Scaling Inference Time Compute for Machine Learning Engineering Agents

Asim Osman (asim@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)
University of Cape Town, South Africa

Supervised by: Dr. Arnu Pretorius, Arnol Fokam
University of Stellenbosch, South Africa & InstaDeep, South Africa

12 June 2025
Submitted in partial fulfillment of an AI for science masters degree at AIMS South Africa

Abstract

Problem and Motivation: The high cost and limited accessibility of proprietary large language models (LLMs) for machine learning engineering tasks has created a significant barrier for researchers and practitioners. While recent inference-time scaling techniques have shown promise in enhancing model capabilities without requiring larger parameters, their application to coding and ML engineering domains—particularly with open-source models—remains largely unexplored. Approach: This work presents the first systematic implementation of inference-time scaling (ITS) strategies within an open-source agentic framework for ML engineering tasks. We augment the AIDE (AI-Driven Exploration) agent scaffold with multiple ITS techniques including self-consistency, iterative self-debugging, self-reflection, and modular task decomposition, specifically targeting DeepSeek-R1 distilled models (7B, 14B, and 32B parameters). We evaluate these enhanced agents on a curated subset of MLE-Bench competitions spanning diverse ML domains. Key Findings: Our experiments reveal 10% improvement in valid submission rate, 10% increase in medal-winning performance. Notably, we observe that ITS effectiveness correlates strongly with model size—the 32B DeepSeek model shows substantial gains from our strategies, achieving performance comparable to GPT-4 Turbo, while smaller 7B models demonstrate limited improvement. Self-consistency and prompt chaining prove most effective, while self-reflection shows minimal gains, suggesting limitations in these models' self-correction capabilities. [PLACEHOLDER: Specific numerical comparisons with baselines] Impact and Novelty: This work delivers the first open-source, accessible framework for ML engineering automation that demonstrates competitive performance with proprietary alternatives. We provide four distinct AIDE variants, each optimized for different ITS strategies, establishing a foundation for future research in open-source coding agents. Our findings offer crucial insights into the scaling behavior of inference-time techniques in coding domains, demonstrating that strategic computational allocation during inference can bridge the capability gap between open-source and proprietary models.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Asim Osman, 12 June 2025

Contents

1. Introduction

Over the last few years, advancements in large language models have primarily relied on scaling up the number of parameters alongside increasing dataset size and compute budgets. This strategy, often referred to as train-time compute scaling, has been remarkably effective, significantly and consistently improving model performance on tasks such as translation, general question answering, and sophisticated reasoning in mathematics and coding. However, the exponential growth in model size and the associated costs have become prohibitively expensive, pushing training budgets to the billion-dollar scale and raising concerns about resource sustainability, particularly as high-quality training data becomes scarcer.

Consequently, significant interest has shifted toward inference-time (or test-time) scaling techniques. Rather than focusing on costly retraining and larger models, inference-time scaling enhances existing models by dedicating additional computational resources during prediction. These methods enable models to "think longer" on challenging problems by dynamically generating multiple reasoning paths, employing voting mechanisms, or iteratively refining their outputs. Recent studies, such as DeepMind's compute-optimal scaling (Doe, 2023), demonstrate how smaller models can achieve impressive performance through adaptive inference strategies, sometimes even outperforming substantially larger models.

In parallel with these developments, there has been a notable emergence of "reasoning" or "thinking" models, designed explicitly to enhance models' capabilities to reason through complex problems. A prime example is DeepSeek-R1, which explicitly trains models using reinforcement learning to produce long, structured chains of thought before providing an answer, enabling LLMs to systematically tackle tasks that were previously challenging even for large-scale models. DeepSeek released a series of small distilled models trained on reasoning datasets generated from the original R1 model, empowering them with remarkable reasoning capabilities and improving their performance relative to larger models.

Motivated by these developments, this work explores the promise of inference-time scaling techniques applied to small-scale language models-particularly those distilled from DeepSeek-that are specifically designed for coding and machine learning engineering tasks, areas still challenging even for frontier models. Smaller models have substantial advantages in accessibility, rapid deployment, and lower operational costs, but traditionally lack the advanced reasoning capabilities exhibited by larger models. Through systematic implementation and evaluation of strategies such as chain-of-thought prompting, iterative self-refinement, self-consistency, and other methods, this project aims to bridge the performance gap. Ultimately, this thesis evaluates the extent to which inference-time scaling can address complex tasks such as machine learning engineering. We demonstrate that, with carefully designed inference-time enhancements, small-scale models can significantly improve performance in complex coding and engineering contexts, presenting a viable and cost-effective alternative to training massive foundational models.

2. Background and Related Work

2.1 Background

2.1.1 Large Language Models (LLMs)

Large Language Models (LLMs) are neural networks trained on extensive text datasets, capable of generating human-like text. Based primarily on the transformer architecture, these models utilize a straightforward predictive objective, leveraging enormous volumes of data to tackle complex problems and diverse natural language tasks [CITATION], including translation [CITATION], semantic analysis [CITATION], and information retrieval.

Recent advancements in LLMs, exemplified by models such as the GPT series [CITATION*3] and the Llama family [CITATION], have been driven by significant scale, reaching billions of parameters. This substantial scale has unlocked emergent abilities—advanced skills like reasoning and in-context learning—that smaller models typically do not exhibit. These emergent capabilities enable LLMs to solve arithmetic problems, answer intricate questions, and summarize complex passages, often without explicit task-specific training.

2.1.2 Scaling Laws and Train-Time Compute

Scaling lows shows how the performance of Large Language models improves in a predeictable way with the increase in the parameter size, training data and computationa; resources, openai ahs shown that LLMs performance as measured by metrics such as accuracy or perplexity follows a power low relationship with these scaling factors. Performance enhancements are steady yet exhibit diminishing returns as scale increases, motivating the development of increasingly larger models like PaLM and Chinchilla.

While scaling laws provided performance guarantees and drove a competitive race for building larger models, leading to current models in the hundreds of billions of parameters, practical limitations have surfaced. High-quality data is becoming scarce relative to growing model demands, and the significant costs and resource consumption associated with these models pose substantial challenges. Moreover, LLMs auto-regressive and token level prediction style is itself a limitation when it comes to complex reasoning tasks such as advanced mathematics or complex question answering. While recent years witnessed substantial developments in the models abilities, especially using post training and fine tuning, this requires massive computational resources and large scale datasets, encouraging researchers to to find cheaper approaches.

2.1.3 Inference-Time Scaling (ITS)

To address limitations associated with traditional scaling strategies, a new paradigm has emerged: inference-time scaling, also known as test-time scaling. Instead of allocating more resources to pre-training, this approach dynamically adjusts computational resources during inference, enabling models to "think longer" and thus handle more complex problems. Inference-time scaling involves techniques such as generating longer reasoning chains, sampling multiple possible answers, and iteratively refining outputs. This strategy has shown significant promise, allowing smaller models to surpass larger counterparts.

Techniques in inference-time scaling include:

- **Chain-of-Thought Prompting (CoT):** Encouraging models to explicitly generate intermediate reasoning steps.
- **Iterative Self-Refinement:** Models iteratively identify and correct errors in their outputs.
- **Self-Consistency:** Generating multiple independent outputs and selecting the most consistent result via voting.
- **Verifier-Guided Search Techniques:** Utilizing reward-based verifiers or structured search methods to identify optimal answers from multiple generated solutions.

2.1.4 Emergence of Reasoning Models

Recent developments have highlighted the emergence of models focused on reasoninging specifically designed to enhanthe capabilities in multistep logical and mathematical reasoning. These reasoning models utilize inference-time strategies like structured prompting, external tool usage, and systematic search without altering model weights. Techniques such as ReAct, PAL, Toolformer, and Tree-of-Thoughts have demonstrated significant improvements in reasoning abilities.

DeepSeek-R1 serves as a notable landmark, explicitly trained via reinforcement learning on structured reasoning datasets to generate coherent reasoning chains, significantly improving performance on complex reasoning tasks. DeepSeek's release of smaller distilled models, trained on datasets generated by R1, demonstrates the potential of distillation combined with structured inference strategies, achieving performance comparable or superior to larger models.

* the effect of rl training on producing long chains of thoughts * the distilled models that were trained on the reasoning datata from R1 model

2.1.5 Agentic Systems and Scaffolding

"Scaffolding" reffers to code built up around an LLM in order to augment its capabilities. This does not typically include code which alters the LLM's internals, such as fine-tuning or activation steering. People use scaffolding because it can allow LLMs to use tools, reduce error rates, search for info, all of which make LLMs more capable. Common types of scaffolds include: Prompt templates: Perhaps the simplest scaffolds, "prompt templates" are just prompts with some blank sections to be filled in at runtime using some local data. E.g., a template like "You are a good assistant. The date is INSERT DATE HERE. Tell me how many days are left in the month." can be filled in with the current date before being used to prompt an LLM. Retrieval Augmented Generation (RAG): For tasks which have some associated data, RAG is a way of automatically adding task-relevant info to prompts. RAG does this by looking for snippets of text/data in the database which are relevant to the user's prompt, and then adding those snippets to the prompt. Search engines: Similar to RAG, giving an LLM access to a search engine lets it find info relevant to its prompt. Unlike RAG, the LLM decides what to search for and when. Agent scaffolds: Scaffolds which try to make LLMs into goal-directed agents, i.e., giving an LLM some goal/task to perform, and ways to observe and act on the world. Usually, the LLM is put into a loop with a history of its observations and actions, until its task is done. Some agent frameworks give LLMs access to much more powerful high-level actions, which saves the AI from repeating many simple primitive actions and, therefore, speeds up planning.

Agent scaffolds are characterized by their capacity for perception, planning, and action within an environment, often governed by an iterative process that refines, reflects, and improves the solution over multiple cycles.

While agents scaffolds can be used in different domains and wide array of tasks, there has been increasing interest dedicated to developing agents capable of automating the machine learning research and engineering process. A notable example is the [AI scientist] system, which presents an end-to-end pipeline designed to automate the research workflow. Currently, data science, machine learning engineering, and more generally, coding agents and assistants have shown consistent improvements. as we now see agents like Gihub Capilot, Cognition | Introducing Devin, Cursor, or Claude code exhibit phenomenon coding abilities, but there is no significant progress in their open-source coding agents.

The nature of coding problems presents a unique challenge for LLMs compared to other reasoning tasks, such as mathematics. In mathematical reasoning, there is often a single, unambiguously verifiable answer that can be derived through logical deduction. Conversely, generating a coding solution involves not only producing syntactically correct and executable code, but also navigating a vast solution space, where multiple functionally equivalent but structurally diverse implementations can exist. Crucially, the correctness and optimality of a coding solution often necessitate empirical validation through actual code execution and looking at traceback, rather than purely symbolic or logical verification. This inherent complexity, which demands iterative refinement and empirical testing, underscores the critical need for sophisticated agentic systems that can autonomously generate, execute, and iteratively refine solutions, even for highly capable foundational models.

2.1.6 Relevance and Challenges in Coding and ML Engineering Tasks

Machine learning engineering and coding tasks represent particularly challenging yet highly relevant applications for inference-time scaling due to their inherent complexity and precision requirements. Benchmarks like HumanEval, MBPP, and MLE-Bench exemplify these tasks, providing challenging evaluation contexts.

However, significant gaps persist in current research, such as limited open-source implementations and insufficient evaluation on realistic, complex coding tasks. This project explicitly addresses these gaps by systematically applying inference-time scaling techniques to small-scale distilled reasoning models, aiming to substantially enhance their performance and practical applicability in coding and ML engineering domains.

CITATION NEEDED

2.2 Related Work

3. Experimental Setup

in this chapter, we describe the Experimental setup, explain the design choices that went into adapting, building and evaluating our methods, starting with the agent scaffold, the Mlebench benchmark and the implemented ITS strategies

This chapter details the comprehensive methodology employed in this research, encompassing the experimental setup, the design choices made in adapting and building upon existing frameworks, and the specific procedures for evaluating the proposed inference-time scaling (ITS) strategies. The primary objective is to establish a transparent, rigorous, and replicable evaluation framework capable of quantitatively assessing the impact of these ITS techniques when applied to open-source Large Language Models (LLMs) engaged in complex machine learning engineering tasks.

3.1 AIDE: AI-Driven Exploration in the Code Space

At the center of this work, and the most critical component in my experimental setup is the agent scaffold itself, For this research work, I am leveraging AI-Driven Exploration (Aide), an open source agent scaffold, developed by WecoAi (Schmidt et al., 2025). AIDE is specifically designed to automate the trial and error process in developing machine learning models, working iteratively to find a solution in a tree search manner, exploring the 'Code space'.

The core principle behind AIDE is to address the significant amount of time that engineers and scientists spend on iterative experimentation, rather than to focus on conceptualizing and innovating. To achieve that, AIDE frames the entire machine learning engineering process as a code optimization problem, modeling the trial and error as a tree search within a space of potential code solutions, each node is a potential solution problem (i.e. a code script), and each edge is an attempt at improving/debugging that solution.

At its heart. and as many other agent scaffolds, AIDE is powered by Large Language Models (LLMs), typically a strong model like gpt4 or Claude. These LLMs are responsible for proposing new code, debugging existing scripts, or suggesting and improvement or refinement to promising solutions. AIDE then reuses and refines these solutions, effectively trading computational resources for improved performance. Essentially, AIDE is performing some sort of inference time scaling, but the scope and the level of this scaling is of high-level nature. The implementation of AIDE is publicly available, which was a key factor in its selection for this thesis, allowing integration of custom inference time scaling methods.

Acknowledgements

I want to acknowledge AIMS and it's funders for supporting this work, as well as my supervisor, Prof Arnu Pretorius from University of Stellenbosch, and my co-supervisor, Arnol Fokam from InstaDeep.

I would also like to thank my family and friends, and my colleagues at AIMS for their support and encouragement.

References

Adolphson, A., Sperber, S., and Tretkoff, M., editors. *p-adic Methods in Number Theory and Algebraic Geometry*. Number 133 in Contemporary Mathematics. American Mathematical Society, Providence, RI, 1992.

Aitkin, M. A., Francis, B., Hinde, J., and Darnell, R. *Statistical modelling in R*. Oxford University Press Oxford, 2009.

Beardon, A. From problem solving to research, 2006. Unpublished manuscript.

Davey, M. *Error-correction using Low-Density Parity-Check Codes*. Phd, University of Cambridge, 1999.

Doe, J. A placeholder article. *Journal of Placeholder Research*, 1:1–10, 2023.

Lamport, L. *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.

MacKay, D. Statistical testing of high precision digitisers. Technical Report 3971, Royal Signals and Radar Establishment, Malvern, Worcester. WR14 3PS, 1986a.

MacKay, D. A free energy minimization framework for inference problems in modulo 2 arithmetic. In Preneel, B., editor, *Fast Software Encryption (Proceedings of 1994 K.U. Leuven Workshop on Cryptographic Algorithms)*, number 1008 in Lecture Notes in Computer Science Series, pages 179–195. Springer, 1995b.

MacKay, D. J. C. and Neal, R. M. Good codes based on very sparse matrices. Available from www.inference.phy.cam.ac.uk, 1995.

Shannon, C. A mathematical theory of communication. *Bell Sys. Tech. J.*, 27:379–423, 623–656, 1948.

Shannon, C. The best detection of pulses. In Sloane, N. J. A. and Wyner, A. D., editors, *Collected Papers of Claude Shannon*, pages 148–150. IEEE Press, New York, 1993.

Webots. Commercial mobile robot simulation software. Webots, www.cyberbotics.com, Accessed April 2013.

Wikipedia. Black scholes. Wikipedia, the Free Encyclopedia, http://en.wikipedia.org/wiki/Black%E2%80%93Scholes, Accessed April 2012.