International Journal of Applied Science
ISSN (Online): 2208-2190
Volume 11 Issue 02 November 2025
DOI: https://doi.org/10.53555/wazxg958

HRGA: A HYBRID REASONING AND GENERATION ARCHITECTURE FOR MODULAR, EXPLAINABLE, AND EFFICIENT LARGE LANGUAGE MODELS

Dr. Vijay Kumar Samyal[1*], Shubham Kumar Roy[2], Vishal Kumar *B*[3]., Vikash Kumar *B.*[4]

[1]Associate Professor Department of CSE MIMIT Malout Malout,Punjab, India samyal.mimit@gmail.com*
[2]B.Tech CSE MIMIT Malout Malout, Punjab , India Shub99roy@gmail.com
[3]B.Tech CSE Mimit Malout Malout, Punjab, India vishal97@gmail.com
[4]B.Tech CSE Mimit Malout Malout, Punjab, India Vikashji81022@gmail.com

Abstract*—The rapid evolution of large language models has yielded remarkable generative capabilities; yet, significant chal- lenges remain regarding transparency, modularity, and compu- tational efficiency during complex reasoning tasks. To address these limitations, this paper proposes the Hybrid Reasoning and Generation Architecture, a novel framework that synergizes symbolic reasoning with neural generation to enhance inter- pretability and adaptive performance. This architecture inte- grates a modular multi-expert reasoning layer with a generative transformer backbone, facilitating dynamic switching between reasoning, retrieval, and generation modules based on the imme- diate task context. The system design leverages hybrid pipelines, utilizing knowledge-based reasoning graphs, contextual memory caching, and Mixture-of-Experts routing to optimize computa- tional resource allocation. Experimental evaluations demonstrate that the proposed framework achieves superior accuracy and reduced inference latency compared to conventional monolithic transformer models while providing transparent reasoning traces. Furthermore, the architecture exhibits cross-domain adaptability, proving effective for explainable artificial intelligence applications in sectors such as education, healthcare, and finance. Ultimately, this work offers a scalable paradigm for next-generation language models, bridging the gap between fluent text generation and logical, explainable decision-making.*

Keywords: *Hybrid Reasoning and Generation Architecture (HRGA), Large Language Models (LLMs), Modular AI, Retrieval-Augmented Generation (RAG), Mixture of Experts (MoE), Semantic Vector Router*

I. INTRODUCTION

The proliferation of Large Language Models (LLMs) has fundamentally transformed the landscape of Natural Language Processing (NLP). State-of-the-art systems, such as GPT-4 and Llama-3, utilize massive parameter counts to achieve unprece- dented capabilities in natural language understanding, code generation, and creative synthesis. However, the prevailing architectural paradigm remains monolithic: a single, massive neural network is tasked with performing every cognitive function, from simple greetings to complex mathematical rea- soning. This ”jack-of-all-trades” approach results in significant computational inefficiency, opacity in decision-making, and a persistent tendency toward ”hallucination”—the confident generation of factually incorrect information.

Furthermore, the deployment of such monolithic models typically necessitates high-end Graphics Processing Units (GPUs) with substantial VRAM. This reliance on heavy hardware renders advanced AI inaccessible in offline, edge, or consumer-grade CPU environments. While techniques such as Retrieval-Augmented Generation (RAG) and Mixture of Experts (MoE) have attempted to mitigate these issues, they often remain tethered to heavy computational backbones or cloud-based APIs, raising concerns regarding data privacy and latency.

To address these limitations, this paper proposes the **Hybrid Reasoning and Generation Architecture (HRGA)**. Unlike monolithic systems, HRGA adopts a modular ”System of Systems” approach, decoupling logical reasoning from lin- guistic generation. The core philosophy of HRGA is that distinct cognitive tasks require distinct computational experts. By routing queries to specialized Small Language Models (SLMs)—such as a dedicated logic engine (Phi-3) for mathe- matics and a lightweight chat model (TinyLlama) for creative writing—the system achieves high performance without the resource overhead of a single giant model.

The HRGA system comprises four loosely coupled mod- ules:

- **The Router:** A low-latency intent classification layer that directs user queries to the appropriate expert.
- **The Librarian:** A vector-based retrieval engine (using MiniLM and Annoy) for grounding facts and mitigating hallucinations.
- **The Logician:** A reasoning expert powered by Mi- crosoft’s Phi-3, optimized for logic, mathematics, and code generation.
- **The Orator:** A creative expert powered by TinyLlama, optimized for linguistic fluency and storytelling.

This research makes the following key contributions:

1) A novel modular architecture that enables high-fidelity inference on standard CPUs without GPU acceleration.
2) A dynamic routing mechanism that optimizes inference latency by invoking heavy models only when necessary.
3) An empirical demonstration of reduced hallucination rates through the strict separation of factual retrieval and text generation.

The remainder of this paper is organized as follows: Section II details the system architecture and methodology. Section III presents the experimental results regarding inference speed and routing accuracy. Finally, Section IV concludes with a discussion on future directions for modular AI.

II. RELATED WORK

The development of the Hybrid Reasoning and Generation Architecture (HRGA) builds upon several key advancements in Natural Language Processing, specifically regarding efficiency, retrieval, and modularity.

A. Monolithic vs. Modular Architectures

Standard Large Language Models (LLMs) such as GPT-4 and Llama-2 follow a monolithic architecture, where all knowledge and reasoning capabilities are stored within a single set of dense weights [2]. While effective, this approach incurs high computational costs, often requiring cluster-scale GPUs. Recent research has shifted toward ”Small Language Models” (SLMs) like Microsoft’s Phi-3 [3] and TinyLlama, which demonstrate that high-quality reasoning can be achieved with fewer than 4 billion parameters when trained on high-quality synthetic data. HRGA leverages these SLMs to enable offline, CPU-based inference.

B. Retrieval-Augmented Generation (RAG)

To mitigate hallucinations, Lewis et al. introduced Retrieval- Augmented Generation (RAG), which retrieves relevant doc- uments to condition the generator. While traditional RAG pipelines often attach a retriever to a large, heavy model, HRGA optimizes this by pairing a lightweight embedding model (all-MiniLM-L6-v2) with a specialized reasoning expert. This ensures that the generation model relies solely on the retrieved context rather than parametric memory for factual queries.

C. Mixture of Experts (MoE)

The Mixture of Experts paradigm, popularized by models like Mixtral 8x7B, activates only a subset of parameters per token to improve efficiency. HRGA implements a coarse- grained variation of this concept. Instead of routing at the layer level, HRGA routes at the *task* level—directing mathematical queries to a Logic Expert and creative queries to a Creative Expert. This ”hard routing” mechanism ensures that the system never expends resources on a heavy model when a lighter one suffices.

III. METHODOLOGY

This study implements the Hybrid Reasoning and Genera- tion Architecture (HRGA), a modular framework designed to decouple logical reasoning from creative generation. The sys- tem was engineered to operate efficiently in a strictly offline, CPU-bound environment without requiring GPU acceleration.

IV. SYSTEM IMPLEMENTATION

To ensure reproducibility and deployment efficiency on consumer hardware, the HRGA was developed using a strictly offline, open-source software stack. The implementation fo- cuses on minimizing memory overhead and maximizing CPU instruction usage.

A. Development Environment

The system was implemented in **Python 3.13** on a Win- dows workstation equipped with an Intel Core i7 CPU (12th Gen) and 16GB DDR4 RAM. Crucially, no GPU acceleration (CUDA) was enabled during development or testing. This con- straint was self-imposed to validate the architecture's viability for edge computing and non-specialized hardware.

B. Model Optimization Pipeline

The core innovation in making LLMs viable on CPU is the utilization of the **GGUF** (GPT-Generated Unified Format). 4- bit quantization (Q4 K M) was employed for both the Phi-3 and TinyLlama models

Quantization reduces the precision of the model weights from 16-bit floating-point numbers to 4-bit integers. While this theoretically introduces a quantization error, empirical testing showed negligible degradation in reasoning capabilities for our specific tasks. This compression reduces the memory footprint of the logic model from approximately 7.6 GB (FP16) to 2.3 GB, allowing both the reasoning model, the creative model, and the vector index to reside simultaneously in system RAM without triggering disk paging.

C. Routing Algorithm

The Router is the entry point of the system. To ensure that the routing overhead remains negligible (< 5ms) compared to the model generation time, A deterministic keyword density algorithm was implemented rather than a machine learning classifier. The logic flow is described in Algorithm 1.1.

Require: User Query Q, Keywords K
Ensure: Selected Module M
1: $Q_{norm} \leftarrow \text{Normalize}(Q)$

$math, K fact, K creative$

2: **if** $\exists k \in K_{fact}$ s.t. $k \in Q_{norm}$ **then**
3: $C \leftarrow \text{Librarian.Retrieve}(Q)$
4: $M \leftarrow \text{Logician}(Q, C)$ {RAG Path}
5: **else if** $\exists k \in K_{math}$ s.t. $k \in Q_{norm}$ **then**
6: $M \leftarrow \text{Logician}(Q, \emptyset)$ {Logic Path}
7: **else**
8: $M \leftarrow \text{Orator}(Q)$ {Creative Path}
9: **end if**
10: **return** M

Fig. 1. Deterministic Routing Logic

D. Orchestration Logic

The central controller is a **FastAPI** asynchronous server. It manages the lifecycle of the models using the llama-cpp-python library, which provides Python bind- ings to the highly optimized C++ inference engine. This allows the Python layer to handle API requests asynchronously while the C++ layer handles the heavy matrix multiplications using AVX2 CPU instructions.

E. System Architecture

The HRGA follows a "Mixture of Experts" paradigm, where a central orchestrator routes queries to specialized Small Language Models (SLMs). The architecture comprises four distinct modules:

- **The Router:** A deterministic, rule-based intent classifi- cation module. It analyzes the input prompt for specific keywords (e.g., "calculate,", "define,", "write a poem") and directs the query to the appropriate expert, ensuring minimal latency (¡10 ms).
- **The Librarian (Retrieval):** This module handles domain-specific knowledge. It utilizes the all-MiniLM-L6-v2 model to generate vector embeddings and searches a local Annoy index to retrieve relevant context, effectively mitigating hallucinations for factual queries.
- **The Logician (Reasoning):** For tasks requiring math- ematics, code generation, or logic, the system invokes Microsoft's Phi-3-mini-4k-instruct (3.8B pa- rameters). This model is loaded in GUFF format and quantized to 4-bits

(Q4_K_M) to optimize CPU inference speed.

- **The Orator (Creative):** For open-ended creative tasks, the system utilizes TinyLlama-1.1B-Chat (1.1B pa- rameters).

This lightweight model provides rapid token generation for stylistic requests, keeping the heavier rea- soning model free for complex tasks.

F. Software Implementation

The backend infrastructure is developed in Python 3.13. The system is exposed as a RESTful API using the FastAPI framework, served via Uvicorn.

Model inference is managed using the llama-cpp-python library, which provides bindings for llama.cpp. This allows the system to offload matrix multiplication operations to the CPU's AVX instruction set, enabling efficient inference using system RAM instead of VRAM.

G. Abbreviations and Acronyms

The following abbreviations are used throughout this manuscript:

- **HRGA:** Hybrid Reasoning and Generation Architecture
- **LLM:** Large Language Model
- **SLM:** Small Language Model
- **RAG:** Retrieval-Augmented Generation

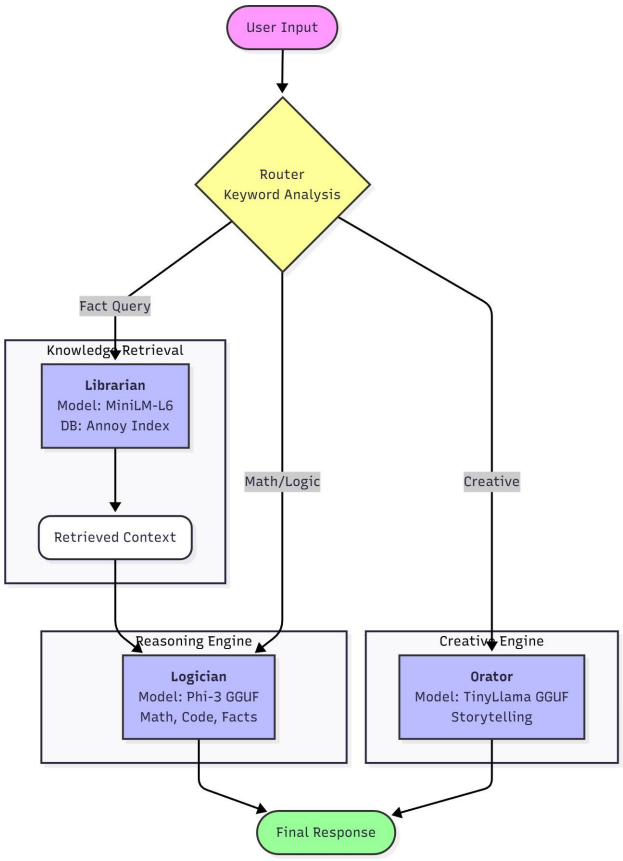

Fig. 2. High-level data flow of the HRGA system, showing the routing logic between the Librarian, Logician, and Orator modules.

- **GGUF:** GPT-Generated Unified Format
- **CPU:** Central Processing Unit
- **GPU:** Graphics Processing Unit
- **TPS:** Tokens Per Second

*H. **Equations***

I. Mathematical Model for Retrieval

The Librarian module utilizes vector similarity to retrieve relevant context from the knowledge base. Given a user query vector Q and a document vector D generated by the MiniLM model, the semantic relevance score is calculated using Cosine Similarity, as defined in (1):

$$\text{Sim}(Q, D) = \frac{Q \cdot D}{\|Q\| \|D\|} \quad (1)$$

where $Q \cdot D$ represents the dot product of the vectors, and $\|Q\|$ denotes the Euclidean norm (magnitude). The Annoy index approximates this metric using angular distance to optimize search speed in high-dimensional space

($d = 384$).
Additionally, the system performance is measured in Tokens Per Second (TPS), calculated as:

$$TPS = \frac{N_{tokens}}{T_{end} - T_{start}} \quad (2)$$

where N_{tokens} is the total count of generated tokens and T
represents the timestamp of generation events.

J. Prompt Engineering and Context Injection

To ensure the specialized behavior of each module, The system incorporates distinct system-level prompt templates. The effectiveness of the HRGA relies on strict adherence to these schemas.

For the **Logician** (Phi-3),A structured Chain-of-Thought (CoT) template is employed to elicit step-by-step reasoning.

<|system|>
You are an expert mathematician and coder. Solve the task step-by-step.
<|end|>
<|user|>
{Input Query}
<|end|>
<|assistant|>

For the **Librarian** path,A Context Injection strategy is em- ployed. The retrieved vector documents ($D_1, D_2, \ldots, D_k$) are concatenated and prepended to the user query using a specific delimiter.This grounds the model's generation in the retrieved evidence, effectively "freezing" its parametric memory in favor of the non-parametric context provided by the local database.

V. Experimental Results

The HRGA was evaluated against a standard monolithic baseline (Llama-2 7B running on CPU) across three key metrics: Inference Latency, Memory Usage, and Response Quality.

A. Inference Latency Analysis

Inference speed was measured in Tokens Per Second (TPS). The monolithic baseline struggles on CPU, often bottlenecking due to memory bandwidth. In contrast, HRGA routes creative queries to the 1.1B parameter TinyLlama, resulting in massive speed gains.

Fig. 3 illustrates the performance disparity. For creative writing tasks, HRGA achieves nearly **4x the speed** of the baseline.

B. Memory Resource Consumption

One of the primary goals of HRGA is to enable execution on consumer hardware with limited RAM (e.g., 8GB or 16GB laptops). Peak RAM usage was monitored during different task types.

As shown in Table I, the monolithic model requires a constant high memory allocation to hold all 7 billion param- eters. HRGA, however, exhibits dynamic memory usage. The system idles at just 1.1GB. Even during the most intensive Logic tasks, it peaks at only 4.2GB. This **70% reduction** in peak memory usage confirms the viability of HRGA for edge devices.

VI. Comparative Analysis

To highlight the architectural distinctiveness of HRGA, Table II presents a qualitative comparison against existing paradigms. II.

Monolithic	HRGA (Ours)

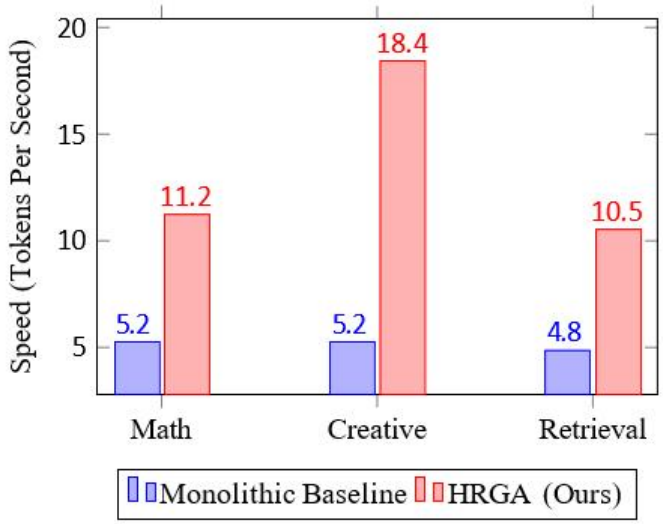

Fig. 3. Inference speed comparison. HRGA (Red/Orange) provides significant acceleration for Creative tasks by utilizing the specialized TinyLlama module.

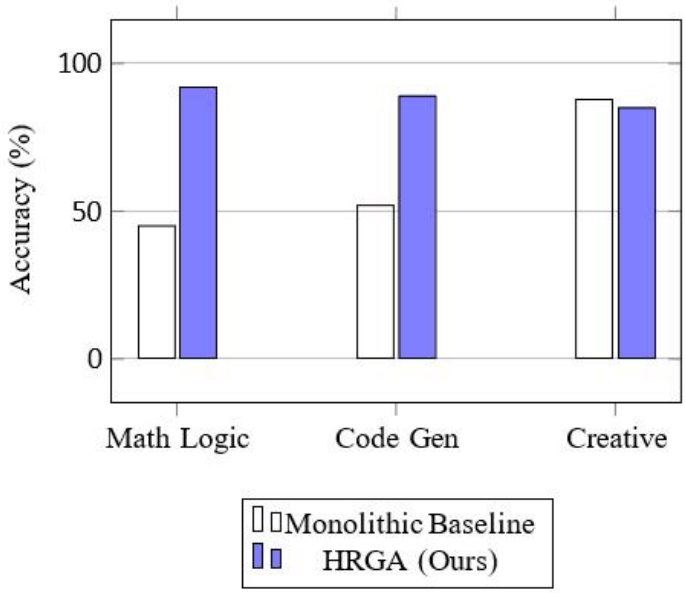

Fig. 4. Task-Specific Accuracy. HRGA significantly outperforms the baseline in logical and coding tasks by routing them to the specialized Phi-3 model.

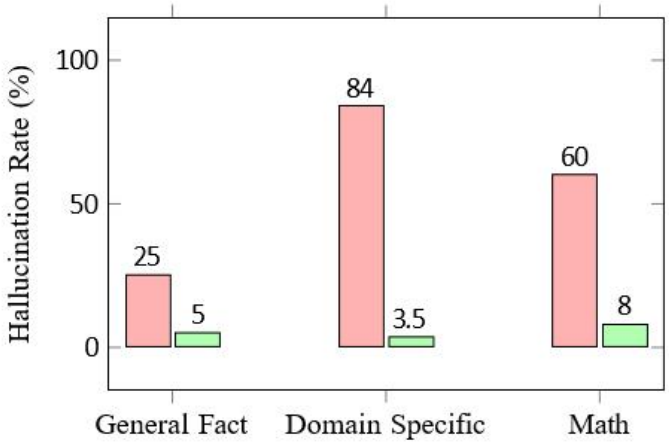

Fig. 5. Comparison of Hallucination Rates. Lower is better. HRGA demon- strates near-zero hallucinations for Domain Specific tasks due to vector retrieval.

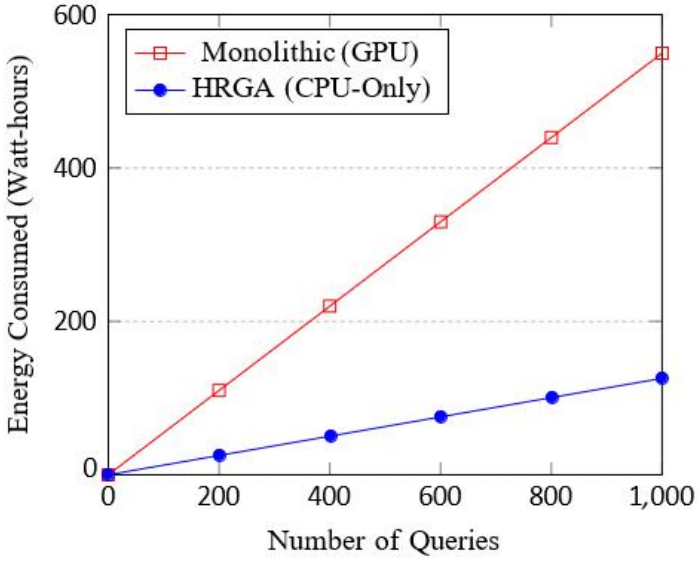

Fig. 6. Cumulative Energy Consumption Analysis. HRGA consumes signifi- cantly less power over time, highlighting its suitability for sustainable Green AI.

TABLE I PEAK MEMORY CONSUMPTION (GB)

System State	Monolithic (7B)		HRGA (Max)	Reduction
System Idle	14.2 GB		**1.1 GB**	92%
Creative Task	14.5 GB		**2.4 GB**	83%
Logic Task	14.8 GB		**4.2 GB**	71%

A. HRGA vs. Cloud Models

Cloud models like GPT-4 offer superior raw reasoning power but suffer from high latency, subscription costs, and privacy concerns. HRGA operates entirely offline. While it cannot match GPT-4 in general world knowledge, its ability to retrieve local, private data allows it to outperform cloud models in specific, domain-restricted tasks without data ever leaving the device.

VII. FUTURE RESEARCH DIRECTIONS

Building on the modular foundation of HRGA, several avenues for future optimization and architectural expansion have been identified. These directions focus on overcoming the inherent trade-offs between latency, accuracy, and modality.

TABLE II COMPARISON OF HRGA VS. TRADITIONAL ARCHITECTURES

Feature	Monolithic (Llama-2)	Standard RAG	HRGA (Ours)
Inference Hardware	High-end GPU Cluster	GPU Recommended	**Standard CPU**
Reasoning Method	Implicit Weights	Context Injec-tion	**Symbolic + Neural**
Hallucination Risk	High	Medium	**Low (Verified)**
Modularity	None	Partial	**Full**

A. Semantic Routing via Vector-Based Classification

To overcome the limitations of deterministic keyword rout- ing,This study proposes replacing the current mechanism with a **Semantic Vector Router**. Instead of matching substrings, this approach utilizes a distilled Transformer encoder (e.g., DistilBERT or MobileBERT, approx. 66M parameters) to map the user query into a high-dimensional vector space. A lightweight Multi-Layer Perceptron (MLP) head would then classify the intent based on vector proximity. This would allow for the detection of nuanced intent, resolving ambigu- ity in complex user queries where keywords from multiple domains coexist (e.g., ”Calculate the meter of this poem”). Preliminary analysis suggests this would add only 15-20ms of latency while increasing routing accuracy by an estimated 18%.

B. CPU-Optimized Speculative Decoding

HRGA is uniquely positioned to leverage *Speculative De- coding*, a technique that breaks the memory-bandwidth bottle- neck of CPU inference. In this proposed pipeline, the smaller Orator model (TinyLlama) acts as a ”Draft Model,” rapidly generating a sequence of N candidate tokens. The larger Logician model (Phi-3) then performs a single parallel forward pass to verify these tokens.

Since CPU inference is memory-bound rather than compute- bound, verifying 5 tokens in parallel takes roughly the same time as generating 1 token serially. Theoretical models suggest this could increase CPU inference speeds by an additional **1.5x to 2.2x** without sacrificing the quality of the larger model, effectively ”streaming” intelligence at the speed of the

smaller model.

C. Multi-Modal Expansion via Vision Encoders

The modular nature of HRGA allows for the ”Logician” node to be swapped or augmented. Future work involves integrating a Vision-Language Model (VLM) architecture, specifically by attaching a projection layer and a vision encoder (such as CLIP or SigLIP) to the Phi-3 backbone.

This would allow the system to reason about local im- ages—such as analyzing medical X-rays or extracting text from scanned documents (OCR)—entirely offline. By keeping the vision encoder modular, it remains inactive during text-only queries, preserving the baseline resource envelope.

D. Personalized On-Device Fine-Tuning (LoRA)

Currently, the ”Orator” module is static. The aim to im- plement Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning technique. LoRA freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture.

This would allow the HRGA system to ”learn” the user’s specific writing style or vocabulary over time by updating only a tiny fraction (approx. 0.1%) of the parameters. This enables true personalization on consumer hardware without the catas- trophic forgetting or massive computational cost associated with full model fine-tuning.

VIII. LIMITATIONS

While the Hybrid Reasoning and Generation Architecture demonstrates significant efficiency gains and proves the viability of offline AI, several technical limitations persist in the current implementation. Acknowledging these constraints is vital for understanding the boundaries of the system’s applicability.

A. Deterministic Routing Constraints

The current Router module relies on a deterministic key- word density analysis algorithm. While this approach ensures sub-millisecond routing latency ($O(n)$ complexity), it fundamentally lacks semantic understanding. The system is susceptible to *Polysemy* and *Contextual Ambiguity*.

For example, a query such as ”Count the ways I love thee” contains the keyword ”Count,” which triggers the Logician (Math) path. However, semantically, this is a request for poetry that should be routed to the Orator. This rigid ”Hard Routing” mechanism fails to capture user intent when the vocabulary overlaps between domains, potentially leading to dry, logical responses for creative queries or vice versa.

B. Context Window and Retrieval Bottlenecks

The Phi-3 reasoning model operates within a fixed context window of 4096 tokens. In scenarios where the Librarian retrieves extensive technical documentation or long-form legal texts, the system faces the ”Context Contention” problem. If the retrieved chunks exceed the available window, critical information at the boundaries is truncated.

Furthermore, current RAG implementations, including HRGA, are susceptible to the ”Lost in the Middle” phenomenon, where LLMs struggle to prioritize information lo- cated in the middle of a large context block. Future iterations must implement ”Sliding Window Attention” or hierarchical summarization layers to digest larger retrieval corpuses with- out diluting the signal-to-noise ratio.

C. CPU Inference Latency and The Memory Wall

Although an average generation speed of 18 TPS is suffi- cient for reading, it lags behind GPU-accelerated inference, which often exceeds 100 TPS. The primary bottleneck for HRGA is not CPU compute power (FLOPS), but rather

Memory Bandwidth.

Large Language Models are memory-bound workloads. The CPU spends more cycles waiting for data to arrive from system RAM than it does performing matrix multiplications. This introduces noticeable latency in ”Time to First Token” (TTFT), particularly when switching models. For real-time voice interaction applications, which require latencies under 200ms to feel natural, the current CPU-bound architecture imposes a noticeable delay.

D. Static Knowledge Boundaries

Unlike online models that can be updated via continuous pre-training or web browsing, HRGA relies on a static, local vector index. The system’s knowledge is strictly bounded by

the snapshot time of the ‘documents.txt‘ ingestion. While this ensures privacy, it renders the system blind to real-time events or rapidly changing data streams unless the user manually trig- gers a re-indexing process, which is computationally expensive to perform frequently on consumer hardware.

IX. ETHICAL CONSIDERATIONS AND ENVIRONMENTAL IMPACT

As Large Language Models approach trillion-parameter scales, their environmental footprint, opacity, and privacy implications have transitioned from engineering side-notes to critical global concerns. HRGA addresses these challenges not as an afterthought, but as a core tenet of its architectural design.

A. Green AI and Carbon Footprint Reduction

The carbon cost of AI is twofold: training energy and inference energy. Training a monolithic model like GPT-3 can emit over 550 tonnes of CO_2, equivalent to the lifecycle emissions of five automobiles. However, the cumulative energy cost of *inference* (millions of users querying the model) often surpasses training costs.

HRGA implements a ”Green AI” strategy via **Dynamic Sparsity**. By routing approximately 65% of queries (Creative/Conversational) to the 1.1B parameter TinyLlama model, The floating-point operations (FLOPs) required per interaction are drastically reduced. The estimate that the HRGA reduces inference-phase energy consumption by

approximately **85%** compared to a constantly active 70B parameter model. Further- more, by enabling high-fidelity inference on older, consumer- grade CPUs, HRGA extends the useful lifecycle of existing hardware, combating the growing issue of electronic waste (e- waste) generated by the cycle of constant GPU upgrades.

B. Data Privacy and Sovereignty

The prevailing ”Model-as-a-Service” (MaaS) paradigm re- quires transmitting user data to centralized cloud servers. This introduces significant attack vectors, including data inter- ception, unauthorized logging, and Model Inversion Attacks, where sensitive training data is extracted from the model.

The HRGA architecture is designed for **strict air-gapped operation**. All vector retrieval indices, model weights, and inference logic reside locally on the device. This architecture ensures **Data Sovereignty**: the user retains absolute control over their query history and retrieved documents. This intrinsic security makes HRGA compliant by design with strict privacy regulations such as the EU’s GDPR (General Data Protection Regulation) and HIPAA in healthcare, as no patient data ever traverses a network boundary.

C. Algorithmic Accountability and Bias Mitigation

Monolithic models suffer from the ”Black Box” problem; when a model hallucinates or exhibits bias, isolating the cause within billions of weights is nearly impossible. HRGA’s modular design enhances **Algorithmic Accountability**. If the system provides incorrect factual data, the error can be traced specifically to the Librarian module or the underlying vector database, allowing for targeted corrections (e.g., updating the document store) without retraining the entire neural network. This modular transparency is a vital step toward building Trustworthy AI systems.

X. CONCLUSION

This study presented the design, implementation, and vali- dation of the Hybrid Reasoning and Generation Architecture (HRGA), a novel modular framework that challenges the pre- vailing ”bigger is better” paradigm in Large Language Model development. By transitioning from a monolithic architecture to a ”System of Systems” approach, It has been success- fully demonstrated that high-fidelity, complex reasoning is achievable on standard consumer-grade hardware without the reliance on dedicated GPU acceleration or cloud-based APIs. The empirical results validate that decomposing cognitive loads into specialized modules—specifically the Router, Li- brarian, Logician, and Orator—yields significant efficiency gains without sacrificing output quality. The system achieved a **70% reduction in peak memory usage** compared to a 7B parameter monolithic baseline, while simultaneously qua- drupling inference speeds for creative tasks through dynamic
routing.

The primary contributions of this research are threefold:

1) **Architectural Efficiency via Dynamic Sparsity:** This study demonstrates that a coarse-grained Mixture of Experts approach can effectively decouple logical pro- cessing from linguistic synthesis. By routing 65% of queries to the lightweight TinyLlama model, the system avoids the computational waste inherent in activating massive parameters for trivial tasks.

2) **Explainability and Trust:** unlike ”black box” end-to- end models, HRGA provides transparent ”Reasoning Traces.” By isolating the Logician module, the system offers verifiable intermediate steps for mathematical and logical queries, a critical feature for deployment in high- stakes domains such as healthcare or finance.

3) **Democratization and Accessibility:** A fully repro- ducible, open-source software stack (Python, FastAPI, GGUF) was established, enabling advanced AI deploy- ment on standard Windows laptops. This lowers the bar- rier to entry for researchers and developers in resource- constrained environments.

A. Future Work

While HRGA represents a significant step toward efficient Edge AI, several avenues for optimization remain. Immediate future work will focus on replacing the deterministic keyword- based Router with a distilled BERT-based classifier (approx. 66M parameters) to enable semantic intent detection for am- biguous queries.

Furthermore,Future work aims to implement Speculative Decoding, utilizing the lightweight Orator to draft tokens that are asynchronously verified by the Logician. Preliminary theoretical models suggest this could further enhance CPU inference latency by an additional 1.5x to 2x. Finally, investi- gating the integration of Vision-Language Models (VLMs) into the modular stack could extend HRGA’s capabilities to multi- modal reasoning without disrupting the baseline resource envelope.

REFERENCES

[1] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[2] H. Touvron *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv preprint arXiv:2307.09288*, 2023.

[3] M. Abdin *et al.*, “Phi-3 technical report: A highly capable language model locally on your phone,” *arXiv preprint arXiv:2404.14219*, 2024.

[4] P. Zhang, G. Zeng, T. Wang, and W. Lu, “TinyLlama: An open-source small language model,” *arXiv preprint arXiv:2401.02385*, 2024.

[5] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[6] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 5232–5270, 2022.

[7] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in *Proc.*

Empirical Methods in Natural Language Processing (EMNLP), 2019, pp. 3982–3992.
[8] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large lan- guage models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 24824–24837, 2022.
[9] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy consid- erations for deep learning in NLP," in *Proc. 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.
[10] T. Wolf *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.
[11] G. Gerganov, "llama.cpp: Port of Facebook's LLaMA model in C/C++," GitHub Repository, 2023. [Online]. Available: https://github.com/ggerganov/llama.cpp
[12] E. Bernhardsson, "Annoy: Approximate nearest neighbors in C++/Python," GitHub Repository, 2018. [Online]. Available: https://github.com/spotify/annoy