early step towards the broader vision of AI4AI. Before the advent of LLMs, AutoML research have mainly focused on the repetitive and labor-intensive aspects of machine learning, such as data preprocessing, model selection and parameter tuning. [24, 25, 26, 27, 28, 29] For example, AutoGluon-Tabular [28] automates emsemble learning to fit end-to-end pipeline on tabular data with minimal user inputs. MOSAIC [30] combines tree search with Bayesian optimization to effectively search for optimal model architecture and hyperparameter. However, these works have all relied upon heuristic methods, lacking adaptability and generalization abilities. Other works have used learning-based methods to optimize hyperparameters and select model architecture [9, 31, 10, 32, 33]; for example, Zoph and Le [9] trained an RNN to search for optimal neural networks on the CIFAR-10 dataset, rivaling the performance of human-designed models. However, these works still require human to pre-define the pipeline before training, and are often restricted to specific tasks and datasets. Overall, classical AutoML works remain constrained by predefined search spaces and static configurations, lacking the adaptability and capabilities for continuous learning.

LLMs and multi-agent systems for AI4AI. Recent advancements in LLMs have unlocked a wide range of new opportunities for AI4AI. In contrast to earlier AutoML systems, LLMs are capable of complex reasoning, knowledge-based judgment and code generation. [11, 34, 35, 36] These advanced capabilities enable LLM-based systems to act freely and self-improve with minimal human intervention, thereby representing a significant step toward AI4AI. For example, early works like AutoML-GPT [14] and MLCopilot [37] exploit LLM's strong capabilities through prompt engineering to automate the entire machine learning pipeline. Recent works have focused on LLM-based multi-agent system (MAS) frameworks: AutoKaggle [38] and Agent K [39] designs a fully-automated multi-agent framework modeled after human engineering process to allow LLM agents to compete in Kaggle competitions; MLAgentBench [19] and MLZero [40] introduces external tools and memory to enhance LLM agents in AI research and development. Dolphin [20] introduces retrieval-augmented generation and an iterative refinement strategy to enhance the idea proposal process. Some recent works such as Agent Laboratory [16], NovelSeek [41], the AI Scientist [17] and AI-Researcher [42] have gone further to automate the entire AI research process from idea proposal to code implementation, validation and paper generation. While powerful, due to limitations in pipeline design, LLM-based multi-agent systems often struggle with insufficient exploration, a significant drawback considering the immense scale and complexity of machine learning problems.

LLM-driven tree search. Tree search algorithms have become integral to advancing AI4AI by enabling efficient and complete exploration of complex search spaces proposed by LLM Agents. For instance, AIDE [2] focuses on optimizing the machine learning engineering process through iterative tree search and refinement strategies, employing trial-and-error strategies to explore potential solutions. But it simply uses greedy strategy at the cost of search efficiency and neglect of global optimum. Recent developments [18, 43, 44] have integrated MCTS with LLM Agents to improve performance in machine learning tasks. While SELA [18] utilize MCTS as a better policy to iteratively refine pipeline configurations, it restricts searching to pre-defined sections and can not handle complex machine learning tasks.

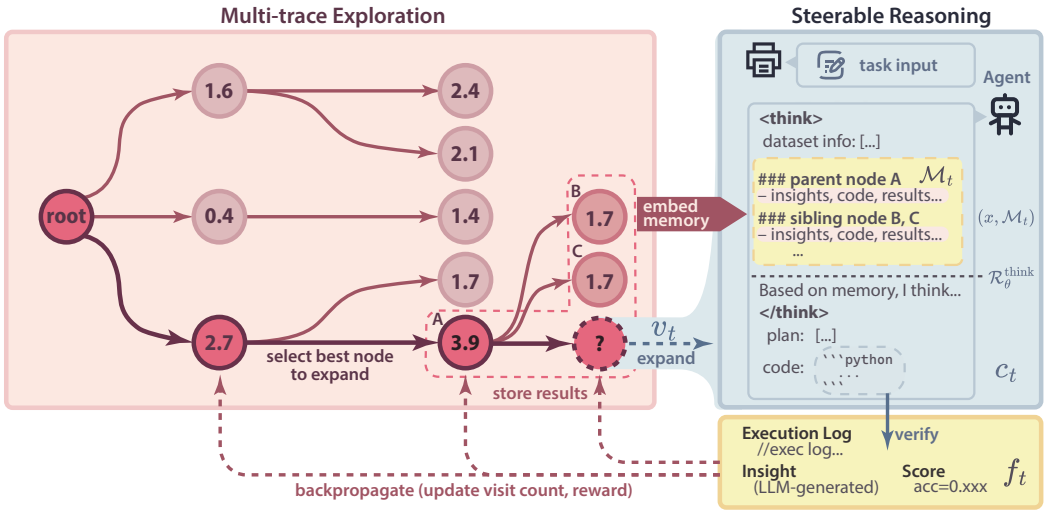

Figure 3: An overview of ML-Master's two modules: **multi-trace exploration** and **steerable reasoning**. Several draft solutions (nodes) are initialized at the start of each run. In each exploration step, an LLM agent is prompted to either improve or debug a previous solution, expanding from different draft nodes following an MCTS-inspired approach, enhanced with parallelism to ensure efficiency. Memories and insights from all explored branches are then fed into the LLM agent's reasoning process, leading to more steerable reasoning and higher performance.

3 Methodology

The development of robust AI systems, particularly within the AI4AI paradigm, demands more than isolated competence in either exploration or reasoning. Instead, it necessitates a unified methodology in which exploration and reasoning interact synergistically. Exploration empowers the agent to traverse diverse solution paths, generate new knowledge, and adapt to the uncertainties inherent in complex problem spaces. Reasoning, by contrast, enables the agent to interpret, evaluate, and synthesize information with coherence and precision, guiding it toward logically grounded solutions.

While reasoning provides direction and theoretical rigor, exploration contributes empirical grounding and diversity. In isolation, each is limited: (1) Exploration alone may devolve into inefficient trial-and-error. (2) Reasoning alone risks analytical stagnation when confined to prior knowledge. Therefore, the integration of these two cognitive faculties is essential for enabling autonomous agents to achieve both depth and breadth in AI development.

To realize this integration, we propose ML-Master, a tightly coupled iterative agent combining Steerable Reasoning module and Multi-trace Exploration module through a adaptively scoped memory machanisim. As shown in Figure 3, Multi-trace Exploration (§ 3.1) systematically generates and evaluates multiple solution trajectories in parallel, enriching the reasoning process with diverse empirical insights. This parallel exploration actively produces diverse empirical insights and execution feedback, which are adaptively captured and structured into a concise memory buffer. Rather than overwhelming the reasoning process with extensive and redundant historical data, this memory buffer strategically retains only the most relevant and actionable insights derived from exploration. Concurrently, Steerable Reasoning (§ 3.2) embeds contextual memory within the reasoning process, ensuring structured, context-sensitive inference and reducing hallucinations. The reasoning module thus effectively interprets exploration outcomes, identifies promising solution directions for subsequent exploration. Together, these two modules form a closed-loop system, where exploration continuously enriches reasoning with empirical insights, and reasoning systematically directs exploration toward promising trajectories. This iterative interplay, guided by the adaptively scoped memory mechanism, enables ML-Master to progressively refine its solutions, robustly navigate complex problem spaces, and achieve superior performance in AI4AI.

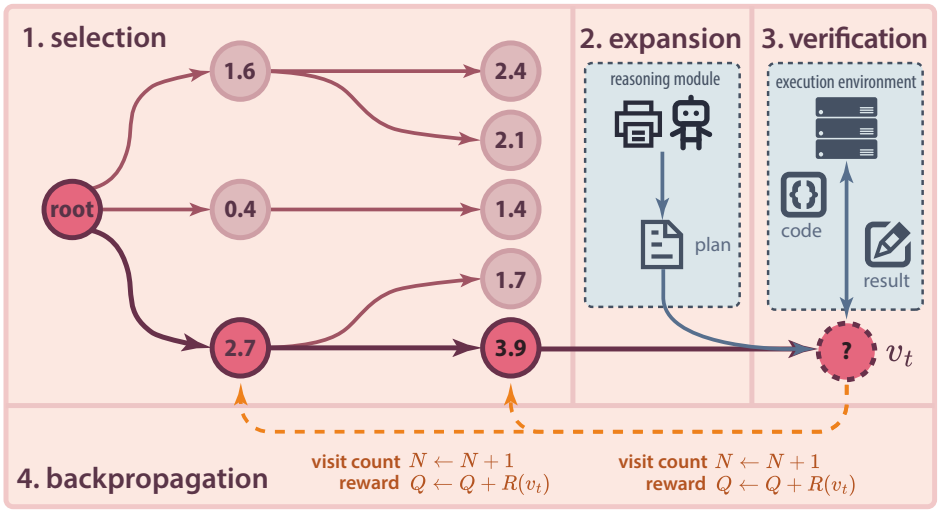

Figure 4: Multi-trace exploration pipelines operates through a tree-guided exploration with test-time-scaling search. It combines MCTS-inspired tree search with parallel exploration: (1) selection traverses from root to leaf using UCT criterion; (2) expansion generates child nodes through the steerable reasoning module; (3) verification evaluates solutions in the execution environment; (4) backpropagation propagates rewards and visit counts upward. Multiple workers explore different branches asynchronously, with top-k nodes serving as entry points for deeper parallel search.

3.1 Multi-Trace Exploration

In this section, we present the Multi-Trace Exploration module in ML-Master, which is designed to efficiently explore multiple solution trajectories in parallel. Given the complexity and scale of the search space in AI4AI tasks, it is essential to balance exploration breadth and depth effectively. As illustrated in Figure 4, our multi-trace exploration module adopts a structured, tree-based approach to guide exploration strategically, inspired by Monte Carlo Tree Search (MCTS). Specifically, Multi-Trace Exploration consists of two complementary components: (1) Tree-Guided Exploration, which reformulates the AI development process as an iterative exploration of potential solutions, using MCTS to efficiently navigate the solution space by constructing and expanding a search tree. This allows ML-Master to prioritize under-explored solution paths and dynamically adjust exploration efforts. (2) Test-Time-Scaling Search, which allows for concurrently exploration of multiple branches within the search tree, significantly improving both the efficiency and scalability of the search process. In the following subsections, we detail these two components and explain how their integration enables comprehensive and efficient exploration, ultimately enriching the reasoning process with diverse empirical insights.

3.1.1 Tree-Guided Exploration

To comprehensively explore the vast and complex solution space inherent in AI development tasks, we propose, for the first time, a novel formulation that explicitly models the AI development process as a Monte Carlo Tree Search (MCTS). Specifically, ML-Master constructs and expands a structured search tree, where each node represents a distinct solution state, and edges correspond to specific refinement actions. By leveraging a tree-based structure, ML-Master can efficiently manage and prioritize exploration efforts, ensuring comprehensive coverage of diverse solution trajectories while maintaining computational efficiency.

Specifically, the exploration process involves four key phases: (a) selection, where ML-Master employs a novel context-aware criterion to efficiently prioritize promising yet under-explored solution states; (b) expansion, where ML-Master takes specialized refinement actions uniquely tailored to AI development tasks; (c) verification, where ML-Master adopts a reward function to accurately and efficiently assess the quality of candidate solutions, and (d) backpropagation, where ML-Master

propagate structured evaluation feedback through the search tree, dynamically guiding subsequent exploration decisions. They are executed iteratively and in parallel across multiple solution paths.

Selection. The selection process begins at the root node and recursively selects the child node according to a context-aware Upper Confidence Bound for Trees (UCT) value. Formally, the UCT value for node v is defined as:

$$\text{UCT}(v) = \frac{Q_v}{N_v} + C \cdot \sqrt{\frac{\ln N_{\text{parent}}}{N_v}}, \tag{1}$$

where Q_v is the total reward of node v, N_v is the visit count of node v, N_{parent} is the total number of visits to the parent node, and C is a constant controlling the exploration-exploitation trade-off. Nodes with higher UCT values represent promising yet under-explored solution paths, thus guiding the exploration towards potentially valuable regions of the solution space.

During the selection phase of MCTS, a node is treated as terminal and excluded from further expansion if it satisfies any of the following stopping conditions:

First, we define an **improvement-based termination** criterion to eliminate nodes that show persistent stagnation. Let Δ_i denote the relative improvement over the best ancestor node on the current path after the i-th improve operation. If the number of failed improvements—i.e., those not exceeding a predefined threshold t—exceeds a tolerance level τ_{improve}, the node is considered terminal:

$$\sum_{i=1}^{k} \mathbb{I}[\Delta_i < t] > \tau_{\text{improve}}, \tag{2}$$

where $\mathbb{I}[\cdot]$ is the indicator function and k is the number of Improve attempts made at the node.

Second, we enforce a **debugging depth constraint** to prevent the search from persistently attempting to fix nodes.If the number of consecutive debug operations up to the current node exceeds τ_{debug}, the node is marked as terminal.

These two constraints jointly act as a pruning mechanism to suppress unproductive search trajectories and allocate computational resources toward more promising regions of the solution space.

Expansion. The expansion process starts from the selected node and applies three types of actions to generate new child nodes. These actions—Draft, Debug, and Improve—are designed to guide the search toward higher-quality solutions by addressing different aspects of code refinement and generation:

- **Draft**: The Draft action generates an initial, runnable code solution for node. Its primary function is to produce a basic implementation that satisfies the task requirements, serving as a starting point for further refinement.
- **Debug**: The Debug action focuses on identifying and correcting the errors in the current code. Its main role is to ensure that the node's code compiles and executes correctly before any further enhancements.
- **Improve**: The Improve action enhances the quality of functionally correct code by tune data preprocessing, model architecture, or optimization approaches. Its purpose is to generate a node that incorporates these refinements and achieves quantifiable performance gains.

For each node v, we provide a formal specification of the decision rule that determines which action to apply at each step. The selection is governed by the following condition:

- If there is no existing solution for the task in node v, the next step is taking the action **Draft** to draft a new one.
- If node v contains a solution that still has bugs, and debugging is not yet complete, the next step is taking the action **Debug** to identify and fix the remaining issues.
- If the solution in node v is currently bug-free, but further improvement is still needed, the next step is taking the action **Improve** to enhance its performance.

To allow iterative progress, we define stopping conditions for both the debugging and improving phases. The debugging process ends when either a correct solution has been produced or the number

of debugging attempts exceeds a predefined limit. The improving process terminates once no further progress can be made. The design supports iterative refinement while bounding the number of attempts, striking a balance between thoroughness and efficiency.

Verification. The verification process evaluates the quality of the newly expanded node v by computing a reward signal that reflects its effectiveness in addressing the task objectives. The reward function $R(v)$ is defined as:

$$R(v) = \begin{cases} -1, & \text{if } \mathcal{D}(v) \\ r_q(v) + r_d(v) + r_s(v), & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{D}(v)$ determines whether node v contains defects and the reward components are defined as follows:

- Quality reward $r_q(v)$: Indicates whether the solution represented by v improves upon the best evaluation metric observed so far. Formally:

$$r_q(v) = \begin{cases} 1, & \text{if } M(v) > M^* \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

  Here, $M(v)$ denotes the evaluation metric used to assess the solution quality at node v, and M^* is the best score observed so far during the search.
- Debugging reward $r_d(v)$: Reflects whether the transition from the parent node to v successfully eliminates a previously identified fault. That is, $r_d(v) = 1$ if the fault present in the parent node is resolved in v; otherwise, $r_d(v) = 0$.
- Structural improvement reward $r_s(v)$:This term reflects whether the node v represents the successful completion of an improvement process. Specifically, if v satisfies a predefined stopping criterion $\mathcal{T}_i(v)$, which signals that the stopping condition for the improvement process has been met, then $r_s(v) = 1$; otherwise, $r_s(v) = 0$.

Backpropagation. After the verification phase, the obtained reward is propagated back along the path from the expanded node to the root. During this backpropagation process, each node along the path updates its visit count N and accumulated reward Q accordingly.

3.1.2 Test-Time-Scaling Search

To effectively scale Monte Carlo Tree Search (MCTS) to parallel environments and large search spaces, we aim to design a search framework that enables asynchronous exploration across promising subregions of the tree while preserving the core principles of MCTS.

To this end, we introduce an asynchronous branch-parallel MCTS strategy. The search begins with all workers jointly expanding the root node in parallel. Once the root's children are fully expanded, the top-k nodes with the highest UCT values are selected as new entry points for deeper search. Each selected node then initiates an independent search thread, allowing selection, expansion, verification, and backpropagation to proceed asynchronously and without cross-thread interference.

When a thread completes its search within a branch, it returns to the root and selects the best available child node—based on UCT score—from among those not currently being explored by other threads. This mechanism ensures that parallel exploration proceeds efficiently without duplication, and that computational resources are dynamically reallocated to promising yet unoccupied regions of the search tree.

The proposed scheme enables broad and adaptive exploration of the search space while maintaining consistency with the MCTS framework. It is particularly well-suited to tasks with large branching factors or non-uniform value distributions among subtrees.

3.2 Steerable Reasoning

Steerable Reasoning is essential to any AI4AI that seeks to improve through self-guided iteration. In ML-Master, we significantly enhance the reasoning capabilities of the advanced reasoning model (Deepseek-R1 [11]) by explicitly embedding contextual memory directly into the reasoning (`"think"`) component, rather than into the instruction component. Specifically, it adaptively leverages only

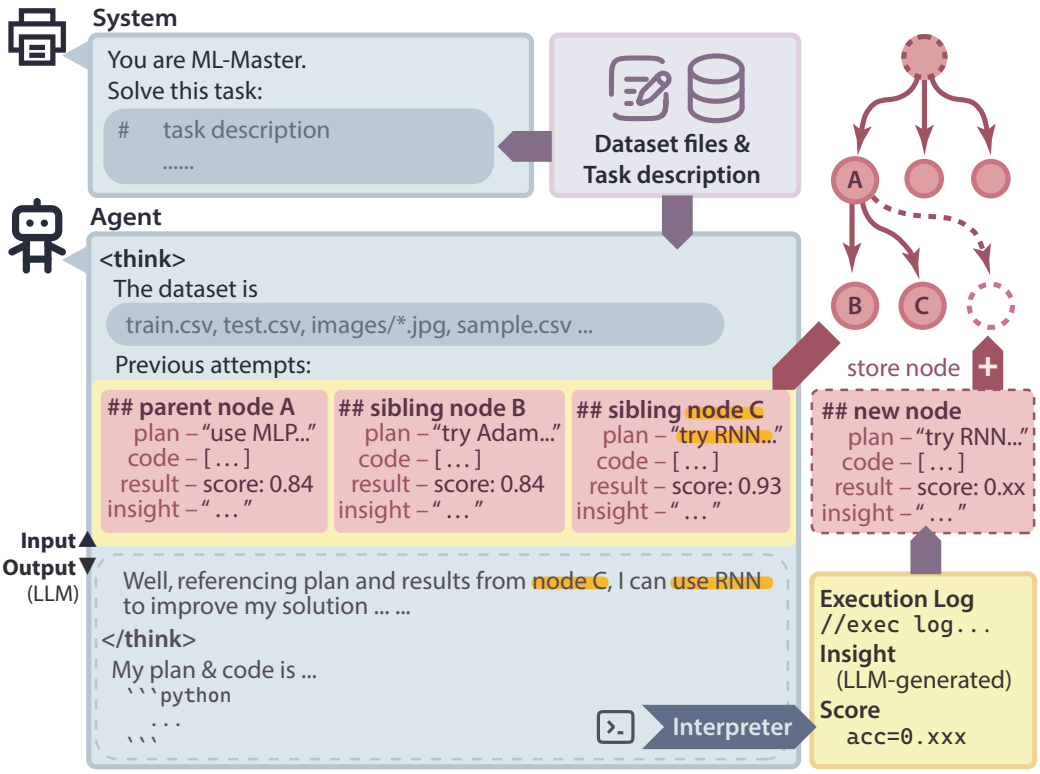

Figure 5: Steerable reasoning with adaptive memory. The memory buffer aggregates insights from the immediate parent node and parallel sibling nodes at the same exploration depth, containing execution results, code snippets, and performance scores. During reasoning, this curated memory is explicitly embedded into the LLM's "`think`" component, enabling contextually grounded decision-making while avoiding information overload. The generated plan and code are executed through an interpreter, producing execution logs, submission files, and performance scores that are saved back to the search tree, forming a closed-loop learning system.

the immediate previous reasoning node and parallel sibling nodes within the same exploration depth. This targeted memory mechanism ensures precise, coherent, and contextually grounded reasoning, effectively reducing hallucinations and redundancy. By dynamically integrating concise execution feedback, ML-Master continuously learns from past experiences, enabling robust analysis, effective debugging, and informed decision-making, thereby substantially improving the reliability and performance of AI4AI.

Formally, at each iteration node t, ML-Master performs reasoning based on its current task input x. To facilitate controlled reasoning, we first construct a memory buffer $\mathcal{M}_t$ that encapsulates relevant context from previous execution nodes while avoiding redundant trajectories. This memory buffer is then directly embedded into the reasoning part to guide the LLM's reasoning process from the outset. The memory buffer $\mathcal{M}_t$ is defined as:

$$\mathcal{M}_t = \{(c_{t-1}, f_{t-1})\} \cup \left\{ (c_t^{(s)}, f_t^{(s)}) \;\middle|\; s \in \mathcal{S}_t \right\}, \tag{5}$$

where (c_{t-1}, f_{t-1}) denotes reasoning trace, and feedback from the immediately preceding node $t-1$ within the current exploration branch. $\mathcal{S}_t$ is the set of sibling nodes at the same depth within parallel exploration branches, with each sibling node s providing alternative reasoning contexts $(c_t^{(s)}, f_t^{(s)})$. By incorporating both immediate historical context and parallel alternative reasoning paths, the memory buffer $\mathcal{M}_t$ provides comprehensive contextual grounding for the reasoning process.

Given this memory buffer, the reasoning trace c_t at node t is generated by conditioning the LLM $\mathcal{R}_\theta$ on the current input x and the memory buffer $\mathcal{M}_t$. Specifically, we denote the reasoning node as

follows:

$$c_t = \mathcal{R}_\theta^{think}(x, \mathcal{M}_t), \tag{6}$$

where $\mathcal{R}_\theta^{think}$ explicitly indicates the reasoning ("`think`") component of the reasoning LLM, conditioned on the input x and the memory buffer $\mathcal{M}_t$. The inclusion of sibling-node information introduces *contrastive signals*, allowing the model to recognize and avoid producing reasoning paths that mirror those already explored in parallel. Meanwhile, the direct lineage from the previous node ensures logical continuity and progressive refinement within each branch.

By explicitly embedding contextual memory into the LLM reasoning component, ML-Master achieves Steerable reasoning that is both contextually coherent and diversified across parallel exploration paths. This controlled integration significantly reduces common pitfalls such as hallucinations, redundant reasoning, and convergence to suboptimal solutions, thereby enhancing the robustness and effectiveness of the AI4AI.

3.3 Discussions

Rationality and Advantages of Integrating Exploration and Reasoning. ML-Master's integration of exploration and reasoning mirrors expert AI developers, combining empirical experimentation with analytical reasoning to iteratively refine solutions. Unlike existing AI4AI methods, which often favor either exploration or reasoning, ML-Master's Multi-Trace Exploration and Steerable Reasoning, linked via a adaptively scoped memory mechanism, create a virtuous cycle. Exploration provides diverse insights, while reasoning guides efficient solution refinement, reducing hallucinations and boosting reliability. This unified approach enables ML-Master to navigate complex AI tasks effectively, scaling human-like expertise to accelerate AGI development.

Comparison with Existing AI4AI Methods. Table 1 highlights ML-Master's superiority over existing AI4AI frameworks. Methods like MLAB[19] and OpenHands[1] focus on LLM-driven exploration but lack reasoning enhancements, limiting adaptability. SELA[18] uses MCTS but restricts exploration and ignores reasoning LLMs. AIDE[2], Agent Laboratory[16], and R&D-Agent [23] incorporate reasoning but suffer from inefficient exploration and fixed memory, leading to unreliable outputs. In contrast, ML-Master integrates exploration and reasoning into a coherent iterative methodology, where each component mutually reinforces the other without compromising either.

4 Experiment

4.1 Experiment Setup

Settings. Our ML-Master agent is tested on MLE-Bench [4], a diverse and realistic benchmark introduced by OpenAI to assess AI agents on end-to-end machine learning engineering tasks. The following configuration is informed by the algorithmic components introduced earlier. The predefined threshold t is used to determine whether an improve action is successful. To tolerate occasional stagnation, the process allows up to τ_{improve} consecutive failed improve attempts. In our experiments, we use $t = 0.001$ and $\tau_{\text{improve}} = 3$. To prevent persistent attempts at fixing nodes, a debug depth constraint $\tau_{\text{debug}} = 20$ is enforced: nodes exceeding this limit are marked terminal. Additionally, each debug action sequence ends once it reaches a depth of 3, thereby enabling iterative progress. The initially released first-version DeepSeek-R1-0120 [45] is employed to generate plans and code for the corresponding actions. The overall search procedure is executed in parallel at the MCTS level, with a parallelism degree of 3.

Environment. We try our best to ensure the same testing environment with AIDE. However, due to the limitation of hardware, there are still some differences. In our experiments, each agent is equipped with 36 AMD EPYC vCPUs and one NVIDIA A100 Tensor Core GPU(with only 24GB memory available). Every three agent share 512GB memory and 1TB SSD to produce submissions and any intermediate files. The total time of a task is set to 12 hours. Overall, our testing environment is slightly inferior to the one reported by MLE-Bench.

Baselines. To provide a comprehensive comparision, we compare ML-Master with OpenHands [1], MLAB [19], AIDE [2] and R&D-Agent [3]. Due to the expensive cost of running complete MLE-Bench, we use some results report by MLE-Bench itself. We run AIDE, which achieved the best

Table 2: ML-Master outperforms all baselines on all evaluation dimensions defined by MLE-Bench. The results of MLAB, OpenHands, AIDE (gpt-4o-2024-08-06 and o1-preview) and R&D-Agent are report by official MLE-Bench. Results for ML-Master are averaged over 3 runs with different random seeds and reported as mean ± one standard error of the mean (SEM). The top-performing model is shown in **bold**. * indicates that only a single run is conducted due to time and resource constraints.

Agent	Valid Submission (%)	Above Median (%)	Bronze (%)	Silver (%)	Gold (%)	Any Medal (%)
MLAB [19]						
gpt-4o-2024-08-06	44.3 ± 2.6	1.9 ± 0.7	0.0 ± 0.0	0.0 ± 0.0	0.8 ± 0.5	0.8 ± 0.5
OpenHands [1]						
gpt-4o-2024-08-06	52.0 ± 3.3	7.1 ± 1.7	0.4 ± 0.4	1.3 ± 0.8	2.7 ± 1.1	4.4 ± 1.4
AIDE [2]						
gpt-4o-2024-08-06	54.9 ± 1.0	14.4 ± 0.7	1.6 ± 0.2	2.2 ± 0.3	5.0 ± 0.4	8.7 ± 0.5
o1-preview	82.8 ± 1.1	29.4 ± 1.3	3.4 ± 0.5	4.1 ± 0.6	9.4 ± 0.8	16.9 ± 1.1
Deepseek-R1*	78.6 ± 0.0	34.6 ± 0.0	2.7 ± 0.0	4.0 ± 0.0	8.0 ± 0.0	14.7 ± 0.0
R&D-Agent [3]						
o1-preview	86.1 ± 1.1	32.8 ± 1.2	3.5 ± 0.5	4.5 ± 0.5	14.4 ± 0.5	22.4 ± 0.5
ML-Master						
Deepseek-R1	**93.3 ± 1.3**	**44.9 ± 1.2**	**4.4 ± 0.9**	**7.6 ± 0.4**	**17.3 ± 0.8**	**29.3 ± 0.8**

performance on MLE-Bench prior to our approach, with the initially released first-version Deepseek-R1-0120 [45] additionally to provide a fair comparison between ML-Master and AIDE.

4.2 Main Results

We evaluate our ML-Master on complete MLE-Bench among 75 machine learning tasks. ML-Master is compared against 4 methods driven by 3 models. We use the same evaluation metric as MLE-Bench. Results are shown in Table 2 and demonstrates that:

- **ML-Master achieves a medal on 29.3% machine learning tasks, including 17.3% gold medals.** This indicates that ML-Master exhibits a remarkably high upper bound in performance and exceeds most human machine learning researchers when addressing specific machine learning tasks.

- **ML-Master makes a valid submission on 93.3% tasks and its submission outperforms more than half human submissions on 44.9% tasks.** This indicates that ML-Master has a very high lower bound when handling a wide variety of machine learning tasks of various difficulty, demonstrating its ability to tackle diverse machine learning challenges.

ML-Master outperforms all baselines on every evaluation dimension defined by MLE-Bench. To better show the ability of ML-Master, we replace Bronze and Silver using Bronze+, Silver+ respectively. The plus notation indicates an equal or better result to the threshold. For example, Silver+ represents reaching a silver medal or better and it is equal to adding Silver and Gold together. The results are shown in Figure 6. We see that ML-Master outperforms other methods no matter which evaluation metric we choose as the minimum standard.

ML-Master is good at handling more complex machine learning tasks. We further split the complexity of machine learning tasks according to MLE-Bench and calculate the percentage of reaching any medals on different task complexities. The results are shown in Table 3. We see that ML-Master achieves 20.2% medal rate on medium complexity tasks and 24.4% medal rate on high complexity tasks, which benefits from the continuous reasoning and exploring process of ML-Master.

Table 3: Percentage of achieving any medals across different machine learning task complexities. ML-Master outperforms all baselines at each complexity level. Results for ML-Master are averaged over 3 runs with different random seeds and reported as mean ± one standard error of the mean (SEM). The top-performing model is shown in **bold**. * indicates that only a single run is conducted due to time and resource constraints.

Agent	Low(%)	Medium(%)	High(%)	Average(%)
MLAB [19]				
gpt-4o-2024-08-06	4.2 ± 1.5	0.0 ± 0.0	0.0 ± 0.0	1.3 ± 0.5
OpenHands [1]				
gpt-4o-2024-08-06	11.5 ± 3.4	2.2 ± 1.3	1.9 ± 1.9	5.1 ± 1.3
AIDE [2]				
gpt-4o-2024-08-06	19.0 ± 1.3	3.2 ± 0.5	5.6 ± 1.0	8.6 ± 0.5
o1-preview	34.3 ± 2.4	8.8 ± 1.1	10.0 ± 1.9	16.9 ± 1.1
Deepseek-R1*	27.3 ± 0.0	7.9 ± 0.0	13.3 ± 0.0	14.7 ± 0.0
R&D-Agent [3]				
o1-preview	48.2 ± 1.1	8.9 ± 1.0	18.7 ± 1.3	22.4 ± 0.5
ML-Master				
Deepseek-R1	**48.5 ± 1.5**	**20.2 ± 2.3**	**24.4 ± 2.2**	**29.3 ± 0.8**

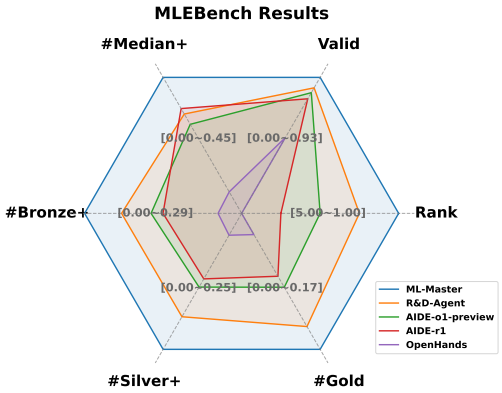

Figure 6: Performance of OpenHands, AIDE-r1, AIDE-o1-preview, R&D-Agent and ML-Master. The plus notation indicates an equal or better result to the threshold. ML-Master performs better on all dimensions.

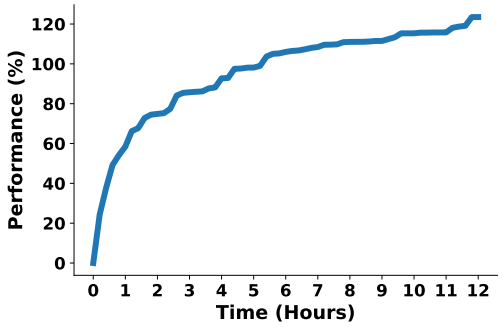

Figure 7: ML-Master's performance improves over time. Here, performance refers to the percentage improvement of the best version of the solution up to a certain point in time, compared to its initial version.

4.3 Analysis

ML-Master continues improving its solution over time. In Figure 7, we show how the answer solution given by ML-Master evolves over iteration time. The vertical axis represents the percentage improvement of the best version of the answer provided by ML-Master up to a certain point in time, compared to its initial version. The horizontal axis represents the iteration time. We can observe that as the iteration time increases, ML-Master continues to produce better solutions, demonstrating the effectiveness of its self-exploration and iterative reasoning process.

Ablations. We are actively conducting further ablation experiments on ML-Master and will report them in updated versions of this report.

5 Conclusions

In this paper, we introduced ML-Master, a novel AI4AI agent designed to seamlessly integrate exploration and reasoning into a unified iterative methodology. By employing an adaptively scoped memory mechanism, ML-Master effectively combines parallel solution exploration with reasoning, resulting in significant advancements in AI4AI.

Our extensive evaluation on the MLE-Bench demonstrates the effectiveness of ML-Master, surpassing existing AI4AI methods. Notably, ML-Master achieved an average medal rate of 29.3%, outperforming the previous state-of-the-art methods. Particularly in medium-difficulty tasks, it attains an impressive medal rate more than doubling the previous best result. Importantly, ML-Master achieved these remarkable results within just 12 hours, which is half the time typically allocated in previous studies. These results not only highlight the potential of ML-Master in solving complex AI development problems but also emphasize its capability to accelerate the path toward AGI.

The integration of exploration and reasoning within ML-Master offers a robust framework for AI systems that can autonomously evolve, learn, and adapt to increasingly complex challenges. As such, this work represents an important step in advancing AI4AI technologies, which are critical for the realization of AGI. In future work, we aim to further refine the scalability and adaptability of ML-Master, particularly in dynamic and multi-agent environments, to continue pushing the boundaries of AI agent autonomy and generalization.

References

[1] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents, 2025.

[2] Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code, 2025.

[3] Xu Yang, Xiao Yang, Shikai Fang, Bowen Xian, Yuante Li, Jian Wang, Minrui Xu, Haoran Pan, Xinpeng Hong, Weiqing Liu, et al. R&d-agent: Automating data-driven ai solution building through llm-powered automated research, development, and evolution. arXiv preprint arXiv:2505.14738, 2025.

[4] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. arXiv preprint arXiv:2410.07095, 2024.

[5] Marcin Szczepanski. Economic impacts of artificial intelligence (ai), 2019.

[6] Justus Wolff, Josch Pauling, Andreas Keck, and Jan Baumbach. The economic impact of artificial intelligence in health care: systematic review. Journal of medical Internet research, 22(2):e16866, 2020.

[7] Kunal Handa, Alex Tamkin, Miles McCain, Saffron Huang, Esin Durmus, Sarah Heck, Jared Mueller, Jerry Hong, Stuart Ritchie, Tim Belonax, et al. Which economic tasks are performed with ai? evidence from millions of claude conversations. arXiv preprint arXiv:2503.04761, 2025.

[8] Ammar Abulibdeh, Esmat Zaidan, and Rawan Abulibdeh. Navigating the confluence of artificial intelligence and education for sustainable development in the era of industry 4.0: Challenges, opportunities, and ethical dimensions. Journal of Cleaner Production, 437:140527, 2024.

[9] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In 5th International Conference on Learning Representations, 2017.

[10] Esteban Real, Chen Liang, David R So, and Quoc V Le. Automl-zero: Evolving machine learning algorithms from scratch. arXiv preprint arXiv:2003.03384, 2020.

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

[12] OpenAI. Introducing openai o1. `https://openai.com/o1/`, 2024.

[13] Anthropic AI. System card: Claude opus 4 & claude sonnet 4, 2024.

[14] Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. Automl-gpt: Automatic machine learning with gpt. arXiv preprint arXiv:2305.02499, 2023.

[15] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments, 2025.

[16] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants, 2025.

[17] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024.

[18] Yizhou Chi, Yizhang Lin, Sirui Hong, Duyi Pan, Yaying Fei, Guanghao Mei, Bangbang Liu, Tianqi Pang, Jacky Kwok, Ceyao Zhang, et al. Sela: Tree-search enhanced llm agents for automated machine learning. arXiv preprint arXiv:2410.17238, 2024.

[19] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. arXiv preprint arXiv:2310.03302, 2023.

[20] Jiakang Yuan, Xiangchao Yan, Shiyang Feng, Bo Zhang, Tao Chen, Botian Shi, Wanli Ouyang, Yu Qiao, Lei Bai, and Bowen Zhou. Dolphin: Moving towards closed-loop auto-research through thinking, practice, and feedback, 2025.

[21] Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2code: Automating code generation from scientific papers in machine learning, 2025.

[22] Shunyu Yao. The second half. Blog post, 2025. Available at: https://ysymyth.github.io/The-Second-Half/.

[23] Yuante Li, Xu Yang, Xiao Yang, Minrui Xu, Xisen Wang, Weiqing Liu, and Jiang Bian. R&d-agent-quant: A multi-agent framework for data-centric factors and model joint optimization, 2025.

[24] Zhiqiang Tang, Haoyang Fang, Su Zhou, Taojiannan Yang, Zihan Zhong, Tony Hu, Katrin Kirchhoff, and George Karypis. Autogluon-multimodal (automm): Supercharging multimodal automl with foundation models. arXiv preprint arXiv:2404.16233, 2024.

[25] Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In Workshop on automatic machine learning, pages 66–74. PMLR, 2016.

[26] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. arxiv 2020. arXiv preprint arXiv:2007.04074, 2022.

[27] Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Ml-plan: Automated machine learning via hierarchical planning. Machine Learning, 107:1495–1515, 2018.

[28] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. arXiv preprint arXiv:2003.06505, 2020.

[29] Sijia Liu, Parikshit Ram, Deepak Vijaykeerthy, Djallel Bouneffouf, Gregory Bramble, Horst Samulowitz, Dakuo Wang, Andrew Conn, and Alexander Gray. An admm based framework for automl pipeline configuration. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 4892–4899, 2020.

[30] Herilalaina Rakotoarison, Marc Schoenauer, and Michèle Sebag. Automated machine learning with monte-carlo tree search. arXiv preprint arXiv:1906.00170, 2019.

[31] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

[32] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8697–8710, 2018.

[33] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10734–10742, 2019.

[34] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. Advances in Neural Information Processing Systems, 36:46595–46623, 2023.

[35] 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), 2023.

[36] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. arXiv preprint arXiv:2409.12186, 2024.

[37] Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. Mlcopilot: Unleashing the power of large language models in solving machine learning tasks. arXiv preprint arXiv:2304.14979, 2023.

[38] Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, et al. Autokaggle: A multi-agent framework for autonomous data science competitions. arXiv preprint arXiv:2410.20424, 2024.

[39] Antoine Grosnit, Alexandre Maraval, James Doran, Giuseppe Paolo, Albert Thomas, Refinath Shahul Hameed Nabeezath Beevi, Jonas Gonzalez, Khyati Khandelwal, Ignacio Iacobacci, Abdelhakim Benechehab, Hamza Cherkaoui, Youssef Attia El-Hili, Kun Shao, Jianye Hao, Jun Yao, Balazs Kegl, Haitham Bou-Ammar, and Jun Wang. Large language models orchestrating structured reasoning achieve kaggle grandmaster level, 2024.

[40] Haoyang Fang, Boran Han, Nick Erickson, Xiyuan Zhang, Su Zhou, Anirudh Dagar, Jiani Zhang, Ali Caner Turkmen, Cuixiong Hu, Huzefa Rangwala, Ying Nian Wu, Bernie Wang, and George Karypis. Mlzero: A multi-agent system for end-to-end machine learning automation, 2025.

[41] NovelSeek Team, Bo Zhang, Shiyang Feng, Xiangchao Yan, Jiakang Yuan, Zhiyin Yu, Xiaohan He, Songtao Huang, Shaowei Hou, Zheng Nie, Zhilong Wang, Jinyao Liu, Runmin Ma, Tianshuo Peng, Peng Ye, Dongzhan Zhou, Shufei Zhang, Xiaosong Wang, Yilan Zhang, Meng Li, Zhongying Tu, Xiangyu Yue, Wangli Ouyang, Bowen Zhou, and Lei Bai. Novelseek: When agent becomes the scientist – building closed-loop system from hypothesis to verification, 2025.

[42] Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. Ai-researcher: Autonomous scientific innovation, 2025.

[43] Zujie Liang, Feng Wei, Wujiang Xu, Lin Chen, Yuxi Qian, and Xinhui Wu. I-mcts: Enhancing agentic automl via introspective monte carlo tree search, 2025.

[44] Kou Misaki, Yuichi Inoue, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling llm inference-time compute with adaptive branching tree search, 2025.

[45] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.