LLMs on a Budget? Say HOLA

Zohaib Hasan Siddiqui[1]*, Jiechao Gao[2]*, Ebad Shabbir[3]*, Mohammad Anas Azeez[1], Rafiq Ali[3], Gautam Siddharth Kashyap[4], Usman Naseem[4]†

[1]Jamia Hamdard, New Delhi, India
[2]Center for SDGC, Stanford University, California, USA
[3]DSEU-Okhla, New Delhi, India
[4]Macquarie University, Sydney, Australia

Abstract

Running Large Language Models (LLMs) on edge devices is constrained by high compute and memory demands—posing a barrier for real-time applications in industries like healthcare, education, and embedded systems. Current solutions such as quantization, pruning, and Retrieval-Augmented Generation (RAG) offer only partial optimizations and often compromise on speed or accuracy. We introduce **HOLA**, an end-to-end optimization framework for efficient LLM deployment. Internally, it leverages Hierarchical Speculative Decoding (HSD) for faster inference without quality loss. Externally, AdaComp-RAG adjusts retrieval complexity based on context needs. Together with Lo-Bi, which blends structured pruning (LoRA) and quantization, HOLA delivers significant gains: +17.6% EMA on GSM8K, +10.5% MCA on ARC, and reduced latency and memory on edge devices like Jetson Nano—proving both scalable and production-ready. Our code is available at: https://github.com/zohaibhasan066/HOLA_Codebase

1 Introduction

Large Language Models (LLMs) have transformed NLP applications—from question answering (Ehtesham et al., 2025) and code generation (Zhong and Wang, 2024) to summarization (Ghosh et al., 2024) and conversational agents (Zenimoto, 2024). However, their deployment on edge and low-resource environments remains constrained by high compute and memory demands (Dutta et al., 2025), limiting real-world impact in domains like healthcare (Lysandrou et al., 2023), education (Bayarri-Planas et al., 2025), and personalized assistants (Tack et al., 2024). While techniques such as quantization (Tack et al., 2024), pruning (Chimoto et al., 2024), and Retrieval-Augmented Generation (RAG) (Parekh et al., 2024) offer partial relief, they often sacrifice accuracy, speed, or generality. Moreover, most approaches optimize isolated components—either internal inference (Liu et al., 2023) or external retrieval (Zheng et al., 2023)—without addressing the full pipeline. We introduce **HOLA** (Hierarchical Optimized Language Augmentation), a unified framework targeting both internal and external efficiency. It integrates (1) Hierarchical Speculative Decoding (HSD) for faster, accurate inference; (2) AdaComp-RAG, an adaptive retrieval system based on context relevance; and (3) Lo-Bi Optimization, combining LoRA-based structured pruning with mixed-precision quantization.

2 Related Works

Deploying LLMs in constrained environments has driven advances in decoding acceleration, retrieval optimization, and model compression. Techniques like Speculative Decoding (Sun et al., 2024) and Medusa (Cai et al., 2024) use draft-and-verify schemes to speed up inference, though they often require extra verifier networks and struggle with long sequences. Adaptive retrieval methods (Labruna et al., 2024) selectively route queries to improve efficiency but introduce added system complexity and retraining needs. Model compression remains central to edge deployment. Quantization approaches like GPTQ (Bumgardner et al., 2025) and Bi-LLM (Weng et al., 2025) reduce precision for memory savings, but aggressive quantization can impact accuracy. Structured pruning using LoRA (Hu et al.) and LoRA-Pruner (Zhang et al., 2024b) compress model weights with minimal retraining. While a few efforts combine pruning and quantization (Ali et al., 2025), most lack holistic, hardware-aware integration across the inference stack.

3 Methodology

Our proposed framework, **HOLA**, is composed of three synergistic modules—HSD accelerates au-

* Equal Contributions.
† Corresponding Author: usman.naseem@mq.edu.au

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1035–1043
November 4-9, 2025

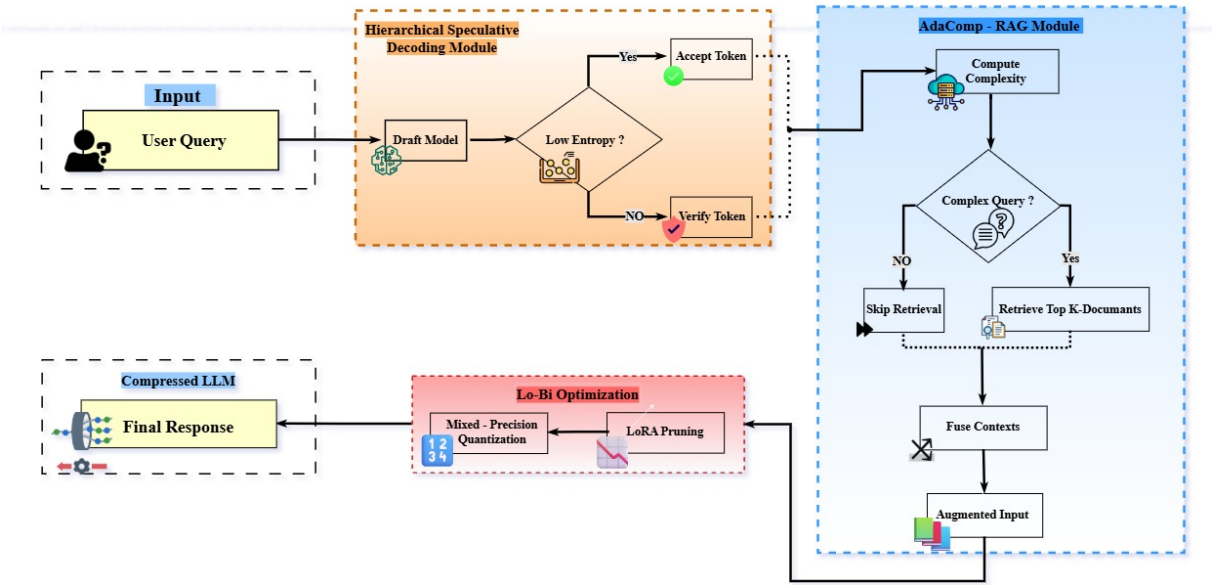


Figure 1: **HOLA** architecture.

toregressive generation via entropy-aware verification, AdaComp-RAG adaptively modulates retrieval granularity, and Lo-Bi Optimization ensures compact and resource-aware model deployment. The broader process of the **HOLA** is illustrated in Algorithm 1. On the other hand, Figure 1 presents the detailed architecture.

Algorithm 1 : HOLA

Require: Input context $\mathbf{x}$
Ensure: Generated output sequence $\mathbf{y}$
Step 1: HSD
Generate draft $\hat{\mathbf{y}} = f_{\text{draft}}(\mathbf{x})$
for $t = 1$ to T **do**
  Compute token entropy $H(\hat{y}_t)$
  if $H(\hat{y}_t) < \tau$ **then** ▷ Fast-path
    $y_t \leftarrow \hat{y}_t$
  else ▷ Fallback to verifier
    $y_t \leftarrow f_{\text{ver}}(\mathbf{x}, y_{<t})$
  end if
end for
Step 2: AdaComp-RAG
Measure query complexity $C(\mathbf{q}) = \|\nabla_{\mathbf{q}} \mathcal{L}\|_2$
if $C(\mathbf{q}) \geq \delta$ **then**
  Retrieve top-k documents and form $\mathbf{x}' = [\mathbf{x}; \mathbf{d}_1, \ldots, \mathbf{d}_k]$
else
  $\mathbf{x}' \leftarrow \mathbf{x}$
end if
Apply compositional attention on $\mathbf{x}'$
Step 3: Lo-Bi Model Optimization
Compute low-rank update: $\Delta \mathbf{W} = \mathbf{AB}$
Update weights: $\mathbf{W}' = \mathbf{W} + \Delta \mathbf{W}$
for each subblock i in $\mathbf{W}'$ **do**
  Select bit-precision $p_i = \arg\min_{p \in \{4,8,16\}} \mathbb{E}[\|f_{\text{orig}} - f_{\text{quant}}^{(p)}\|_2]$
  Quantize subblock using p_i bits
end for
return Optimized output sequence $\mathbf{y}$

3.1 HSD Module

To mitigate the inherent latency of sequential token-by-token generation in autoregressive transformers, we introduce a HSD mechanism. Let the target sequence be $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ conditioned on input context $\mathbf{x}$. Conventional decoding computes $y_t \sim p(y_t \mid \mathbf{x}, y_{<t})$ in series, which becomes computationally expensive for long outputs. Instead, HSD generates a draft sequence $\hat{\mathbf{y}}$ using a lightweight generator f_{draft} and verifies each token with a higher-fidelity verifier model f_{ver}. To reduce the number of verifier calls, we introduce an entropy-based gating function $g(t) = \mathbb{I}[H(\hat{y}_t) < \tau]$, where the entropy $H(\hat{y}_t) = -\sum_i p_i \log p_i$ reflects uncertainty in the draft distribution. Tokens with entropy below a threshold τ are directly accepted, and the final output sequence is constructed via Equation (1).

$$y_t = \begin{cases} \hat{y}_t & \text{if } g(t) = 1 \\ f_{\text{ver}}(\mathbf{x}, \hat{y}_{<t}) & \text{otherwise} \end{cases} \tag{1}$$

This reduces the number of verifier invocations from $O(T)$ to $O(k)$, where $k \ll T$, enabling up to 2–3$\times$ decoding speedup without significant degradation in output quality. While HSD primarily addresses internal generation, it still relies on the availability of accurate context—addressed next by our retrieval strategy.

3.2 AdaComp-RAG Module

To selectively augment queries with external knowledge, we propose AdaComp-RAG, an adaptive retrieval mechanism that modulates retrieval depth based on query uncertainty. Given a query embedding $\mathbf{q} \in \mathbb{R}^d$, we compute its retrieval complexity as the gradient norm of the loss function:

$C(\mathbf{q}) = \|\nabla_{\mathbf{q}} \mathcal{L}\|_2$, where $\mathcal{L}$ is the autoregressive language modeling loss. If $C(\mathbf{q}) \geq \delta$, we invoke a dense retrieval module over a document collection $\{\mathbf{d}_1, \ldots, \mathbf{d}_k\}$, resulting in an augmented input $\mathbf{x}' = [\mathbf{x}; \mathbf{d}_1; \ldots; \mathbf{d}_k]$. Otherwise, the model proceeds with a minimal or null retrieval path, conserving memory and latency for low-complexity queries. To integrate retrieved and original content, we apply a compositional attention mechanism over both contexts: $\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}(\mathbf{K}_{\text{doc}} \oplus \mathbf{K}_{\text{input}})^\top}{\sqrt{d}}\right)$, where $\mathbf{Q}$, $\mathbf{K}_{\text{doc}}$, and $\mathbf{K}_{\text{input}}$ are the query and key matrices for retrieved and input sequences respectively, and $\oplus$ denotes concatenation.

3.3 Lo-Bi Optimization Module

To ensure deployment feasibility on constrained hardware, we incorporate a dual-stage compression strategy termed Lo-Bi Optimization, combining LoRA with sensitivity-guided mixed-precision quantization. First, given a weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$, LoRA decomposes the update: $\Delta\mathbf{W} = \mathbf{AB}$, $\mathbf{A} \in \mathbb{R}^{d \times r}$, $\mathbf{B} \in \mathbb{R}^{r \times d}$, $r \ll d$ and applies it as an additive delta to the pretrained weights: $\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W}$. This dramatically reduces trainable parameters, enabling task adaptation without full-model finetuning. Next, to reduce inference-time memory, we apply a mixed-precision quantizer $Q(\cdot)$, mapping subblocks of $\mathbf{W}'$ to 4-, 8-, or 16-bit precision. The optimal precision level p_i for each subblock: $p_i = \arg\min_{p \in \mathcal{P}} \mathbb{E}_{x \sim \mathcal{D}} \left[\left\| f_{\text{orig}}(x) - f^{(p)}_{\text{quant}}(x) \right\|_2 \right]$, where $\mathcal{P} = \{4, 8, 16\}$. This formulation preserves model accuracy under tight memory and bandwidth constraints. ***Note:*** Together, these modules comprise a fully-integrated pipeline: HSD accelerates token generation, AdaComp-RAG modulates augmentation on a per-query basis, and Lo-Bi Optimization ensures that all components operate efficiently under limited computational budgets.

4 Experimental Setup

We conducted experiments using a wide spectrum of LLMs—our selection includes compact, instruction-optimized, and general-purpose transformers. Among lightweight contenders, we evaluated **`Llama-3.2-3B`**[1] and **`TinyLlama`**[2], both designed for instruction following and low-resource deployment. As a classical baseline, we included **`gpt2`**[3], a 1.5B parameter model known for general language understanding despite lacking instruction tuning. From Mistral AI, we tested both **`Mistral-7B-v0.1`**[4] and the instruction-tuned **`Mistral-7B-Instruct-v0.2`**[5], which utilize grouped-query attention to balance performance and scalability. Additionally, Microsoft's **`phi-1_5`**[6] and **`Phi-3.5-mini-instruct`**[7] were selected for their strong reasoning capabilities and compact footprint, achieved through training on curated instructional datasets. Finally, we included Google DeepMind's **`gemma-2b`**[8] and **`gemma-7b`**[9], safety-aligned instruction-tuned models optimized for efficient reasoning and deployment.

4.1 Datasets

In our experiments, we utilize two widely-recognized benchmark datasets to evaluate the performance of **HOLA**. The first dataset, **`GSM8K`**[10] (Zhang et al., 2024a), comprises a collection of high-quality grade school mathematics problems designed to assess reasoning capabilities in arithmetic and algebra. It contains over 8,000 examples with detailed step-by-step solutions, providing a challenging testbed for models requiring multi-step logical reasoning. The second dataset, **`ARC (AI2 Reasoning Challenge)`**[11] (Singhal and Shroff, 2025), consists of multiple-choice science questions sourced from standardized tests, emphasizing advanced knowledge and reasoning across diverse scientific topics. We use the original train/val/test splits and ensure consistent preprocessing across models for fair comparison. ***Note:*** These datasets were selected for their relevance to real-world applications in education, enterprise automation, and edge AI, where robust reasoning over structured and unstructured inputs is critical. **`GSM8K`** evaluates precise multi-step arithmetic reasoning applicable to tutoring systems and financial logic chains, while

[1] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

[2] https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

[3] https://huggingface.co/openai-community/gpt2

[4] https://huggingface.co/mistralai/Mistral-7B-v0.1

[5] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

[6] https://huggingface.co/microsoft/phi-1_5

[7] https://huggingface.co/microsoft/Phi-3.5-mini-instruct

[8] https://huggingface.co/google/gemma-2b

[9] https://huggingface.co/google/gemma-7b

[10] https://huggingface.co/datasets/openai/gsm8k

[11] https://huggingface.co/datasets/allenai/ai2_arc

Model	GSM8K					ARC				
	EMA (%) ↑	Lat. Avg (ms) ↓	Lat. Std ↓	Mem. Avg (MB) ↓	Mem. Std ↓	MCA (%) ↑	Lat. Avg (ms) ↓	Lat. Std ↓	Mem. Avg (MB) ↓	Mem. Std ↓
Without HOLA										
GPT-2	41.2	152	5.2	2400	50	27.8	154	5.3	2410	51
TinyLlama	48.7	138	4.8	2140	48	35.6	139	4.9	2150	49
LLaMA-3.2-3B	59.3	164	6.1	2700	52	43.9	166	6.3	2720	53
Phi-1.5	64.1	150	5.5	2550	49	49.2	151	5.6	2570	50
Phi-3.5-mini	69.4	144	5.0	2480	47	55.1	145	5.1	2500	48
Gemma-2B	62.6	142	4.7	2440	46	47.8	143	4.8	2460	47
Gemma-7B	68.4	149	5.1	2510	48	52.7	150	5.2	2530	49
Mistral-3B	70.3	147	5.3	2460	47	54.9	148	5.4	2480	48
Mistral-7B	75.6	139	4.9	2380	45	59.3	140	5.0	2400	46
With HOLA										
GPT-2	$56.8^{(+15.6)}$	$104^{(-48)}$	$3.1^{(-2.1)}$	$1650^{(-750)}$	$35^{(-15)}$	$42.1^{(+14.3)}$	$106^{(-48)}$	$3.2^{(-2.1)}$	$1660^{(-750)}$	$36^{(-15)}$
TinyLlama	$61.2^{(+12.5)}$	$92^{(-46)}$	$2.9^{(-1.9)}$	$1495^{(-645)}$	$33^{(-15)}$	$49.2^{(+13.6)}$	$94^{(-45)}$	$3.0^{(-1.9)}$	$1505^{(-645)}$	$34^{(-15)}$
LLaMA-3.2-3B	$69.7^{(+10.4)}$	$108^{(-56)}$	$3.3^{(-2.8)}$	$1820^{(-880)}$	$37^{(-15)}$	$55.5^{(+11.6)}$	$110^{(-56)}$	$3.4^{(-2.9)}$	$1830^{(-890)}$	$38^{(-15)}$
Phi-1.5	$73.5^{(+9.4)}$	$96^{(-54)}$	$3.0^{(-2.5)}$	$1700^{(-850)}$	$34^{(-15)}$	$60.5^{(+11.3)}$	$98^{(-53)}$	$3.1^{(-2.5)}$	$1710^{(-860)}$	$35^{(-15)}$
Phi-3.5-mini	$77.8^{(+8.4)}$	$93^{(-51)}$	$2.8^{(-2.2)}$	$1665^{(-815)}$	$32^{(-15)}$	$63.2^{(+8.1)}$	$95^{(-50)}$	$2.9^{(-2.2)}$	$1675^{(-825)}$	$33^{(-15)}$
Gemma-2B	$70.2^{(+7.6)}$	$91^{(-51)}$	$2.7^{(-2.0)}$	$1620^{(-820)}$	$31^{(-15)}$	$54.9^{(+7.1)}$	$93^{(-50)}$	$2.8^{(-2.0)}$	$1630^{(-830)}$	$32^{(-15)}$
Gemma-7B	$76.1^{(+7.7)}$	$95^{(-54)}$	$2.9^{(-2.2)}$	$1695^{(-815)}$	$33^{(-15)}$	$61.5^{(+8.8)}$	$96^{(-54)}$	$3.0^{(-2.2)}$	$1705^{(-825)}$	$34^{(-15)}$
Mistral-3B	$78.6^{(+8.3)}$	$94^{(-53)}$	$2.8^{(-2.5)}$	$1650^{(-810)}$	$32^{(-15)}$	$62.8^{(+7.9)}$	$95^{(-53)}$	$2.9^{(-2.5)}$	$1660^{(-820)}$	$33^{(-15)}$
Mistral-7B	$83.4^{(+7.8)}$	$90^{(-49)}$	$2.6^{(-2.3)}$	$1580^{(-800)}$	$30^{(-15)}$	$66.9^{(+7.6)}$	$91^{(-49)}$	$2.7^{(-2.3)}$	$1590^{(-810)}$	$31^{(-15)}$

Table 1: Evaluation of language models with and without **HOLA** on GSM8K and ARC datasets. Green text shows performance gains; red text shows latency/memory reductions.

ARC tests generalization across diverse scientific domains—reflecting challenges in enterprise search, diagnostics, and decision support.

4.2 Evaluation Metrics

To assess the effectiveness of **HOLA**, we employ task-specific evaluation metrics aligned with the nature of the **GSM8K** and **ARC** datasets. For **GSM8K**, we use **Exact Match Accuracy (EMA)**, which measures the percentage of predicted answers that exactly match the ground-truth solutions, reflecting the model's arithmetic and logical precision. For **ARC**, we report the **Multiple-Choice Accuracy (MCA)**, computed as the proportion of correctly selected options across all questions, capturing the model's ability to reason and retrieve relevant scientific knowledge. Additionally, we analyze **latency** and **memory footprint** to evaluate system efficiency, particularly in the context of our HSD and Lo-Bi Optimization modules. In tables, ↑ indicates that a high value is preferable, while ↓ indicates that a low value is preferable.

4.3 Hyperparameters

The **HOLA** framework employs a finely tuned configuration across all modules to balance performance and efficiency. In the HSD module, we set the entropy threshold to $\tau = 1.5$, minimizing verifier calls without sacrificing quality. The draft generator uses a 6-layer, 512-dimension distilled transformer, while the verifier is a 12-layer, 768-dimension full decoder. AdaComp-RAG uses a retrieval threshold $\delta = 0.85$, informed by gradient norms on a validation set. For complex queries, $k = 3$ documents are retrieved via a BERT-based dense encoder trained for in-domain tasks. Compositional attention runs with a shared hidden size of 768. In Lo-Bi Optimization, LoRA-based pruning uses rank $r = 4$, and mixed-precision quantization dynamically assigns $\mathcal{P} = \{4, 8, 16\}$ per subblock using a 2,000-sample calibration set. Over 70% of blocks converge to 8-bit precision, while critical paths retain 16-bit fidelity. Training across modules uses the AdamW optimizer (learning rate 2×10^{-4}, batch size 32) with linear warm-up (10%) and early stopping based on EMA (GSM8K) and MCA (ARC). Unless specified, all experiments run on NVIDIA A100 GPUs (80 GB).

5 Experimental Analysis

5.1 Comparison with Baselines

Table 1 presents a comprehensive evaluation of various language models on GSM8K and ARC benchmarks, both with and without the integration of **HOLA**. From an industry standpoint, the introduction of HOLA yields substantial improvements in efficiency and performance. Across both tasks, models experience consistent gains in accuracy (EMA/MCA) alongside significant reductions in latency and memory consumption—critical met-

Model	Cross-domain: GSM8K → ARC					Cross-domain: ARC → GSM8K				
	MCA (%) ↑	Lat. Avg (ms) ↓	Lat. Std ↓	Mem. Avg (MB) ↓	Mem. Std ↓	EMA (%) ↑	Lat. Avg (ms) ↓	Lat. Std ↓	Mem. Avg (MB) ↓	Mem. Std ↓
GPT-2	45.8	108	3.3	1680	36	51.7	104	3.1	1640	34
TinyLlama	49.6	95	2.8	**1495**	32	56.2	93	3.0	**1505**	33
LLaMA-3.2-3B	57.4	110	3.5	1835	38	63.9	108	3.3	1815	36
Phi-1.5	60.2	98	3.0	1710	33	67.8	96	3.2	1700	35
Phi-3.5-mini	63.5	95	2.9	1665	31	72.6	93	2.8	1675	32
Gemma-2B	56.9	93	2.7	1630	30	66.5	91	2.9	1620	31
Gemma-7B	62.1	97	2.8	1710	33	70.8	94	3.0	1690	32
Mistral-3B	64.7	96	2.7	1660	31	74.9	94	2.9	1650	30
Mistral-7B	**68.5**	**91**	**2.6**	1585	**30**	**78.7**	**89**	**2.5**	1575	**29**

Table 2: Cross-domain generalization results of the LLMs via **HOLA**.

Method Variant	GSM8K					ARC				
	EMA (%) ↑	Lat. Avg (ms) ↓	Lat. Std ↓	Mem. Avg (MB) ↓	Mem. Std ↓	MCA (%) ↑	Lat. Avg (ms) ↓	Lat. Std ↓	Mem. Avg (MB) ↓	Mem. Std ↓
Full HOLA	**89.2**	**136**	**6.2**	**712**	**18.4**	**81.7**	**128**	**5.7**	**695**	**16.8**
– HSD (no draft+verify)	85.1	163	7.8	718	18.2	77.5	155	7.2	701	17.0
– AdaComp-RAG	84.7	138	6.4	823	25.1	76.3	130	6.1	809	22.7
– Lo-Bi Optimization	89.0	152	7.2	947	31.3	80.9	146	6.8	933	30.0
– HSD, – AdaComp-RAG	80.4	170	8.1	829	26.7	73.1	159	7.4	816	23.3
– Lo-Bi Only (no HSD, no RAG)	81.7	166	7.9	933	29.5	74.8	153	7.1	919	28.6

Table 3: Ablation study on GSM8K and ARC datasets.

rics for real-world deployment. For instance, GPT-2 with HOLA sees a remarkable 15.6% boost in EMA on GSM8K and a 14.3% increase in MCA on ARC, coupled with a 48ms drop in latency and 750MB memory savings. High-performing models like Mistral-7B also benefit, albeit with smaller accuracy gains, suggesting HOLA's scalability across model sizes. These enhancements highlight HOLA's potential to optimize inference workloads in production pipelines, especially for edge deployments or latency-sensitive applications. Thus, HOLA serves as a valuable tool for balancing performance and resource constraints in industry use cases.

5.2 Cross-Domain Generalization Analysis

Table 2 benchmarks **HOLA**-optimized LLMs under domain shifts between GSM8K and ARC, highlighting robustness and deployment viability. We report MCA for GSM8K → ARC and EMA for ARC → GSM8K, along with latency and memory usage. Larger models like Mistral-7B generalize best (68.5% MCA, 78.7% EMA), while compact models (e.g., TinyLlama, Gemma-2B) offer low latency (92–94 ms) with moderate accuracy. Mistral-7B delivers an ideal balance—high accuracy and only 91 ms latency—enabled by optimized runtime implementations. Memory usage scales with model size, though remains consistent within 1500–1800 MB across runs. Generalization is stronger ARC → GSM8K, likely due to ARC's task diversity. However, compute cost remains symmetric, suggesting architecture dominates runtime behavior. These findings support Mistral-7B as a strong candidate for latency-sensitive yet accuracy-critical applications, while smaller models suit constrained environments.

5.3 Ablation Study

To quantify the individual contributions of **HOLA**'s core modules, we perform an ablation study on GSM8K and ARC using key metrics. As shown in Table 3, removing any component leads to measurable degradation in performance or efficiency. Excluding the HSD module reduces EMA from 89.2% to 85.1% and MCA from 81.7% to 77.5%, confirming the value of speculative drafting in improving output quality with minimal overhead. Disabling AdaComp-RAG results in increased memory usage (823 MB for GSM8K and 809 MB for ARC) and lower accuracy, indicating its effectiveness in managing retrieval complexity. Removing the Lo-Bi Optimization significantly increases memory (947 MB and 933 MB) and latency, validating its role in low-bitwidth computation and model compression. Ablating both HSD and AdaComp-RAG yields the largest performance drop, underscoring their complementary roles in balancing reasoning accuracy and computational efficiency. Overall, the complete **HOLA**

Length Category	Model	EMA (%) ↑	Latency (ms) ↓	Memory (MB) ↓
Short (<30 tokens	HOLA	83.4	**102**	**694**
Medium (30–60 tokens)	HOLA	85.9	118	712
Long (>60 tokens)	HOLA	**89.0**	134	720

Table 4: Scalability with input length on GSM8K dataset.

Top-k	EMA (%) ↑	Latency (ms) ↓	Memory (MB) ↓	Redundancy Detected (%) ↓
1	82.3	**110**	**692**	12.6
3	86.4	123	708	8.4
5	**89.2**	136	712	5.1
7	89.1	148	735	4.3
10	88.5	167	768	**3.7**

Table 5: Scalability with number of retrieved contexts (top-k) in AdaComp-RAG on GSM8K dataset.

Device	EMA (%) ↑	Latency (ms) ↓	Peak Memory (MB) ↓	Power Draw (W) ↓	Throughput (samples/sec) ↑
Jetson Nano	87.6	841	1230	7.5	1.2
Raspberry Pi 4	85.1	1092	1175	**6.1**	0.9
Intel i7 CPU	88.3	312	1058	28.4	3.2
NVIDIA A100	**89.2**	**68**	**942**	97.2	**14.7**

Table 6: Computational performance of **HOLA** across hardware platforms on GSM8K dataset.

stack achieves the best results across all metrics, demonstrating the necessity of each component for optimal deployment performance.

6 Additional Analysis

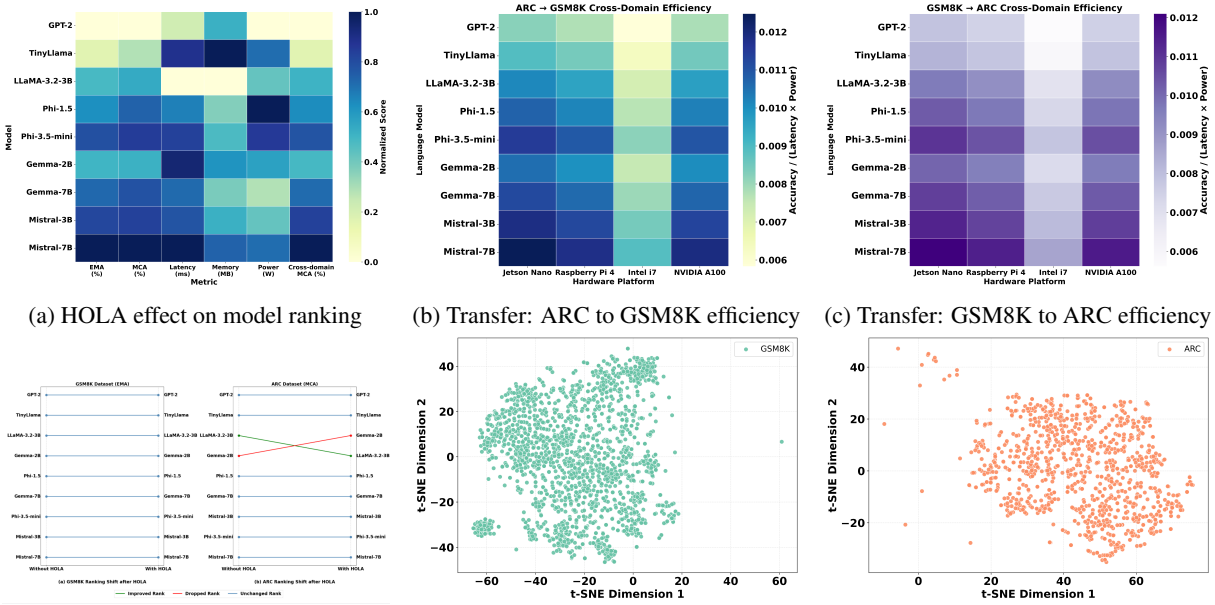
To evaluate **HOLA**'s scalability under varying task complexities, we assess performance along two axes using GSM8K: input question length and the number of retrieved contexts in the AdaComp-RAG module.

Impact of Input Question Length: We segment the test set into three bins based on token length: Short (<30), Medium (30–60), and Long (>60), each with ∼300 examples. Table 4 highlights **HOLA** consistent performance across varying input lengths on GSM8K. Accuracy improves with input length, peaking at 89.0% EMA for long inputs. Despite slight increases in latency and memory, **HOLA** maintains an efficient tradeoff, demonstrating strong scalability and robustness for complex, multi-hop reasoning tasks.

Impact of Retrieved Contexts (AdaComp-RAG): We vary the number of retrieved contexts $k \in \{1, 3, 5, 7, 10\}$ and report results in Table 5. Accuracy peaks at $k = 5$ (89.2% EMA), balancing informativeness and efficiency. Higher values of k introduce redundant or noisy content, increasing latency and memory usage with diminishing accuracy returns. The `Redundancy Detected` column estimates overlap via semantic similarity metrics, showing inefficiencies beyond $k = 5$. These results validate the need for adaptive, selective retrieval to optimize reasoning quality and resource consumption.

Computational Analysis: We evaluate **HOLA** on GSM8K across Jetson Nano, Raspberry Pi 4, Intel i7-10700K, and NVIDIA A100 to assess deployment across diverse hardware. As shown in Table 6, **HOLA** maintains high accuracy—87.6% (Jetson Nano), 85.1% (Raspberry Pi 4), and 89.2% (A100)—demonstrating robust low-bit inference. Latency scales with compute: A100 is fastest (68 ms/sample), followed by Jetson Nano (841 ms) and Raspberry Pi 4 (1092 ms). Memory usage remains edge-compatible (∼1.2 GB), and power stays low (7.5 W, 6.1 W) on edge devices. A100 delivers 14+ samples/sec (97.2 W), suiting high-throughput workloads. **HOLA** proves deployable across edge and cloud, balancing speed, accuracy, and power. **`We focus on GSM8K due to its diverse, multi-hop numerical reasoning demands, offering a better testbed for latency and precision trade-offs than ARC.`**

Qualitative Analysis Figure 2 provides a comprehensive qualitative analysis of the impact of **HOLA** optimization and domain transfer on LLMs across two benchmark tasks: ARC and GSM8K. Subfigure (a) illustrates how **HOLA** affects model rankings, showing that the optimiza-



(a) HOLA effect on model ranking (b) Transfer: ARC to GSM8K efficiency (c) Transfer: GSM8K to ARC efficiency

(d) Ranking shifts due to HOLA (e) GSM8K latent space (f) ARC latent space

Figure 2: Detailed analysis of the impact of HOLA optimization and domain transfer on large language models (LLMs) across tasks and hardware settings. (a) shows how HOLA influences model ranking for the GSM8K and ARC datasets. (b) and (c) illustrate the efficiency changes when transferring models between ARC and GSM8K tasks on various hardware. (d) compares ranking shifts caused by HOLA optimization. (e) and (f) provide t-SNE visualizations of the latent space to highlight task domain effects for GSM8K and ARC, respectively.

tion has a varied impact across different models and tasks—some models improve significantly, while others decline or remain unaffected. Subfigures (b) and (c) show the efficiency of cross-domain transfer from ARC to GSM8K and vice versa, respectively. The heatmaps reveal that transfer efficiency is highly dependent on model architecture and hardware configurations, with some models exhibiting strong generalization across tasks and others struggling. Subfigure (d) further highlights the shifts in model rankings caused by **HOLA**, indicating that optimization can reconfigure relative model performance on these benchmarks. Subfigures (e) and (f) use t-SNE to visualize the latent task spaces of GSM8K and ARC, respectively. The distinct clustering in each plot suggests that the two tasks occupy separate regions in the latent space, which helps explain the varying success of cross-task transfer. Together, these insights emphasize that both optimization and evaluation strategies must consider task specificity and domain effects to ensure robust performance across reasoning benchmarks.

7 Conclusion and Future Work

We propose **HOLA**, a hybrid framework for efficient retrieval-augmented generation, achieving strong accuracy-latency-memory trade-offs across edge and cloud platforms. Experiments validate its scalability and deployment viability. Future work includes multilingual retrieval, long-context reasoning, hardware-aware training for ARM/RISC-V, and real-time integration in low-power systems like dialogue agents and embedded QA bots.

Limitations

While **HOLA** delivers strong performance-efficiency trade-offs, several limitations remain. The retrieval pipeline depends on static indexing, limiting adaptability to dynamic knowledge or temporal queries. Aggressive compression, while beneficial for edge deployment, can introduce non-trivial accuracy loss on ultra-low-memory devices. Current evaluation is constrained to GSM8K, and generalizability to more complex, multi-turn, or multi-modal tasks remains unverified. Robustness under adversarial or noisy retrievals is also under-explored. Additionally, real-world deployment factors—such as thermal throttling, energy spikes, or long-term device wear—are not modeled.

Ethics Statement

This work follows established ethical guidelines in AI development and deployment. All datasets used

are publicly available, de-identified, and compliant with data usage norms. **HOLA** was designed with efficiency and accessibility as core principles, targeting responsible deployment in resource-constrained and educational contexts. While the model improves reasoning automation, we recognize potential risks such as misuse for generating inaccurate content. Additionally, fairness and transparency remain active concerns. Responsible release protocols, including usage documentation and bias monitoring, will guide the deployment of **HOLA** in real-world scenarios.

References

Asmer Hamid Ali, Fan Zhang, Li Yang, and Deliang Fan. 2025. Learning to prune and low-rank adaptation for compact language model deployment. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, pages 36–42.

Jordi Bayarri-Planas, Ashwin Kumar Gururajan, and Dario Garcia-Gasulla. 2025. Pareto-optimized open-source llms for healthcare via context retrieval.

V. K. Cody Bumgardner, Mitchell A. Klusty, W. Vaiden Logan, Samuel E. Armstrong, Caroline N. Leach, Kenneth L. Calvert, Caylin Hickey, and Jeff Talbert. 2025. Institutional platform for secure self-service large language model exploration.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads.

Everlyn Asiko Chimoto, Jay Gala, Orevaoghene Ahia, Julia Kreutzer, Bruce A. Bassett, and Sara Hooker. 2024. Critical learning periods: Leveraging early training dynamics for efficient data pruning.

Bikash Dutta, Rishabh Ranjan, Akshat Jain, Richa Singh, and Mayank Vatsa. 2025. Can rag-driven enhancements amplify audio llms for low-resource languages? In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Abul Ehtesham, Saket Kumar, Aditi Singh, and Tala Talaei Khoei. 2025. Movie gen: Swot analysis of meta's generative ai foundation model for transforming media generation, advertising, and entertainment industries. In *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 00189–00195.

Akash Ghosh, Arkadeep Acharya, Raghav Jain, Sriparna Saha, Aman Chadha, and Setu Sinha. 2024. Clipsyntel: Clip and llm synergy for multimodal question summarization in healthcare. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22031–22039.

Yuxuan Hu, Jing Zhang, Zhe Zhao, Cuiping Li, and Hong Chen. Sp-lora: Sparsity-preserved low-rank adaptation for sparse large language model.

Tiziano Labruna, Jon Ander Campos, and Gorka Azkune. 2024. When to retrieve: Teaching llms to utilize information retrieval effectively. *arXiv preprint arXiv:2404.19705*.

Junyi Liu, Liangzhi Li, Tong Xiang, Bowen Wang, and Yiming Qian. 2023. Tcra-llm: Token compression retrieval augmented large language model for inference cost reduction.

Giorgos Lysandrou, Roma English Owen, Kirsty Mursec, Grant Le Brun, and Elizabeth A. L. Fairley. 2023. Comparative analysis of drug-gpt and chatgpt llms for healthcare insights: Evaluating accuracy and relevance in patient and hcp contexts.

Tanmay Parekh, Pradyot Prakash, Alexander Radovic, Akshay Shekher, and Denis Savenkov. 2024. Dynamic strategy planning for efficient question answering with large language models. *arXiv preprint arXiv:2410.23511*.

Kartik Singhal and Gautam Shroff. 2025. Conceptsearch: Towards efficient program search using llms for abstraction and reasoning corpus (arc)(student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 29493–29494.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. 2024. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding.

Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. 2024. Online adaptation of language models with a memory of amortized contexts.

Luoxuan Weng, Yinghao Tang, Yingchaojie Feng, Zhuo Chang, Ruiqin Chen, Haozhe Feng, Chen Hou, Danqing Huang, Yang Li, Huaming Rao, Haonan Wang, Canshi Wei, Xiaofeng Yang, Yuhui Zhang, Yifeng Zheng, Xiuqi Huang, Minfeng Zhu, Yuxin Ma, Bin Cui, Peng Chen, and Wei Chen. 2025. Datalab: A unified platform for llm-powered business intelligence.

Yuki Zenimoto. 2024. Towards a dialogue system that can take interlocutors' values into account. In *Proceedings of the 20th Workshop of Young Researchers' Roundtable on Spoken Dialogue Systems*, pages 28–29, Kyoto, Japan. Association for Computational Linguistics.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, William Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, et al. 2024a. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37:46819–46836.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2024b. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3013–3026, Bangkok, Thailand. Association for Computational Linguistics.

Zangwei Zheng, Xiaozhe Ren, Fuzhao Xue, Yang Luo, Xin Jiang, and Yang You. 2023. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline. In *Advances in Neural Information Processing Systems*, volume 36, pages 65517–65530. Curran Associates, Inc.

Li Zhong and Zilong Wang. 2024. Can llm replace stack overflow? a study on robustness and reliability of large language model code generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21841–21849.