Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2026) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

30th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2026)

Beyond Arithmetic: The Semantic Bottleneck in Neuro-Symbolic LLMs

Bogdan-Marius Dinu[a,*], Ciprian-Mihai Ceausescu[a], Cristian-Hacic Kevorchian[a]

[a] *Faculty of Mathematics and Computer Science, University of Bucharest, Romania*

Abstract

We propose a novel multi-agent distributed framework that leverages neuro-symbolic reasoning to improve deterministic mathematical problem-solving on edge-scale Large Language Models (LLMs). Our method employs a hybrid reasoning paradigm, where a lightweight edge model (DeepSeek-R1 8B) acts as a logic orchestrator augmented by a semantic Retrieval-Augmented Generation (RAG) module to generate executable Programs-of-Thought (PoT). The exact calculations are offloaded to a deterministic symbolic engine (SymPy), while an adaptive self-healing mechanism autonomously routes unresolved queries to a cloud-based baseline (Gemini 2.5 Flash) to ensure fault tolerance. We validate our approach on two challenging benchmark datasets, GSM8K and MATH500, demonstrating its effectiveness in systematically mitigating execution-level arithmetic hallucinations and enhancing accuracy across complex reasoning tasks. The results show consistent improvements over both native, un-augmented edge models and massive cloud baselines. These findings underscore the potential of distributed neuro-symbolic architectures for robust, generalizable, and resource-efficient mathematical reasoning in artificial intelligence applications.

Keywords: neuro-symbolic reasoning; hallucination mitigation; edge computing; semantic bottleneck

1. Introduction

The development of Large Language Models (LLMs) has fundamentally transformed the landscape of artificial intelligence, shifting the paradigm from task-specific algorithms to highly versatile, generalized reasoning engines. However, despite their profound linguistic fluency and capacity for semantic pattern recognition, LLMs inherently struggle with deterministic tasks such as complex mathematical problem-solving. This critical limitation stems from their autoregressive, probabilistic architecture, which frequently generates *arithmetic hallucinations*, outputs that are logically structured and syntactically convincing, yet numerically incorrect. Although current approaches attempt to mitigate these errors by exponentially scaling model parameters, this brute-force strategy imposes prohibitive com-

* Corresponding author.
E-mail addresses: bogdan-marius.dinu@s.unibuc.ro (Bogdan-Marius Dinu)., ciprian-mihai.ceausescu@drd.unibuc.ro (Ciprian-Mihai Ceausescu)., cristian.kevorchian@unibuc.ro (Cristian-Hacic Kevorchian).

putational, memory, and energy costs. Consequently, relying on massive monolithic models severely restricts the deployment of advanced mathematical reasoning in resource-constrained or asynchronous edge computing environments.

In response to these algorithmic and infrastructural challenges, contemporary research has explored a spectrum of mitigation techniques. Advanced prompt engineering methodologies, such as Chain-of-Thought (CoT) and Tree-of-Thoughts (ToT) [1] reasoning, attempt to force the model to explicitly articulate intermediate logical steps, thereby improving the coherence of the final output. Concurrently, tool-augmented language models have been developed to offload rigid arithmetic operations to external calculators or Python environments [2]. While these interventions substantially improve baseline performance, they are still prone to severe error propagation during deep, multi-step problem-solving. Isolated code generation mechanisms frequently fail when models lack the immediate domain knowledge necessary to formulate the exact mathematical rules, proving that purely probabilistic generation remains fundamentally incompatible with tasks demanding absolute precision.

To bridge the gap between neural flexibility and strict symbolic determinism, we propose a Multi-Agent Distributed Neuro-Symbolic framework explicitly optimized for edge-to-cloud architectures. Rather than forcing a lightweight model to guess mathematical outputs, our system redefines the edge LLM as a high-level semantic logic orchestrator. Operating on a heavily quantized edge model (DeepSeek-R1 8B), the framework integrates a specialized Retrieval-Augmented Generation (RAG) module. This module dynamically queries a vector database (ChromaDB) to inject targeted mathematical axioms, formulas, and rules directly into the model's context window. The LLM then synthesizes this retrieved knowledge to generate a highly structured, executable Program-of-Thought (PoT) script. The actual calculation is then entirely offloaded to a deterministic symbolic computation engine (SymPy), which guarantees(by its nature) mathematically flawless execution. Crucially, to address the inherent limitation of autonomous code generation, our architecture features an adaptive self-healing mechanism. If the local symbolic execution fails due to syntactic or logical errors, the system autonomously reroutes [3] the reasoning task to a high-capacity cloud-based baseline (Gemini 2.5 Flash), establishing a robust, fault-tolerant distributed reasoning network.

We comprehensively validate our proposed framework against four rigorous benchmarks: (i) the **GSM8K dataset** [4], which tests multi-step grade-school mathematical word problems; (ii) the **SVAMP dataset** [5], which evaluates semantic robustness against linguistic traps; (iii) the **MATH500 dataset** [6], comprising complex, competition-level algebraic and geometric challenges; and (iv) the **AIME 2025** [7] problem set, representing elite-level mathematical logic and extreme deductive reasoning. Our empirical evaluations demonstrate that wrapping an 8-billion-parameter model in a deterministic neuro-symbolic pipeline significantly amplifies its precision. By systematically eliminating computational hallucinations at the execution layer, our augmented edge model achieves competitive parity with, and in specific ablation configurations, surpasses the performance of un-augmented, native large-scale baseline models.

In summary, the primary contributions of this work are threefold: (i) the architectural design and deployment of a hybrid neuro-symbolic reasoning pipeline that empowers resource-constrained edge LLMs to execute deterministic, mathematically sound problem-solving without algorithmic hallucination; (ii) the novel integration of a domain-specific semantic RAG module coupled with an autonomous, self-healing edge-to-cloud fallback protocol, ensuring absolute system resilience; (iii) an extensive experimental analysis across the GSM8K [4] and MATH500 [6] benchmarks, providing empirical evidence that targeted symbolic orchestration effectively bridges the computational capability gap between lightweight edge agents and massive, parameter-heavy cloud systems.

2. Related work

2.1. Neural Paradigms and Prompt-Based Mathematical Reasoning

Initial approaches to mathematical reasoning in Large Language Models (LLMs) primarily focused on scaling parameter counts and refining prompt engineering strategies [8]. In practice, however, these autoregressive models encounter a fundamental limitation: they predict the next plausible token probabilistically rather than computing arithmetic operations deterministically. To mitigate this, researchers introduced structured prompting techniques such as Chain-of-Thought (CoT) and Tree-of-Thoughts (ToT). By forcing the model to articulate its reasoning step-by-step, these methods minimize logical gaps and enhance error traceability. Advanced iterations even employ multi-

path sampling and majority voting. Nevertheless, the underlying mechanism remains fundamentally probabilistic. As problem complexity increases, such as in the MATH500 [6] dataset, models become significantly more prone to errors, particularly during intermediate calculations. These findings demonstrate that while LLMs excel at modeling semantic relationships, they remain unreliable for exact numerical derivation.

2.2. Tool-Augmented Language Models and Program-of-Thought

Recognizing the intrinsic limitations of purely parametric reasoning, the field has increasingly shifted toward Tool-Augmented Language Models (TALMs) [2]. This approach offloads complex arithmetic operations to external symbolic solvers. A prominent development in this domain is the Program-of-Thought (PoT) paradigm. Unlike CoT, which expresses intermediate reasoning in natural language, PoT leverages the model's proficiency in programming languages to translate mathematical logic into executable scripts, such as Python code. By offloading the calculation to a CPU, PoT entirely circumvents the risk of arithmetic hallucination at the execution layer. However, isolated code generation exhibits its own form of algorithmic limitation. When edge-scale models suffer from parametric knowledge gaps regarding specific axioms or formulas, they tend to produce syntactically sound but logically invalid code. Therefore, while PoT secures the determinism of the final calculation, its success is completely restricted by the model's un-augmented capacity to retrieve and formulate the correct mathematical logic a priori.

2.3. Retrieval-Augmented Generation for Structured Knowledge

To address the limitations of lightweight models, Retrieval-Augmented Generation (RAG) [9] functions as a critical framework for integrating external, verifiable knowledge into the reasoning workflow. Originally developed for open-domain question-answering, RAG architectures utilize high-dimensional vector databases to map semantic similarity between user queries and external document corpora. In the context of mathematical reasoning, recent adaptations of RAG have transitioned from retrieving general text to retrieving highly structured domain knowledge, including axiomatic rules, geometric theorems, and modular algebraic formulas. By dynamically injecting this specific context into the model's context window, RAG provides a factual grounding that dramatically reduces the cognitive load on the LLM. By using external knowledge, smaller models no longer need to memorize every mathematical fact during the pre-training phase, allowing the model to allocate its computational resources entirely to semantic parsing and code formulation.

2.4. Neuro-Symbolic Architectures and Distributed Fallback Mechanisms

The synthesis of neural networks' pattern recognition capabilities with the strict deterministic execution of symbolic logic engines defines the frontier of Neuro-Symbolic AI. While existing hybrid systems [10] have successfully integrated LLMs with symbolic solvers like SymPy, they often assume a monolithic, high-resource deployment environment. Our approach diverges by addressing the infrastructural realities of edge computing. The challenge of maintaining high accuracy under hardware constraints shares architectural parallels with challenges found in other complex AI domains; for instance, the utilization of multi-model collaborative frameworks has proven highly effective in mitigating distribution shifts and data scarcity in medical image segmentation. Drawing inspiration from the robustness of multi-model paradigms, our work extends this collaborative philosophy into the realm of deterministic logic. We implement a distributed, multi-agent architecture where the neuro-symbolic pipeline is primarily orchestrated by a quantized edge model. To ensure absolute systemic resilience against edge-case programmatic failures, our framework incorporates an adaptive self-healing protocol. This mechanism monitors the symbolic execution environment and, upon detecting unrecoverable syntactic or logical errors, autonomously reroutes the reasoning task to a state-of-the-art cloud baseline. This hybrid edge-cloud orchestration creates a resilient reasoning framework, balancing localized computational efficiency with the advanced analytical power of large-scale models.

3. Proposed Architecture

To achieve deterministic mathematical reasoning on resource-constrained devices, we propose a multi-agent distributed architecture. As illustrated in Figure 1, the pipeline strictly separates the logical orchestration from the computational execution, dividing the workload between an Edge Computing Zone and a Cloud Computing Zone.

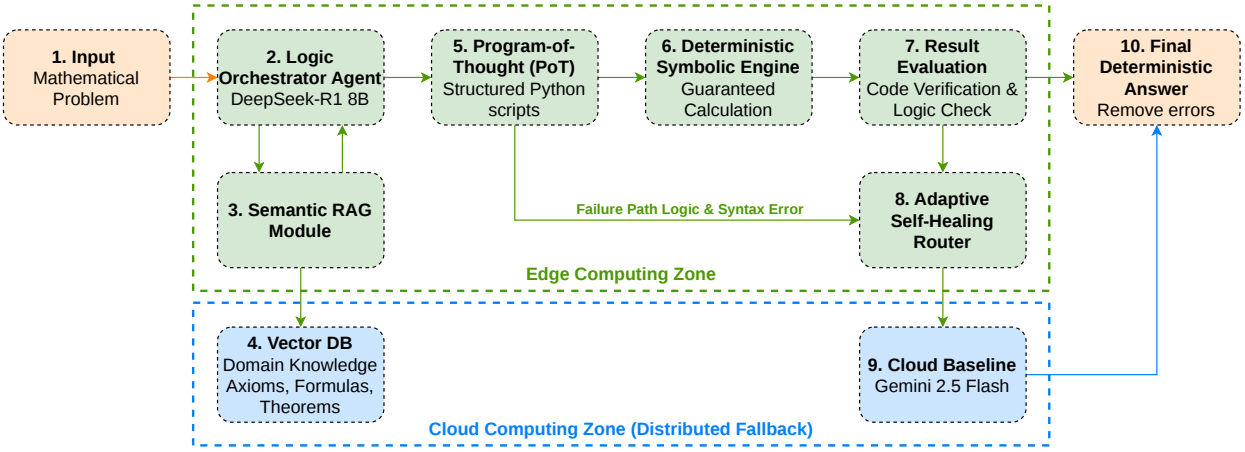

Fig. 1. **The proposed Multi-Agent Distributed Neuro-Symbolic Architecture**. The framework illustrates a bifurcated reasoning pipeline designed to mitigate stochastic hallucinations in mathematical problem-solving. (I) The Edge Computing Zone utilizes a specialized Logic Orchestrator (DeepSeek-R1 8B) to perform semantic parsing and synthesize Program-of-Thought (PoT) scripts. These scripts are executed within a Deterministic Symbolic Engine (SymPy) to ensure computational rigor. An Adaptive Self-Healing Router manages recursive error correction by feeding execution tracebacks back into the orchestrator. (II) The Cloud Computing Zone provides a distributed fallback mechanism, utilizing a Semantic RAG module backed by ChromaDB for axiomatic grounding and a frontier model (Gemini 2.5 Flash) for high-complexity contingencies.

The workflow initiates with a *Mathematical Problem Input*, which is routed to the *Logic Orchestrator Agent* (powered by the DeepSeek-R1 8B model) located within the Edge Computing Zone. To overcome the model's limited parametric memory, the agent queries a *Semantic RAG Module*. This module retrieves relevant domain knowledge such as axioms, formulas, and theorems from a dedicated *Vector Database*. As depicted in Figure 2, we present a 3D t-SNE projection of this vector space, demonstrating the semantic clustering of distinct mathematical domains (e.g., combinatorics, geometry, number theory). This geometric organization enables the RAG module to accurately retrieve the most contextually relevant logical axioms.

Supported by the additional information, the orchestrator generates a *Program-of-Thought (PoT) Script* consisting of structured Python code. This script is passed to the *Deterministic Symbolic Engine* (powered by the SymPy library) to perform a guaranteed calculation. The raw output undergoes a *Result Evaluation* step for strict code verification and logic checks. If successful, the system outputs the *Final Deterministic Answer*, effectively bypassing arithmetic hallucinations during the calculation phase.

Crucially, the architecture accounts for the inherent brittleness of code generation. If the symbolic execution encounters a syntax error or a logical failure (Failure Path), the *Adaptive Self-Healing Router* intercepts the process. To ensure uninterrupted fault tolerance, the router delegates the unresolved problem to the *Cloud Baseline* (Gemini 2.5 Flash) within the Cloud Computing Zone. This distributed fallback mechanism guarantees a high-accuracy response even when local edge processing fails.

4. Methodology

Our framework receives as input a natural language mathematical problem and seeks to provide a deterministic numerical or algebraic execution path. As illustrated in Figure 1, the pipeline is divided into an Edge Computing Zone and a Cloud Computing Zone, interconnected by an adaptive routing mechanism. Our goal is to extract the exact final answer while strictly preventing arithmetic hallucinations.

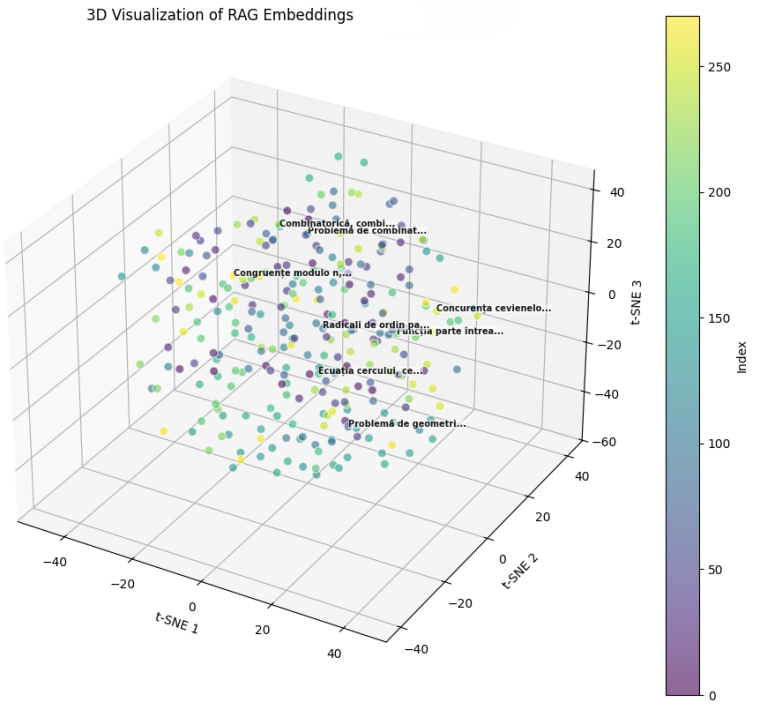

Fig. 2. **3D t-SNE Visualization of Domain Knowledge Embeddings.** The plot illustrates the underlying semantic manifold where distinct mathematical rules and axioms form cohesive clusters, facilitating high-precision angular retrieval for the neuro-symbolic orchestrator. The axes (t-SNE 1, t-SNE 2, and t-SNE 3) represent the synthetic low-dimensional coordinates generated by the t-distributed Stochastic Neighbor Embedding algorithm. While these individual axes do not possess intrinsic semantic meaning, their relative spatial arrangement preserves local neighborhoods from the high-dimensional vector space, ensuring that mathematically related concepts are mapped into close spatial proximity.

4.1. Semantic Retrieval-Augmented Generation (RAG)

To mitigate the limited parametric memory of the quantized edge model, we introduce a semantic retrieval module. We retrieve the most relevant mathematical hint by maximizing the cosine similarity between the problem embedding and the rule embeddings:

$$H = \arg\max_{r_i \in \mathcal{D}} \left(\frac{E(P) \cdot E(r_i)}{\|E(P)\| \|E(r_i)\|} \right), \tag{1}$$

where H represents the retrieved mathematical hint, $\mathcal{D} = \{r_1, r_2, \ldots, r_n\}$ is the domain knowledge base consisting of mathematical rules and axioms, r_i is an individual rule within the database, $E(\cdot)$ denotes the high-dimensional vector embedding function, P is the input mathematical problem, and $\|\cdot\|$ represents the vector norm. This reliance on cosine similarity is not merely heuristic; as recently demonstrated in the context of hyperspherical semantic manifolds [11], the angular distance between embeddings serves as a rigorous geometric primitive for logical truth grounding. Consequently, maximizing this similarity ensures that the injected axioms possess maximal logical coherence relative to

the problem space. If the similarity score falls below a predefined threshold τ, H is set to $\emptyset$, meaning no specific rule is injected.

4.2. Neuro-Symbolic Program-of-Thought (PoT) Generation

The Logic Orchestrator Agent acts as a semantic parser and code generator. We construct a joint prompt by concatenating the original problem, the retrieved hint, and a strict system instruction set. The edge model generates a Program-of-Thought script: $C = \mathcal{M}_{edge}(P \oplus H)$, where C is the generated deterministic computational graph (Python script), $\mathcal{M}_{edge}$ represents the Logic Orchestrator Agent (DeepSeek-R1 8B), P is the initial input problem, H is the retrieved context, and $\oplus$ denotes the text concatenation operation.

4.3. Deterministic Symbolic Execution

The generated script C is passed to an isolated Deterministic Symbolic Engine, which safely executes the Python code. The Result Evaluation module applies a verification function to extract the final numerical value. We define the state of the local extraction as:

$$v(y_{edge}) = \begin{cases} \text{valid output}, & \text{if } C \text{ executes without syntax or logic errors} \\ \epsilon, & \text{if execution fails or output is malformed} \end{cases}, \tag{2}$$

where $v(\cdot)$ is the verification function, y_{edge} is the raw output from the local symbolic execution, C is the generated script, and ϵ denotes an unrecoverable error state (failed extraction).

4.4. Adaptive Self-Healing Routing

To ensure the pipeline is robust against the inherent limitations of autonomous code generation, we implement an Adaptive Self-Healing Router. If the local neuro-symbolic pipeline encounters an unrecoverable error, the router autonomously delegates the reasoning task to a high-capacity Cloud Baseline model. The final output of the entire distributed system is mathematically formalized as a piecewise function:

$$A_{final} = \begin{cases} v(y_{edge}), & \text{if } v(y_{edge}) \neq \epsilon \\ \mathcal{M}_{cloud}(P), & \text{if } v(y_{edge}) = \epsilon \end{cases}, \tag{3}$$

where A_{final} is the guaranteed deterministic answer, $v(y_{edge})$ is the validated local output, $\mathcal{M}_{cloud}$ represents the Cloud Baseline model (Gemini 2.5 Flash), P is the initial problem, and ϵ is the error state that triggers the fallback. This routing mechanism guarantees resource-efficient local processing while maintaining absolute fault tolerance.

5. Experimental evaluation

5.1. Datasets

GSM8K [4] is a dataset of high-quality linguistically diverse grade school math word problems created by human problem writers. The dataset consists of 8,500 problems, typically taking between 2 and 8 steps to solve. These problems primarily involve basic arithmetic operations (addition, subtraction, multiplication, and division) but require strong multi-step logical reasoning. In our experiments, we use the standard test split to evaluate the model's baseline arithmetic and logical orchestration capabilities.

MATH500 [6] is a curated subset of the comprehensive MATH benchmark, containing 500 challenging competition-level mathematics problems. The dataset covers seven distinct mathematical disciplines, including Algebra, Geometry, Number Theory, and Precalculus. Because it requires advanced theorem application and complex algebraic manipulations, this dataset serves as a serious test for our semantic RAG and SymPy-based symbolic execution modules.

AIME [7] (American Invitational Mathematics Examination) represents a collection of elite-level high school mathematical challenges. Historically used to qualify students for the US Math Olympiad, these problems require deep, non-standard analytical thinking. We incorporate recent AIME problem sets (AIME 2025) to evaluate the absolute upper bounds of our system's deductive reasoning and the efficacy of our cloud fallback mechanism when edge-scale models encounter extreme complexity.

SVAMP [5] (Simple Variations on Arithmetic Math word Problems) is a challenge dataset created by applying controlled variations to existing word problems. It contains approximately $1,000$ problems designed to test whether models genuinely understand mathematical logic or merely rely on superficial lexical heuristics. We include this dataset to validate the robustness of our semantic parsing stage against intentionally misleading phrasing.

Dataset grouping. Based on reasoning complexity and domain knowledge requirements, we group the evaluation datasets into two categories: (1) *Foundational reasoning*: GSM8K [4] and SVAMP (focusing on robust, multi-step arithmetic logic), and (2) *Advanced / Competition-level reasoning*: MATH500 [6] and AIME 2025 (focusing on algebra, theorems, and problem-solving scenarios).

5.2. Experimental setup

Dataset evaluation. Unlike traditional supervised training pipelines, our neuro-symbolic framework operates in an inference setting augmented by RAG. For all datasets (GSM8K [4], MATH500 [6], AIME 2025, and SVAMP [5]), we utilize their standard test splits to evaluate the system's performance, as this best reflects real-world, unseen problem-solving scenarios without data leakage.

Implementation details. The edge computing zone of our proposed architecture is deployed locally on a workstation equipped with an NVIDIA GeForce RTX 5070 Ti GPU, an AMD Ryzen 5 7600X CPU operating at 5.4 GHz, and 32 GB of RAM. The edge model, DeepSeek-R1 8B with Q4_K_M quantization is served locally using the Ollama framework to optimize inference latency and resource utilization on edge-grade hardware. For the cloud computing zone, the Gemini 2.5 Flash baseline is accessed through the Google AI Studio API (self-healing fallback mechanism).

Hyperparameter details. To balance the generation of deterministic code with semantic flexibility, the edge LLM is configured with a temperature setting of 0.3 and a standard Top-P Sampling of 0.95. This relatively low temperature restricts the model from generating overly creative but syntactically invalid Python scripts, anchoring the outputs in logical execution. Furthermore, to accommodate the extended context required by the semantic RAG module, which dynamically injects relevant axioms, formulas, and structural prompts along with the problem statement, the context window size (`num_ctx`) is explicitly set to 32.768 tokens. We chose a timeout of 600 seconds and set the number of hints injected in the prompt to 2.

5.3. Interpretation of results

While our proposed neuro-symbolic architecture demonstrates exceptional accuracy on complex datasets, an in-depth analysis of the benchmark results (Figure 3) reveals critical insights regarding the operational trade-offs of edge-based deterministic execution. The proposed augmented pipeline successfully solved 20 problems (66.6% accuracy), drastically outperforming the native 8B baseline (0 solved) and demonstrating superior efficiency even against a larger 14B parameter model (7 solved, 23.3% accuracy)

Execution latency and sample size constraints. A prominent operational trade-off in our system is the inference latency, precluding its use in strict real-time applications. Solving a single problem requires an average of 225.7 seconds for MATH500 [6] and up to 301.5 seconds for AIME 2025 [7]. This high execution time is a direct consequence of the multi-stage pipeline running on resource-constrained hardware: semantic retrieval (RAG), sequential code generation, deterministic SymPy execution, and potential cloud-fallback routing. Due to these temporal constraints inherent to the local RTX 5070 Ti deployment, our evaluation was conducted on initial subsets of the benchmark datasets (e.g., the first 100 problems for MATH500 [6], and 21 for GSM8K [4]). Consequently, we frame these results as

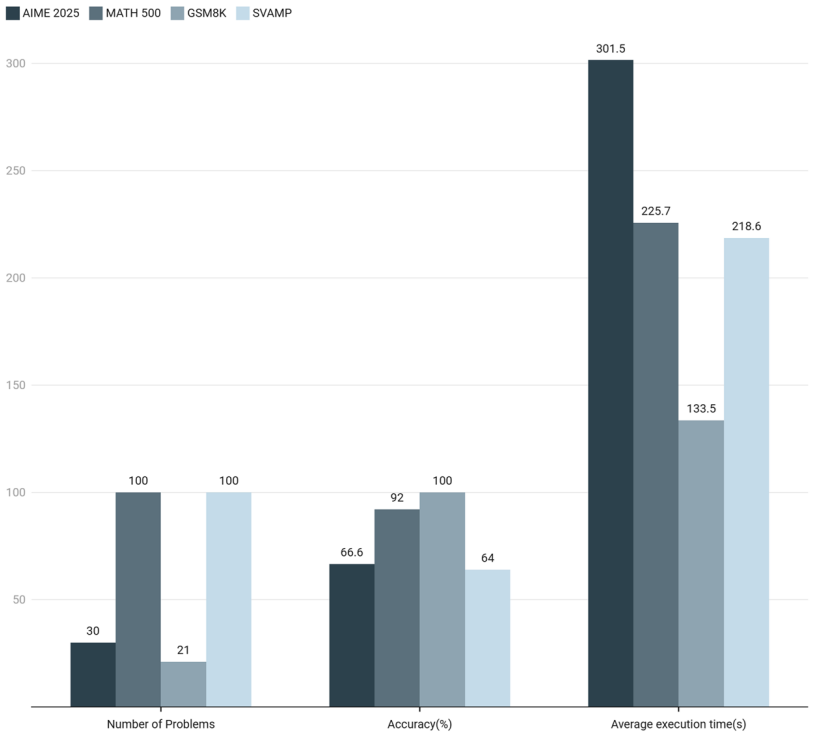

Fig. 3. **Performance evaluation across mathematical reasoning benchmarks.** The grouped bar chart delineates the system's performance on the AIME 2025, MATH 500 [6], GSM8K [4], and SVAMP [5] datasets. The chart details the number of problems tested, the accuracy achieved (%), and the average execution time (s) per problem, highlighting the system's efficiency and precision.

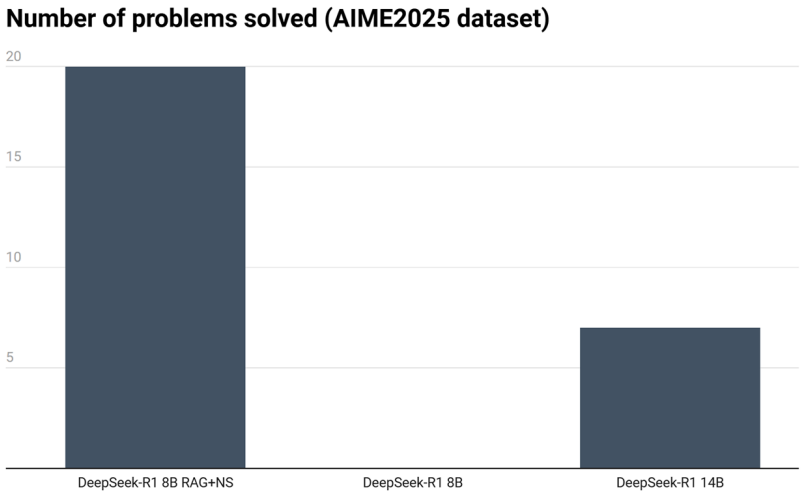

Fig. 4. **Ablation Study and Baseline Comparison on the AIME 2025 benchmark (N = 30)**. We present the system's comparative performance on the AIME 2025 benchmark. The bar chart details the number of problems successfully solved by the proposed DeepSeek-R1 8B RAG+NS alongside the DeepSeek-R1 8B and 14B models, highlighting the effectiveness and improved reasoning of the integrated approach.

a preliminary proof-of-concept. The framework deliberately sacrifices real-time generation speed in favor of rigorous mathematical verifiability, demonstrating that local models can achieve high accuracy in asynchronous or offline high-stakes scenarios, even if large-scale, real-time edge processing remains challenging.

Baseline constraints and the necessity of structural augmentation. To quantify the isolated impact of our proposed architecture, we evaluated the un-augmented baseline models on the AIME 2025 [7] dataset. Without the

Retrieval-Augmented Generation (RAG) module and the deterministic SymPy execution layer, the raw 8-billion parameter model completely failed the benchmark, scoring 0 out of 30 problems (0% accuracy), as presented in Figure 4. Scaling the underlying architecture to 14 billion parameters yielded only a marginal improvement, resolving just 7 out of 30 problems (23.3% accuracy). These baseline results underscore a severe parametric limitation: when forced to rely exclusively on their internal weights, compact edge models inevitably succumb to compounding arithmetic hallucinations. The low performance of the 14B model further confirms that simply increasing parameter count is an inefficient strategy for edge deployments tackling complex mathematical reasoning tasks. Conversely, by wrapping the 0%-performing 8B baseline in our neuro-symbolic pipeline, system performance surges to 66.6%. This stark contrast definitively proves that for edge-scale inference, structural augmentation specifically offloading calculations to a symbolic engine and injecting exact mathematical theorems via RAG is strictly required to unlock competition-level deductive capabilities.

The SVAMP anomaly and semantic vulnerabilities. While the DeepSeek-R1 architecture demonstrates superior reasoning capabilities on verifiable datasets like MATH 500 [6] (92% accuracy) [12], a compelling finding in our evaluation is the stark contrast with its degraded performance on SVAMP [5] (64% accuracy). Although MATH 500 [6] contains highly complex, competition-level mathematics, the problem statements are generally direct and formalized. Conversely, SVAMP [5] consists of elementary-level arithmetic problems that are intentionally injected with deceptive linguistic variations and semantic traps. The lower accuracy on SVAMP [5] highlights a fundamental limitation of the edge orchestrator: while the DeepSeek-R1 8B model successfully avoids *computational* hallucinations by offloading math to SymPy, it remains susceptible to *semantic* hallucinations. If the edge model misinterprets the linguistic phrasing of the problem, it generates a syntactically valid Python script for the flawed logic. The symbolic engine executes this flawed logic perfectly, resulting in an incorrect final answer. This proves that while our pipeline completely eliminates arithmetic errors at the execution layer, the system's overall accuracy remains bounded by the orchestrator's natural language understanding (NLU) capabilities.

Effectiveness of the self-healing fallback. The system achieves a robust 66.6% accuracy on the AIME 2025 dataset, which is historically difficult for models of this parameter class. This success heavily validates the Adaptive Self-Healing Router. In instances where the extreme deductive complexity of an AIME problem caused the edge orchestrator to generate uncompilable code or structurally invalid logic, the system successfully recognized the ϵ error state and autonomously routed the query to the Gemini 2.5 Flash cloud baseline. This edge-to-cloud symbiosis ensures that the system retains high accuracy even when local logic formulation fails.

6. Conclusions and Future Work

In this paper, we introduced a multi-agent distributed neuro-symbolic framework designed to eliminate arithmetic hallucinations in edge-scale Large Language Models. By explicitly decoupling logical orchestration from computational execution utilizing a semantic RAG module for domain knowledge injection and offloading mathematical operations to a deterministic symbolic engine (SymPy), our system consistently ensures high computational accuracy. Furthermore, the integration of an adaptive self-healing cloud fallback mechanism ensures robust fault tolerance against complex deductive tasks that exceed local edge capabilities.

While the experimental results validate the elimination of execution-level arithmetic hallucinations, they simultaneously highlight a critical architectural vulnerability inherent to Program-of-Thought (PoT) paradigms [13]. As evidenced by the performance degradation on the SVAMP [5] dataset, the absolute determinism of the SymPy engine is ultimately bottlenecked by the edge model's Natural Language Understanding (NLU) capacity. Recent architectural analyses have formalized this structural dichotomy between an LLM's surface textual comprehension and its underlying compositional competence in symbolic reasoning [14]. By forcing the LLM to translate ambiguous natural language into strict executable code, the semantic parsing layer becomes an unintended bottleneck. Natural language is inherently fluid, context-dependent, and heavily reliant on implicit assumptions. Symbolic execution environments such as Python and SymPy demand absolute syntactic rigidity, explicit variable declarations, and unambiguous operational hierarchies. In this framework, the edge LLM is essentially tasked with acting as a deterministic compiler for non-deterministic human language. In the presence of the ambiguous phrasing and complex logic found in the SVAMP [5] dataset, the LLM's heuristic pattern-matching often falters. It confidently maps a flawed semantic interpretation to syntactically flawless code, for instance, misinterpreting relational modifiers like "fewer than" or "shared

equally," which leads to the generation of an entirely incorrect computational graph. Consequently, the overall reliability of the neuro-symbolic pipeline is no longer constrained by arithmetic computation, but is instead artificially capped by the model's zero-shot Natural Language Understanding (NLU) threshold [15].

This dynamic shares theoretical parallels with recent developments in Concept Bottleneck Large Language Models (CB-LLMs) [16], which demonstrate that the accurate extraction and alignment of intermediate concepts dictate the success of downstream outputs. In our framework, when confronted with intentionally deceptive linguistic variations, the edge orchestrator acts as a flawed concept extractor generating syntactically valid code for a fundamentally incorrect semantic interpretation. Consequently, the symbolic engine flawlessly executes a flawed premise.

This reveals that while neuro-symbolic execution cures the LLM of arithmetic hallucinations, it inadvertently shifts the point of failure entirely to the semantic interpretation layer. Resolving this translation gap remains a primary directive for future research. A promising avenue involves implementing a pre-execution *self-reflection mechanism*, wherein the logic orchestrator semantically cross-verifies its extracted concepts and generated code against the original natural language constraints before triggering the symbolic engine. Additionally, exploring dynamic prompt adaptation based on detected linguistic complexity could further fortify the edge model against semantic traps. Beyond resolving semantic vulnerabilities, our immediate future work will focus on optimizing the pipeline's overall efficiency and precision. To reduce execution latency, we plan to explore architectural accelerations such as semantic caching and speculative decoding. To further boost precision, we intend to implement traceback-guided iterative refinement, allowing the edge orchestrator to autonomously learn from Python execution errors and correct its code prior to triggering a cloud fallback. Finally, to rigorously evaluate the limits of our adaptive architecture under extreme deductive load, we plan to expand our benchmarking to the recently open-sourced MIT Olympiad-level mathematics dataset [17], advancing the deployment of truly verifiable and robust AI in resource-constrained environments.

References

[1] J. Long, Large language model guided tree-of-thought, 2023. URL: https://arxiv.org/abs/2305.08291. arXiv:2305.08291.

[2] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language models can teach themselves to use tools, 2023. URL: https://arxiv.org/abs/2302.04761. arXiv:2302.04761.

[3] L. Chen, M. A. Zaharia, J. Y. Zou, Frugalgpt: How to use large language models while reducing cost and improving performance, ArXiv abs/2305.05176 (2023). URL: https://api.semanticscholar.org/CorpusID:258564349.

[4] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, J. Schulman, Training verifiers to solve math word problems, 2021. URL: https://arxiv.org/abs/2110.14168. arXiv:2110.14168.

[5] A. Patel, S. Bhattamishra, N. Goyal, Are nlp models really able to solve simple math word problems?, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 2080–2094.

[6] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, J. Steinhardt, Measuring mathematical problem solving with the math dataset, 2021. URL: https://arxiv.org/abs/2103.03874. arXiv:2103.03874.

[7] Z. Shao, P. Wang, Q. Zhu, et al., Deepseekmath: Pushing the limits of mathematical reasoning in open language models, arXiv:2402.03300 (2024).

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Advances in Neural Information Processing Systems (NeurIPS), volume 35, 2022, pp. 24824–24837.

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: NeurIPS, volume 33, 2020, pp. 9459–9474.

[10] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, I. Mordatch, Improving factuality and reasoning in language models through multiagent debate, 2023. URL: https://arxiv.org/abs/2305.14325. arXiv:2305.14325.

[11] C.-H. Kevorchian, Sqan: Simplicial q-attention on the bert hypersphere — a neuro-symbolic architecture with angular logic tensor grounding, 2026. URL: https://doi.org/10.5281/zenodo.19696942. doi:10.5281/zenodo.19696942.

[12] D. Guo, et al., Deepseek-r1 incentivizes reasoning in llms through reinforcement learning, Nature 645 (2025) 633–638. URL: http://dx.doi.org/10.1038/s41586-025-09422-z. doi:10.1038/s41586-025-09422-z.

[13] Z. Bi, N. Zhang, Y. Jiang, S. Deng, G. Zheng, H. Chen, When do program-of-thoughts work for reasoning?, 2023. URL: https://arxiv.org/abs/2308.15452. arXiv:2308.15452.

[14] Z. Zhang, Comprehension without competence: Architectural limits of llms in symbolic computation and reasoning, 2025. URL: https://arxiv.org/abs/2507.10624. arXiv:2507.10624.

[15] K. Mahowald, A. A. Ivanova, I. A. Blank, N. Kanwisher, J. B. Tenenbaum, E. Fedorenko, Dissociating language and thought in large language models, 2024. URL: https://arxiv.org/abs/2301.06627. arXiv:2301.06627.

[16] C.-E. Sun, T. Oikarinen, B. Ustun, T.-W. Weng, Concept bottleneck large language models, 2025. URL: https://arxiv.org/abs/2412.07992. arXiv:2412.07992.

[17] S. Alshammari, K. Wen, A. Zainal, M. Hamilton, N. Safaei, S. Albarakati, W. T. Freeman, A. Torralba, Mathnet: a global multimodal benchmark for mathematical reasoning and retrieval, 2026. URL: https://arxiv.org/abs/2604.18584. arXiv:2604.18584.