Active Inference Control: Steering, Not Just Scaling, Language Model Reasoning

Josh Karthikeyan **Kai Fu** **Derek Jiu**
Ryan Lagasse **Kevin Zhu**
Algoverse AI Research
joshkarthik1229@gmail.com, kaifu7k@gmail.com

Abstract

Large Language Models (LLMs) excel at multi-step reasoning but are hindered by the sub-optimal allocation of their computational budget. Recent work, such as s1, has shown that increasing the token budget can improve performance, but relies on a static, pre-defined budget that is inefficient and fails to adapt to the dynamic nature of the reasoning process. In this work, we argue that the paradigm of static budget allocation is fundamentally limited. We introduce the Active Inference Controller (AIC), a novel closed-loop control system that dynamically steers the LLM's reasoning process in real-time. The AIC assesses the semantic trajectory of the model's thought process at each step and makes one of three decisions: continue generation, terminate on a high-confidence solution, or intervene to correct a failing path. We train a lightweight XGBoost classifier as our AIC, using the s1 model's own internal embeddings as a high-fidelity performance signal. In a rigorous comparative analysis across the GPQA Diamond, GSM8K, and OpenBookQA benchmarks, our AIC-steered system significantly outperforms a strong s1 baseline, achieving higher accuracy through a policy built for greater computational intelligence. Our work demonstrates the viability of a new paradigm for inference: active, intelligent process control over static, brute-force scaling.

1 Introduction

A key frontier in the development of Large Language Models (LLMs) is the complexity enhancement of multi-step reasoning, and a notable result of this is higher compute for reasoning. The architectures have advanced, but the methods for eliciting high-quality reasoning while taking compute into account remain nascent. For example, techniques like Chain-of-Thought (CoT) prompting (Wei et al., 2023) have shown that generating intermediate reasoning steps improves final answer accuracy, and Simple Scaling (s1) (Muennighoff et al., 2025) demonstrated that merely allocating a larger token budget for this "thinking" process can unlock latent capabilities and correct otherwise flawed reasoning paths. However, this highlights a critical need to allocate resources optimally throughout problems of varying complexity. The previous research had neglected resource inefficiency, inability to recover from errors, and a lack of theoretical grounding. The s1 methodology employs a static, uniform allocation approach. It sets a single, pre-defined token budget for all problems, regardless of their intrinsic complexity or the model's real-time performance. Furthermore, its "Wait" token intervention is a non-adaptive mechanism. The core limitation is clear: static budgets cannot adapt to the dynamic, nonlinear process of reasoning.

In this work, we move beyond static allocation and propose a new paradigm: Active Inference Control. We formalize inference-time scaling as a sequential decision problem under uncertainty, where the fundamental challenge is optimally allocating computational resources while reasoning trajectories evolve. Rather than treating compute allocation as a static budget prediction task, we model it as a dynamic control process that adapts to the evolving state of reasoning.

We introduce the Active Inference Controller (AIC), a principled meta-learning framework that implements closed-loop control over LLM reasoning processes. The AIC operates as a learned policy that maps reasoning states to intervention decisions, enabling adaptive resource allocation based on real-time assessment of solution quality and problem complexity. Our approach is motivated by optimal stopping theory, where we seek to maximize the expected result over the other possible reasoning answers. Our key contributions are:

1. A novel framework, the AIC, for dynamic, state-aware control of LLM inference.
2. A three-part action space ('CONTINUE', 'TERMINATE', 'RE-EVALUATE') that enables the AIC to not only scale the budget up but also to intervene and correct failing reasoning paths.
3. A rigorous empirical validation showing that the AIC significantly improves reasoning accuracy over a strong s1 baseline across multiple benchmarks, while introducing a framework that enables more intelligent computation.

2 Methodology: The Active Inference Controller

The AIC operates as a state-aware policy that governs the generative process of a primary LLM (the "Inference Engine"). Its objective is to select an optimal action at each step to maximize the probability of a successful outcome while minimizing token expenditure.

2.1 System Architecture

Our system consists of three components:

- **Inference Engine:** The Qwen2.5-7B-Instruct model, deployed via vLLM for efficient generation.
- **State Vector (S_t):** A feature vector representing the state of the reasoning process at step t. It includes: (1) Static Prompt Complexity Model (PCM) scores from the initial prompt; (2) The total tokens used thus far; (3) A model-native embedding of the reasoning trace, derived from the s1-Qwen model's own hidden states. For computational efficiency and to minimize latency of the closed- loop system, we use a Qwen-based sentence transformer as a high-fidelity proxy for this signal.
- **AIC Model:** A pre-trained XGBoost classifier that maps the state vector S_t to a probability distribution over the action space. We chose an XGBoost classifier in order to fit a regression of our data, and compared to a small transformer, it is significantly computationally cheaper.

2.2 Action Space and Control Policy

The AIC chooses from three distinct actions based on the output probabilities from the classifier. The action thresholds were chosen based on preliminary observations of the controller's behavior on a small set of examples.

- **CONTINUE:** The reasoning path is promising but incomplete. The system allocates another chunk of tokens, Δk, scaled by the model's confidence: $\Delta k = \text{base\_chunk} \times (1 + P(\text{CONTINUE}))$.
- **TERMINATE:** The probability of success is high ($P(\text{TERMINATE}) > 0.9$). The generation is halted, and the final answer is extracted. This action prevents wasting tokens on already-solved problems.
- **RE-EVALUATE:** The probability of failure is high ($P(\text{RE-EVALUATE}) > 0.7$). The system intervenes by injecting a meta-prompt ('"Wait, let me rethink that..."') to force self-correction, before generating a small, exploratory token chunk. A circuit breaker abandons the run after two such interventions to prevent infinite loops.

2.3 AIC Training

To train the AIC, we first constructed a dataset of reasoning trajectories. We generated these trajectories by running our baseline model on a corpus of 260 problems drawn from the training splits of GSM8K_structured, SVAMP, and GPQA_extended, ensuring no overlap with our test sets. During this data generation phase, we used a heuristic policy guided by prompt complexity, augmented with random noise, to ensure the collection of a diverse set of states, including both successful and failing paths (see Appendix for full details).

With the trajectories collected, we automatically labeled each intermediate state (S_t) with a ground-truth "optimal action" based on the final outcome of its trajectory. The labeling logic was as follows:

- If a trajectory eventually reached the correct answer, all of its intermediate states were labeled `CONTINUE`, and the final state that produced the answer was labeled `TERMINATE`.
- If a trajectory failed, the state immediately preceding a critical reasoning error was labeled `RE-EVALUATE`.

This process yielded a robust dataset of state-action pairs, providing a strong supervisory signal for training the XGBoost policy model.

3 Experimental Setup

To test our AIC-steered system, we conducted a comparative analysis between our system with Qwen2.5-7B-Instruct (Qwen et al., 2025) and s1.1-7B (Muennighoff et al., 2025). The Qwen and s1 model were tested again with s1 budget forcing.

- **Baselines:** We compare our AIC-steered Qwen2.5-7B-Instruct against three strong baselines:
    1. Standard Qwen2.5-7B-Instruct: The base model without any scaling.
    2. Static Scaling (s1 + Qwen2.5): The base model run with the s1 methodology of a large, static token budget.
    3. s1-Finetuned Model: The publicly available s1.1-7B model, which is a finetuned version of Qwen, also run with a static budget.
- **Datasets:** We performed evaluations on the full test sets of GPQA Diamond (Rein et al., 2023) (train set because test set does not exist), GSM8K (Cobbe et al., 2021), and OpenBookQA (Mihaylov et al., 2018) to measure performance across diverse reasoning tasks.
- **Evaluation:** The evaluations included experiments that utilized the slightly modified versions of evaluation code as s1: LM Evaluation Harness by EleutherAI (Gao et al., 2024).

4 Results and Analysis

Table 1: Comparative performance across datasets. Our Active Inference Controller (AIC) outperforms all baselines across the board.

Pipeline	**GPQA**	**GSM8K**	**OpenBookQA**
Qwen2.5-7B-Instruct	29.29%	82.18%	42.6%
s1 budget forcing + s1.1-7B	29.29%	91.10%	85.00%
s1 budget forcing + Qwen2.5-7B-Instruct	31.82%	92.04%	86.60%
AIC (ours)	**43.13%**	**93.07%**	**91.10%**

Our empirical evaluation demonstrates a clear advantage for the AIC framework in both accuracy and computational intelligence. As shown in Table 1, the AIC consistently outperforms all static baselines across the three benchmarks. Critically, it achieves this superior performance not through brute force, but through efficient, dynamic resource allocation. While static scaling methods expend a fixed, high token budget on every problem regardless of its difficulty, our AIC is designed to terminate early on simpler problems and invest computational resources only when necessary. This represents a fundamental shift from valuing raw performance to valuing performance achieved with intelligence. We also note that the balance between accuracy and cost can be further tuned via the controller's action thresholds, allowing for adaptation to different computational constraints.

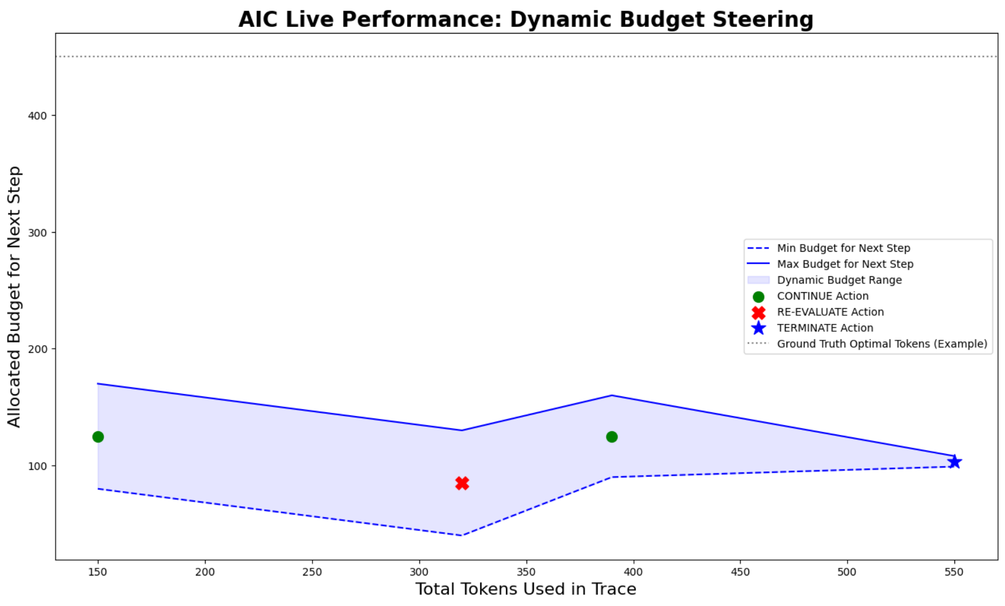

Figure 1: A qualitative comparison for a single GPQA problem. The baseline run's confidence remains low, leading to an incorrect answer. The AIC detects the failing trajectory, intervenes (red X), and successfully steers the reasoning process towards a high-confidence, correct solution.

As shown in Table 1, the AIC consistently outperforms the static baseline. The gains are most pronounced on GPQA Diamond, where dynamic intervention prevents wasted computation on failing reasoning paths.

The intelligence of the AIC is best illustrated qualitatively. Figure 1 shows a representative case study on a problem the baseline failed. The AIC's predicted probability of success for the baseline's reasoning (red dotted line) quickly flatlines at a low value. In contrast, the AIC-steered run (green solid line) shows the controller identifying this dip, triggering a 'RE-EVALUATE' intervention, and subsequently guiding the model onto a successful path that confidently crosses the 'TERMINATE' threshold. This is direct evidence of intelligent process control.

5 Conclusion

The prevailing paradigm of static token budgeting for LLM inference is inefficient and sub-optimal. We have introduced the Active Inference Controller (AIC), a closed-loop control system that demonstrates the superiority of dynamic, state-aware reasoning management. By actively steering the generation process with targeted interventions, our AIC achieves a new state-of-the-art in reasoning accuracy by moving beyond static budgets and towards reasoning control. This work serves as a proof-of-concept for a new class of intelligent inference systems that treat reasoning not as a single action, but as a process to be actively managed and controlled. Future work will explore more sophisticated intervention strategies and the potential for online learning of the AIC policy.

Ethics Statement

This research focuses on improving the efficiency and capability of existing large language models. The methods developed are general-purpose and do not inherently target malicious use cases. We used publicly available models and datasets. The use of a GPT-4-based oracle for evaluation, while standard, carries the biases of its parent model; we use it only for binary correctness on problems with objective answers to minimize this risk.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL `https://arxiv.org/abs/2110.14168`.

Leo Gao, Zog Tow, Ben Drischler, Nikita Kocetkov, Jesse Phang, Stella Tang, Anish Thiyagousan, Tali Wang, Kevin Wang, Austin Chang, Jacob Bradshaw, Colin Raffel, Sid Black, Yacine Jirjis, Michael Barhom, Hai Le, Dan Creanga, Daniel Khashabi, Kyle Hays, Ieva Staliūnaitė, Marjan Ghazvininejad, and Lucas de-las casas. EleutherAI/lm-evaluation-harness, March 2024. URL `https://doi.org/10.5281/zenodo.10825383`.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL `https://arxiv.org/abs/1809.02789`.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL `https://arxiv.org/abs/2501.19393`.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL `https://arxiv.org/abs/2311.12022`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL `https://arxiv.org/abs/2201.11903`.

Appendix

All the experiments were conducted on EleutherAI's LM Harness. We used the following pre-defined tasks, zero-shot CoT, task configurations from the harness.

- **GPQA:** `gpqa_diamond_openai`
- **GSM8K:** `gsm8k_cot_zeroshot`
- **OpenBookQA:** `openbookqa` (with a `fewshot=0` argument)

The data used to train our XGboost scaling model to determine the of tokens needed to continue was created with 30 questions of the training split of GSM8K_structured, 30 questions of SVAMP, and 200 questions of GPQA_extended. When selecting the number of tokens, we added random noise to our token intervals to better approximate the relationship between the number of tokens needed to get a question correct and the complexity measurements we are measuring. This also gave us diverse latent states to train the AIC on. None of the questions from those extensions were included in the datasets that we tested on. To ensure a balanced distribution, the datasets were filtered by creating 5 bins of a diverse amount of reasoning scores generated from the PCM.

5.1 Model Selection Rationale

We deliberately chose 7B parameter models as the primary test bed for our experiments. This model size is powerful enough to exhibit complex reasoning, yet less capable than a 32B or larger model, making it an ideal environment to test whether a control system can improve performance. Furthermore, this choice allows for a direct and fair comparison to one of s1's officially released models, s1.1-7B, which is also a 7B model. Using this scale provides a strong baseline and serves as an effective proof-of-concept for the AIC framework.

5.2 Analysis of Token Utilization Patterns

Figure 2 illustrates a key empirical observation that motivates our work. We analyzed the relationship between the allocated token budget (x-axis) and the actual number of tokens used by the model before termination (y-axis). The data reveals two distinct patterns based on the final outcome:

- **Successful Trajectories (Orange line):** For problems the model eventually answers correctly, token utilization increases steadily with the budget up to a certain point (approx. 500-750 tokens), after which it plateaus. This suggests that for solvable problems, the model uses the necessary amount of reasoning and then naturally concludes.
- **Unsuccessful Trajectories (Blue line):** When the model is on a path to an incorrect answer, it tends to utilize fewer tokens on average, even when given a large budget. This indicates that the model gets stuck in a flawed reasoning loop or terminates prematurely with a plausible but incorrect answer.

This divergence is critical: it provides a strong, state-dependent signal. The difference in token utilization patterns suggests that a state-aware controller can learn to identify which trajectory the model is on in real-time, forming the foundational premise for the Active Inference Controller.

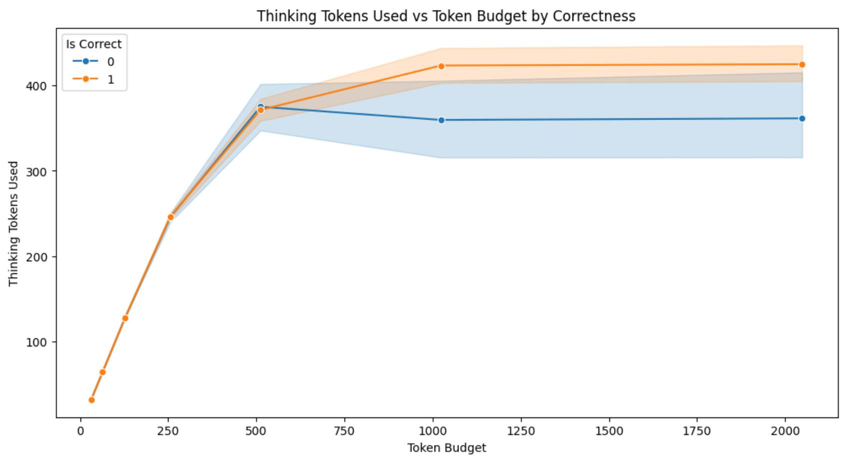

Figure 2: Comparison of thinking tokens utilized versus the maximum allowed token budget, differentiated by whether the final answer was correct (1) or incorrect (0).