Systematic Idea Refinement for Machine Learning Research Agents

Zijun Liu, Cheng Gao, Hanxi Zhu
Department of Computer Science & Technology
Student ID: 2024310673, 2024210899, 2024310666

Abstract

The rapid evolution of large language models (LLMs) has catalyzed new methodologies for automating machine learning (ML) research, particularly in the areas of code generation and model design. While LLM-empowered agents have demonstrated notable success in generating syntactically correct solutions, they often fall short in addressing the complexity of real-world ML tasks, resulting in solutions that may be technically sound but suboptimal in terms of performance and adaptability. To address this gap, this project introduces a multi-agent framework named **Baby-AIGS-MLer** designed to enhance ML research agents in a pluggable way under existing Baby-AIGS framework. The proposed system integrates multiple agents, each with a specialized role, that work together to iteratively propose, refine, and evaluate diverse research ideas. Baby-AIGS-MLer leverages multi-level feedback from different stages of the research process, including LLM-based judgments, experimental results, and code generation, to expand the solution space and improve performance, which allows ideas to evolve based on both theoretical insights and empirical evidence. The effectiveness of Baby-AIGS-MLer is evaluated using MLE-Bench, a comprehensive benchmark suite consisting of *75 Kaggle competitions* across various ML tasks. Our preliminary experimental results show that the proposed multi-agent system significantly improves both the performance and efficiency of ML research agents, enabling them to generate and refine more optimal solutions. By incorporating systematic idea exploration and refinement, this approach paves the way for more advanced, self-improving agents that can contribute to the evolving landscape of machine learning research.

1 Introduction

The field of machine learning (ML) has witnessed transformative advancements with the advent of large language models (LLMs), which have enabled automation in various stages of the research process [1, 2], including research ideation, model generation, data preprocessing, and evaluation. LLM-empowered agents, utilizing the generative capabilities of these models, have been applied to a range of tasks such as code generation for ML models [3], designing and executing data analysis pipelines [4], and automating the process of hyperparameter tuning. By leveraging the immense knowledge embedded and reasoning capabilities in LLMs, these

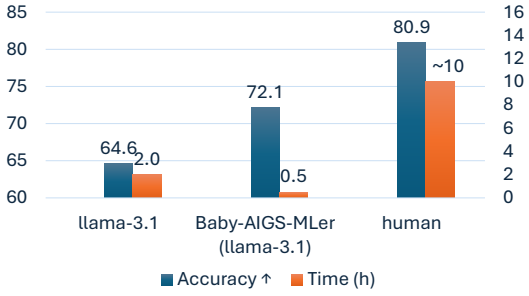

Figure 1: The preliminary experimental results on *Titanic-Spaceship* Kaggle contest[1] from MLE-Bench. Baby-AIGS-MLer system is built on top of LLaMA-3.1 model without agentic scaffolds.

[1] `https://www.kaggle.com/c/spaceship-titanic`.

Preprint. Under review.

agents have the potential to accelerate research, democratize access to sophisticated ML techniques, and allow even non-experts to engage in high-level research tasks.

However, despite these advancements, current applications of LLM-based agents in ML research remain constrained in several key aspects. One of the main limitations is that most existing systems primarily focus on focus on generating code snippets that solve specific tasks based on a narrow set of predetermined or self-determined methodologies [5]. These agents typically use a single, predetermined or self-determined approach to solve a given problem. For example, agents may be determined to solve a classification problem by generating code for a logistic regression model, without noticing it is not an appropriate choice for some dense temporal data they are facing. In practice, agents may consistently default to commonly used techniques that are widely applicable but may not be the most efficient or effective for the given task, resulting high validity rates but rather low performance, as shown by Chan et al. [6]. Moreover, while techniques such as tree search [7] and iterative refinement [8] have been incorporated into some LLM-based systems to ensure syntactic correctness, these approaches focus primarily on improving the technical validity of the generated code. These agents, instead, lack the ability to explore multiple solution strategies and adapt dynamically to evolving challenges, limiting their performance in complex, real-world tasks [9], particularly in novel or poorly defined research scenarios where the optimal solution may not be apparent from the outset.

To overcome these limitations, this project proposes a multi-agent framework that systematically explores and refines research ideas through multi-level feedback integration and is compatible with inference compute scaling techniques. The framework is named **Baby-AIGS-MLer**, which is inherited from Baby-AIGS system [2] but aimed at constructing optimal solutions for ML research problems, rather than falsifying scientific discoveries. Baby-AIGS-MLer expands the solution space by enabling agents to autonomously generate and refine diverse ideas and corresponding methodologies to solve a given ML problem. As shown in Figure 2, the process follows an iterative manner. This mirrors the exploratory and iterative approach employed by human researchers, where multiple research ideas are implemented, tested, refined, and adapted based on empirical results and theoretical insights. At the heart of this multi-agent framework are four specialized agents: PROPOSAL AGENT, which generates initial ideas for possible solutions;CODE GENERATION AGENT, which translates these ideas into executable code; EXPERIMENT CONDUCTOR, which runs experiments and evaluates the performance of the generated solutions; and REVIEW AGENT, which assesses the empirical results based on previous solutions and provides feedback to guide further refinement. Baby-AIGS-MLer could be performed with inference compute scaling techniques [10]. For instance, the initial ideation process could be sampled multiple times, resulting in a batch of various ideas, enhancing the diversity of methodologies.

The proposed system is designed to be plug-and-play, meaning that it can be integrated with existing code generation agents to enhance their ability to explore and evaluate diverse solutions. This is especially crucial in the context of modern ML research, where code generation agents require dedicated efforts to design and implement. To evaluate the efficacy of the proposed framework, we conduct experiments using MLE-Bench, a benchmark suite that includes 75 Kaggle competitions across a wide range of ML tasks. This comprehensive testing ground allows us to assess the multi-agent system's ability to generate high-quality solutions in a variety of contexts, demonstrating its potential to improve the performance and versatility of ML research agents. The preliminary experiment selects two from the 75 contests, and the experimental results show that the idea refinement process is effective on different agentic scaffolds, and that ML research agent systems have the potential to catch up experienced human experts (Figure 1).

By enabling agents to systematically explore diverse methodologies, integrate multi-level feedback, and refine their approaches in real time, Baby-AIGS-MLer aims to push the boundaries of current ML research agents. Ultimately, this project seeks to contribute to the development of more advanced, adaptive, and efficient automated research systems that can more effectively tackle the challenges of front-end scientific problems.

2 Related Work

Machine Learning System Development Agents LLM-based agents have become integral to automating ML researches [11]. These agents leverage the generative capabilities of LLMs, using

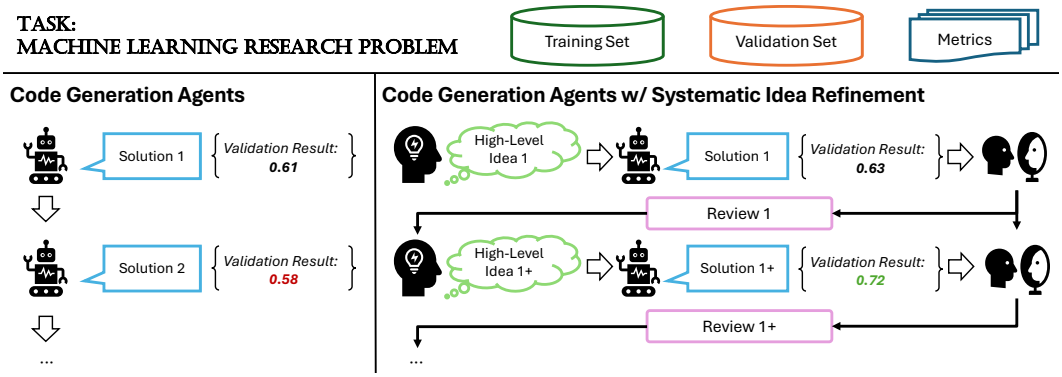

Figure 2: The overview of our proposed multi-agent idea refinement framework **Baby-AIGS-MLer** for ML research. Traditionally, the code generation agents could produce multiple solutions through sampling, but the solutions are mostly independent and follow well-used, sub-optimal methodologies (left). Instead, idea refinement allows agents to iteratively evolve high-level ideas and optimize the methodology based on actual experimental results (right).

methods like multi-agent workflow [12], tree search [7], or iterative refinement [8] to improve code validity and correctness. These systems can produce syntactically correct and functionally valid solutions. However, recent works [6] have demonstrated that such systems can be effective in producing valid solutions across a range of problems while failing to yield higher performance. This underscores the need for frameworks that can propose and refine ideas to guide code generation for general ML tasks. Still, all systems above may face performance degradation when facing poorly defined tasks, e.g., without clear metrics, which we decide to leave for future works.

Research Idea Generation The concept of research idea generation extends beyond coding, aiming to replicate the behavior of human researchers. However, most existing approaches [3, 13] focus on refining an initial idea with inherent knowledge in LLMs or existing literature, rather than taking richer feedback from experimental results. In contrast, human researchers often select the most promising approach based on both theoretical and empirical results. Emulating this idea-driven exploratory process has the potential to enhance the effectiveness. The lessons of previous work [14] also show that integration with the implementation of research ideas in the code is challenging.

AI-Powered Automated Research Systems The automation of ML research through LLM-empowered agents has been the focus of recent studies. Recent work has explored techniques such as multi-agent workflows [3, 1, 2], and reinforcement learning from scholar feedback [15] to ensure that the generated solutions are not only valid but also robust. However, these systems often remain constrained by the capacity of foundation models in multiple aspects, including literature retrieval, ideation, methodology implementation, experimentation, and falsification. This field is much under-explored. It is worth noting that Baby-AIGS-MLer is derived from Baby-AIGS system with specialized ML development capacity and without an explicit falsification process.

3 Task & Framework Formulation

3.1 Machine Learning Research Problems

The core problem addressed in this project is to conduct ML research. Formally, let $\mathcal{D}_{\text{test}} = \{(x_i^{(\text{test})}, y_i^{(\text{test})})\}_{i=1}^{N_{\text{test}}}$ denote the test set of a given ML problem P, where $x_i^{(\text{test})}$ represents input features, $y_i^{(\text{test})}$ represents the corresponding labels for each data point; $\mathcal{D}_{\text{train}} = \{(x_i^{(\text{train})}, y_i^{(\text{train})})\}_{i=1}^{N_{\text{train}}}, \mathcal{D}_{\text{val}} = \{(x_i^{(\text{val})}, y_i^{(\text{val})})\}_{i=1}^{N_{\text{val}}}$ represents the training set and the validation set, respectively. Without loss of generality, the goal could be described as to identify a model $f : \mathcal{X} \to \mathcal{Y}$ that maximizes a performance metric function $\mathcal{L}(f(x), y)$ on the test set $\mathcal{D}_{\text{test}}$, since those descending metrics could be reversed into ascending ones, and multiple metrics could be averaged as one. $\mathcal{L}$ could be a loss function or performance measure relevant to the task. The formal expression

is:

$$f_{\text{goal}} = \text{argmax}_{f \sim \mathcal{S}(P)} (\sum_{i=1}^{N_{\text{test}}} \mathcal{L}(f(x_i^{(\text{test})}), y_i^{(\text{test})})), \quad (x_i^{(\text{test})}, y_i^{(\text{test})}) \in D_{\text{test}} \tag{1}$$

where f_{goal} is the optimal solution and $\mathcal{S}$ represents the agentic system (Equation (2) or (3)).

For a given ML problem P, LLM-based agents are required to design and implement a model $f \in \mathcal{F}$ as a potential solution, where $\mathcal{F}$ represents the space of potential models. Thus, a key challenge in ML research is not just solving a problem but determining which methodology or model f is best suited to the problem at hand, i.e., most approximate to f_{goal}. The space of potential models $\mathcal{F}$ is vast, and identifying the optimal model requires an indicator to identify which f is better. It should be noted that agents could use the validation set $\mathcal{D}_{\text{val}}$ for the development and validation of methods and could access the metric $\mathcal{L}$. Thus, $\mathcal{L}(f(x), y), (x, y) \in \mathcal{D}_{\text{val}}$ could be used as an indicator. If multiple candidates $\mathcal{F}' \subset \mathcal{F}$ are generated, the best outcome $\tilde{f} \in \mathcal{F}'$ could be selected from the candidates as the approximate to f_{goal}.

3.2 Single-Agent & Multi-Agent Systems

In a traditional single-agent system, a single LLM-powered agent is tasked with generating an appropriate solution for a given ML problem. The agent receives an input prompt π that describes the problem P and is responsible for producing a model $f \in \mathcal{F}$. The model is then evaluated using a metric function $\mathcal{L}(f(x), y)$, which assesses the performance of the model. Formally, we can represent a single-agent workflow as:

$$f \sim \mathcal{S}(P) = A(\pi, D_{\text{train}}, D_{\text{val}}, \mathcal{L}), \tag{2}$$

where A is an LLM-empowered agent that generates a model $f \in \mathcal{F}$ by coding based on an initial prompt $\pi \in \Pi$ and its predefined workflow, and Π represents all possible input prompts. The generated model f has the objective to maximize the metric function $\mathcal{L}$.

The multi-agent system leverages the collaboration of several agents, and the collaboration is realized by establishing communications for information exchanges between agents. A multi-agent system

$$f \sim \mathcal{S}(P) = \mathcal{M}(\pi, D_{\text{train}}, D_{\text{val}}, \mathcal{L}) \tag{3}$$

forms communications between a set of agents $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$, where each agent A_i is determined by its input prompt π_i, and its workflow. Agents collectively propose and refine various methodologies for solving P by exchanging messages between each other in a predefined order. A single agent A is the special case of $\mathcal{M}$. In this project, we focus on designing pluggable methods for code generation agents, with the constraints that only one agent generates the final code and thus produces the ML model $f \in \mathcal{F}$, and the others mainly provide guidelines or feedback.

3.3 Inference Compute Scaling

In the context of this project, inference compute scaling [10] techniques can be mainly divided into two main components: iterative refinement and parallel sampling, both of which contribute to improving the exploration and optimization of model solutions. We formally define these concepts below.

Iterative Refinement Iterative refinement refers to the repeated application of a refinement process in which an initial solution is progressively improved by incorporating feedback over time. The iterative process increases the computational effort at each iteration and aims to make the system converge toward an optimal solution through incremental improvements. Mathematically, the iterative refinement process can then be described as:

$$f_{t+1} \sim \mathcal{R}(f_t, \mathcal{S}, P), \quad t = 0, ..., \mathcal{T} - 1, f_0 = \phi, \tag{4}$$

where f_t is the solution at iteration t, f_{t+1} is the updated solution after applying the refinement function $\mathcal{R}$, and $\mathcal{T}$ denotes the maximum iteration time. The compute resources required for the iterative refinement scale with the number of iterations $\mathcal{T}$ and the complexity of the refinement process. As the result, the generated solution candidates are $\mathcal{F}' = \{f_t\}_{t=1}^{\mathcal{T}}$.

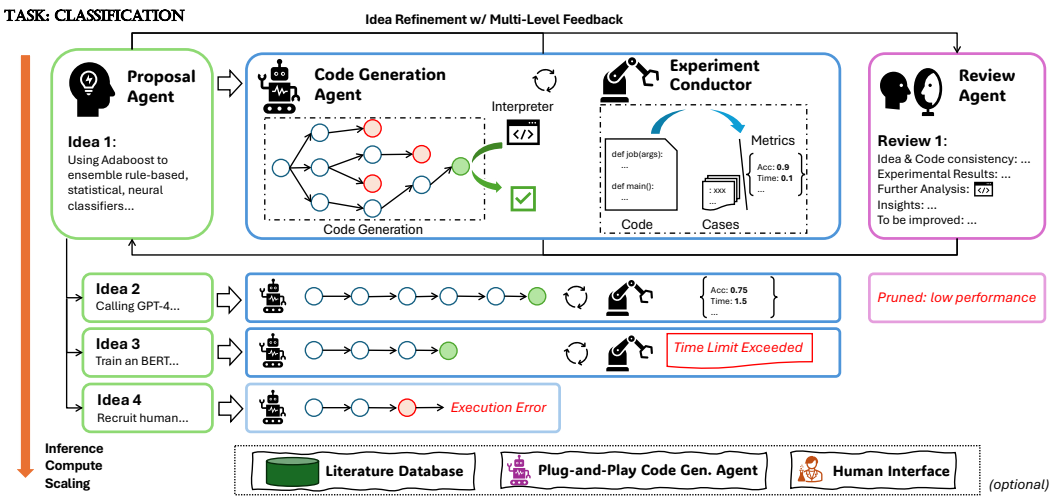

Figure 3: The demonstration of Baby-AIGS-MLer framework with inference compute scaling. The framework comprises PROPOSAL AGENT, CODE GENERATION AGENT, EXPERIMENT CONDUCTOR, and REVIEW AGENT. The inference compute scaling technique here refers to the parallel sampling (Section 3.3). In each parallel thread, the system performs iterative idea refinement.

Parallel Sampling Parallel sampling refers to the simultaneous exploration of multiple candidate solutions across different parts of the solution space, each with different initial conditions or exploration paths. This approach significantly reduces the time required to explore a diverse set of solutions, as multiple samples are generated and evaluated concurrently. The agentic system $\mathcal{S}$ explores different solutions $f_p \sim \mathcal{S}_p(P), p = 1, ..., N_p$ in parallel, where S_p is the p-th instance of S. So that the final solution candidates $\mathcal{F}' = \{f_p\}_{p=1}^{N_p}$. In each thread, the system instance may perform independently, leading to a wider exploration of the solution space. The compute cost depends on the number of parallel threads P and the computational complexity of the system.

4 Proposed Method: Baby-AIGS-MLer

As shown in Figure 3, the proposed methodology introduces a multi-agent system named **Baby-AIGS-MLer** designed to systematically explore, refine, and optimize ML research ideas. This system aims to extend beyond the capabilities of current code generation agents by facilitating the continuous refinement of proposed methodologies through feedback integration and inference-time scaling. Below, we detail the components and functions of the multi-agent framework, followed by a discussion on how the framework can be applied to a wide variety of existing code generation agents.

4.1 The Multi-Agent Idea Refinement Framework

The multi-agent framework proposed in this work is structured around four specialized agents, each tasked with a distinct role in the idea generation and refinement process. These agents work collaboratively, exchanging multi-level feedback to ensure that each idea is sufficiently explored and refined. The idea refinement process mimics the exploratory nature of human research, in which multiple hypotheses are tested, reviewed, and refined based on both theoretical and empirical insights.

The framework consists of the following key agents:

- **PROPOSAL AGENT:** The first agent in the framework is responsible for proposing potential research ideas and methodologies. It generates hypotheses based on the problem description, the **multi-level feedback** from previous iterations, and prior knowledge embedded in the LLM. The multi-level feedback includes the proposed idea from PROPOSAL AGENT, experimental results in metric values and recorded data points from EXPERIMENT CONDUCTOR, and the review from REVIEW AGENT in the last iteration if available.

- **CODE GENERATION AGENT:** Once PROPOSAL AGENT generates an idea, CODE GENERATION AGENT is responsible for translating that idea and the corresponding method into executable code for experimentation. This agent takes the proposed methodology and implements it using appropriate machine learning frameworks, libraries, and algorithms.
- **EXPERIMENT CONDUCTOR:** EXPERIMENT CONDUCTOR is responsible for managing the experimental workflow. After CODE GENERATION AGENT generates the code implementation of a model f, EXPERIMENT CONDUCTOR trains the model on the training set $\mathcal{D}_{\text{train}}$, evaluates the model's performance on the validation set $\mathcal{D}_{\text{val}}$, and collects metric values and all inference outcomes. These results are critical for REVIEW AGENT to understand the effectiveness of the proposed model and guide further refinement.
- **REVIEW AGENT:** Finally, REVIEW AGENT evaluates the code implementation and the performance, and tries to derive constructive insights. The review process involves analyzing the code implementation from CODE GENERATION AGENT and performance metrics obtained by EXPERIMENT CONDUCTOR, and offering suggestions for improvement. REVIEW AGENT may write code to analyze the experimental results in depth. Here are the components of a review, which are generated sequentially:

  Components of a Review

  - Idea & Code Consistency
  - Experimental Results
  - In-Depth Analysis (*conducted with code snippets*)
  - Insights
  - Actionable Improvement

  The feedback provided by REVIEW AGENT helps close the loop in the idea refinement process. By incorporating this feedback, the system iteratively improves the model, evolving the approach towards an optimal solution.

The interaction between these agents is sequential, except for CODE GENERATION AGENT which may test its codes with EXPERIMENT CONDUCTOR. This multi-level feedback ensures that the solution space is continuously explored, empowering the system with the potential to produce better models over time.

According to Section 3.3, Baby-AIGS-MLer combines two inference compute scaling methods. The system performs parallel sampling by sampling a diverse set of high-level ideas. And all system instance in different threads as a whole perform iterative idea refinement. Specifically, when the t-th iteration is finished, the best idea and method ($\tilde{f}$) will be kept for the $(t+1)$-th iteration; and at the beginning of the $(t+1)$-th iteration, N_p new ideas will be sampled according to the feedback from $\tilde{f}$. As a demonstration, idea 2 is filtered out because of the lower performance than idea 1 in Figure 3.

4.2 Applying the Framework to Arbitrary Code Generation Agents

One of the key advantages of the proposed multi-agent framework is its flexibility and plug-and-play nature. The framework is designed to work with arbitrary existing code generation agents [8, 7], including bare LLMs or complicated agents. Most of advanced code generation agents require to access the experiment template and environment. Thus, we expose the interface of EXPERIMENT CONDUCTOR for these agents, which is usually provided by the ML task benchmark, e.g., MLE-Bench. By integrating the multi-agent framework with various code generation agents, we achieve a system that is not only flexible but also capable of continuously adapting to the state-of-the-art agent.

5 Preliminary Experiments

In this section, we present the results of preliminary experiments designed to evaluate the effectiveness of Baby-AIGS-MLer. We compare its performance against three baseline methods: bare models, code generation agents, and human developers. The experiments were conducted on two Kaggle competition datasets, selected to cover both simple and complex machine learning tasks. The

experimental results highlight the effectiveness of Baby-AIGS-MLer in generating high-performance solutions to real-world, complex ML research problems through systematic idea exploration and iterative refinement. Although not surpassing experienced human experts, the effectiveness and efficiency of the framework underscore its potential to enhance the ML research process.

5.1 Experiment Setup

The selected Kaggle tasks are as follows:

- ***Titanic - Spaceship***[1]**:** This dataset is a binary classification problem where the task is to predict whether a passenger teleported or not based on attributes such as age, class, sex, and other information.
- ***LMSYS - Chatbot Arena Human Preference Predictions***[2]**:** This dataset involves predicting human preferences for different LLM responses. The challenge lies in handling hidden features that influence human perception, requiring more sophisticated model exploration.

Both datasets present distinct challenges and allow us to assess the ML research systems to handle different types of machine learning problems.

For each dataset, we compared Baby-AIGS-MLer with three baseline methods with the same LLaMA-3.1-70B-Instruct[3] backbone model:

- **Bare Models**: Standard models trained without any agentic scaffolds, serving as a baseline for typical single-model solutions.
- **AIDE** [7]: An agent that generates solutions based on a search-based workflow, used in the default setting as implemented in MLE-Bench.
- **Human Experts**: Experienced human experts who manually design and implement machine learning solutions based on their expertise and knowledge of the problem domain.[4]

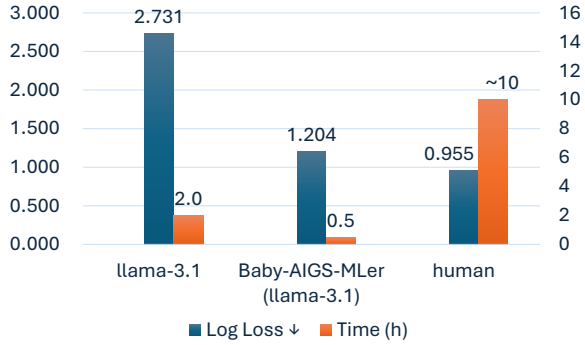

Figure 4: The preliminary experimental results on *LMSYS-Chatbot Arena Human Preference Predictions* Kaggle contest[2] from MLE-Bench. Baby-AIGS-MLer system is built on top of bare LLaMA-3.1 model without agentic scaffolds.

Each ML task is evaluated using metrics as in MLE-Bench [6]: accuracy for the Titanic dataset, and log loss for the LMSYS dataset. We limit the inference time for each method. If a method is stopped, we will concatenate the previous code implementation into the prompting template and start the method again. For Baby-AIGS-MLer, $N_p = 3$, and $\mathcal{T}$ is set to an enormous value to avoid stopping. For human expert solutions, three experienced researchers with top conference publications were recruited to estimate the average time for development, and eventually agreed on about 10 hours.

5.2 Bare Models vs. Baby-AIGS-MLer

The results of bare models and Baby-AIGS-MLer in the *Titanic-Spaceship* and the *LMSYS-Chatbot Arena Human Preference Predictions* tasks are shown in Figure 1 and Figure 4 respectively. In the *Titanic-Spaceship* task, Baby-AIGS-MLer outperforms the bare model by 11.6%, and 126.8% in the *LMSYS-Chatbot Arena Human Preference Predictions* task. The experimental results demonstrate the

[1] `https://www.kaggle.com/c/spaceship-titanic`.

[2] `https://www.kaggle.com/competitions/lmsys-chatbot-arena`.

[3] `https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct`.

[4] For *Titanic-Spaceship*, the human expert solution is from `https://www.kaggle.com/code/samuelcortinhas/spaceship-titanic-a-complete-guide`. For *LMSYS-Chatbot Arena Human Preference Predictions*, the human expert solution is from `https://www.kaggle.com/code/natu2003/fine-tuning-gemma2`.

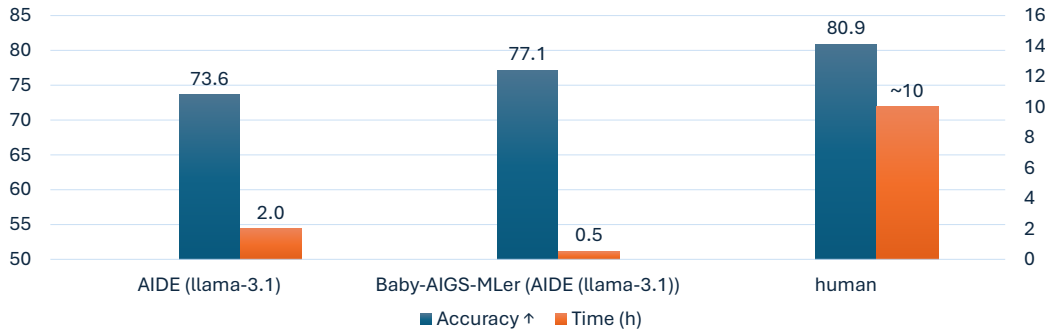

Figure 5: The preliminary experimental results on *Titanic-Spaceship* contest. Baby-AIGS-MLer system is built on top of LLaMA-3.1 model with AIDE.

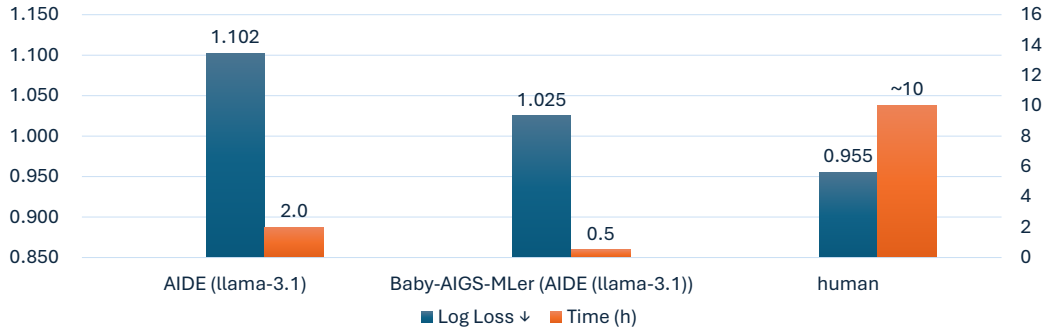

Figure 6: The preliminary experimental results on *LMSYS-Chatbot Arena Human Preference Predictions* contest. Baby-AIGS-MLer system is built on top of bare LLaMA-3.1 model with AIDE.

advantage of systematically refining ideas on single-model solutions. In addition, scaling inference compute on solutions with bare models with techniques like iterative refinement seems to result in limited positive effects.

5.3 Code Generation Agents vs. Baby-AIGS-MLer

The results of AIDE and Baby-AIGS-MLer in the *Titanic-Spaceship* and the *LMSYS-Chatbot Arena Human Preference Predictions* tasks are shown in Figure 5 and Figure 6 respectively. On *Titanic-Spaceship* task, Baby-AIGS-MLer outperforms the code generation agents by 4. 8%, and in the *LMSYS-Chatbot Arena Human Preference Predictions* task Baby-AIGS-MLer reduces the log loss by 7.5% . This improvement demonstrates the multi-agent system's ability to explore a broader set of methodologies and to steadily refine them towards more optimal ones. However, the code generation agent significantly outperforms single-model solutions from Figure 1 and Figure 4, leaving smaller leading space for Baby-AIGS-MLer.

5.4 Human Developers vs. Baby-AIGS-MLer

As an upperbound, we compared Baby-AIGS-MLer with expert human developers, who manually designed models for the Titanic and LMSYS datasets. In both datasets, human experts significantly outperform all other methods, including Baby-AIGS-MLer (Figure 1, Figure 4, Figure 5, and Figure 6), indicating that experienced human experts could achieve much better results as long as they have enough inference compute, or development time in this case. This finding is aligned with Wijk et al. [9] who showed that human researchers have better inference compute scaling capacity compared to the state-of-the-art foundation models.

6 Conclusion and Prospects

This paper presents **Baby-AIGS-MLer**, a multi-agent framework designed to systematically explore, refine, and evolve ideation for machine learning research problems. By integrating systematic idea refinement, the framework transcends the limitations of existing code generation agents, enabling the exploration of diverse and adaptive solutions. Preliminary experiments on Kaggle competitions that Baby-AIGS-MLer achieves significant performance gains over baseline agents and shows better inference compute scaling capacity, though still falling short of experienced human experts. These findings highlight the importance for ML research agents to identify and optimize promising ideas during research automation while emphasizing the persistent gap between agents and human ingenuity. Looking ahead, the prospects of Baby-AIGS-MLer lie in its potential scalability with larger inference compute and its adaptability with arbitrary code generation agents. Future research can address current limitations by testing with more compute budgets. Moreover, transforming existing creativity from human into LLM-driven ideation may narrow the gap with human experts.

7 Limitations

While Baby-AIGS-MLer demonstrates empirical improvements with systematic idea refinement for ML research, it is not without limitations. First, the framework inherits the main design of the Baby-AIGS system, limiting the novelty of the work. Second, the scale of experiments is small and currently confined to Kaggle competitions due to time and resource constraints, which may not fully represent the complexity and variability of real-world ML problems. Third, the lack of extensive ablation studies on the impact of individual agents and inference compute scaling techniques limits insights into the relative contributions of each component, which is also due to limited time. Finally, the use of a single LLM backbone (LLaMA-3.1-70B-Instruct) may constrain the generalizability of findings, as performance could vary significantly with alternative base models or larger foundation models. Addressing these limitations is essential for broadening the applicability of the framework and improving its robustness in diverse research contexts.

8 Ethics Statement

The development of Baby-AIGS-MLer could raise important ethical considerations, particularly in the context of research automation and AI-driven ideation. While the framework aims to enhance the efficiency of machine learning research, its reliance on automated systems to generate and refine ideas may inadvertently perpetuate biases inherent in the training data or algorithms, thus affecting the fairness and diversity of the solutions it proposes. Additionally, the potential for over-reliance on automated agents could reduce human oversight, leading to the marginalization of domain-specific expertise and creativity. It is crucial that future iterations of Baby-AIGS-MLer include mechanisms for transparency, accountability, and interpretability to ensure that the generated solutions are not only effective but also ethically sound, e.g., to include an explicit falsification process [2]. Furthermore, the framework's capacity for large-scale ideation must be balanced with responsible usage to prevent misuse in creating harmful or unethical machine learning applications. Ethical guidelines must be continuously revisited to align the framework's evolution with societal values and human welfare.

References

[1] Alireza Ghafarollahi and Markus J. Buehler. Sciagents: Automating scientific discovery through multi-agent intelligent graph reasoning. *Computing Research Repository*, arXiv:2409.05556, 2024. URL `https://arxiv.org/abs/2409.05556`.

[2] Zijun Liu, Kaiming Liu, Yiqi Zhu, Xuanyu Lei, Zonghan Yang, Zhenhe Zhang, Peng Li, and Yang Liu. Aigs: Generating science from ai-powered automated falsification. *Computing Research Repository*, arXiv:2411.11910, 2024. URL `https://arxiv.org/abs/2411.11910`.

[3] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. *Computing Research Repository*, arXiv:2408.06292, 2024. URL `https://arxiv.org/abs/2408.06292`.

[4] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge,

Taicheng Guo, Tuo Zhou, Wei Tao, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zhibin Gou, Zongze Xu, and Chenglin Wu. Data Interpreter: An LLM Agent For Data Science. *Computing Research Repository*, arXiv:2402.18679, 2024. URL https://arxiv.org/abs/2402.18679.

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

[6] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. *Computing Research Repository*, arXiv:2410.07095, 2024. URL https://arxiv.org/abs/2410.07095.

[7] Dominik Schmidt, Zhengyao Jiang, and Yuxiang Wu. Technical Report. https://www.weco.ai/blog/technical-report, 2024.

[8] 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. *Computing Research Repository*, arXiv:2407.16741, 2024. URL https://arxiv.org/abs/2407.16741.

[9] Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *Computing Research Repository*, arXiv:2411.15114, 2024. URL https://arxiv.org/abs/2411.15114.

[10] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *Computing Research Repository*, arXiv:2407.21787, 2024. URL https://arxiv.org/abs/2407.21787.

[11] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20271–20309. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

[12] Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. AutoML-Agent: A Multi-Agent LLM Framework for Full-Pipeline AutoML. *Computing Research Repository*, arXiv:2410.02958, 2024. URL https://arxiv.org/abs/2410.02958.

[13] Haoyang Su, Renqi Chen, Shixiang Tang, Xinzhe Zheng, Jingzhe Li, Zhenfei Yin, Wanli Ouyang, and Nanqing Dong. Two Heads Are Better Than One: A Multi-Agent System Has the Potential to Improve Scientific Idea Generation. *Computing Research Repository*, arXiv:2410.09403, 2024. URL https://arxiv.org/abs/2410.09403.

[14] Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers. *Computing Research Repository*, arXiv:2409.04109, 2024. URL https://arxiv.org/abs/2409.04109.

[15] Yixuan Weng, Minjun Zhu, Guangsheng Bao, Hongbo Zhang, Jindong Wang, Yue Zhang, and Linyi Yang. Cycleresearcher: Improving automated research via automated review. *Computing Research Repository*, arXiv:2411.00816, 2024. URL https://arxiv.org/abs/2411.00816.