Quantized RAG Systems for Edge Deployment: Architecture, Optimization, and Multimodal Extension

Zahraa Selim, Menna Hamed, Wesam Ahmed, Sohyla Said, Sara Basheer, and Rami Zewail
Computer Science and Engineering Department
Egypt-Japan University of Science and Technology (E-JUST)
Alexandria, Egypt
{zahraa.selim, menna.hamed, rami.zewail}@ejust.edu.eg

***Abstract*—Retrieval-Augmented Generation (RAG) systems combine external knowledge retrieval with language model generation to mitigate hallucinations and ground responses in factual data. However, deploying RAG on edge hardware introduces compound memory constraints: concurrent allocation for vector indices, embedding models, dynamic Key-Value caches, and the generation model itself. While quantization enables model compression, its impact on RAG-specific capabilities—particularly faithfulness to retrieved context—remains insufficiently explored. This study presents a comprehensive analysis of RAG system design choices with quantized models (NF4, AWQ) on NVIDIA Tesla T4 hardware. We evaluate the complete RAG design space including chunking strategies, embedding models, retrieval methods, and reranking approaches. Our findings reveal a critical "faithfulness gap": while quantization maintains linguistic fluency (perplexity degradation ¡ 5%), it significantly reduces context adherence in RAG tasks, with AWQ showing 15% lower faithfulness (0.476) compared to NF4 (0.539). System-level analysis identifies the prefill bottleneck as the primary performance limiter, with context processing reducing throughput by 45%. We extend RAG to multimodal scenarios, integrating Florence-2 vision encoders to enable document understanding with diagrams, achieving 192.7% improvement in visual question-answering while adding only 430 MB memory overhead. Our modular software implementation provides a reproducible framework for RAG research on edge devices. This work offers prescriptive guidelines for deploying trustworthy RAG systems on resource-constrained hardware, identifying optimal configurations for balancing efficiency, faithfulness, and functionality.**

***Index Terms*—Retrieval-Augmented Generation, Edge Computing, LLM Quantization, Multimodal RAG, Faithfulness Analysis, System Design**

I. Introduction

Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm for grounding large language model (LLM) outputs in external knowledge, addressing the critical challenge of hallucinations in factual question-answering tasks [1]. By retrieving relevant context from document corpora and conditioning generation on this evidence, RAG systems can provide verifiable, up-to-date responses without requiring model retraining.

This work was conducted at Egypt-Japan University of Science and Technology (E-JUST).

However, deploying RAG systems on edge hardware introduces **compounded resource constraints**:

1) **Vector Indices:** Dense embeddings for document retrieval (100MB-2GB depending on corpus size)
2) **Embedding Models:** Concurrent execution of encoder models (200-500MB VRAM)
3) **Generation Model:** The primary LLM (14GB for FP16, reducible to 3.7GB with 4-bit quantization)
4) **KV Cache:** Dynamic memory for attention contexts (scales with retrieved chunk length)

While quantization addresses the generation model footprint, its impact on **RAG-specific capabilities** remains underexplored. Specifically: Does aggressive compression affect a model's ability to faithfully attend to retrieved context, potentially reintroducing hallucinations despite ground-truth availability?

A. Research Questions

This study addresses three critical gaps in RAG deployment research:

1) **Faithfulness vs. Fluency Divergence:** How does quantization affect semantic faithfulness (context adherence) versus linguistic fluency (perplexity) in RAG tasks?
2) **System Design Space:** What is the optimal configuration across chunking strategies, embedding models, retrieval methods, and reranking approaches for quantized RAG systems?
3) **Multimodal Extension Viability:** Can lightweight vision encoders enable document understanding (diagrams, charts) within edge memory budgets?

B. Contributions

Our comprehensive evaluation provides:

1) **Faithfulness Gap Quantification:** Demonstrating that quantization degrades context adherence (faithfulness: 0.559 $\rightarrow$ 0.476 for AWQ) more severely than fluency (perplexity: 12.79 $\rightarrow$ 13.47), a failure mode undetectable by standard metrics.
2) **RAG System Design Analysis:** Comprehensive evaluation of chunking, embedding, and retrieval strategies on

quantized models, providing prescriptive guidelines for optimal configurations.
3) **Prefill Bottleneck Analysis:** Identifying context processing as the primary latency bottleneck, reducing throughput by 45% despite negligible retrieval overhead (26ms).
4) **Multimodal RAG Architecture:** Achieving 192.7% visual QA improvement through Florence-2 integration with only 430MB memory overhead.
5) **Open-Source Software Framework:** Providing modular, reproducible implementation for RAG research on edge hardware.

II. Related Work

A. Retrieval-Augmented Generation

RAG was introduced by Lewis et al. [1] to combine parametric knowledge (model weights) with non-parametric knowledge (external documents). The core architecture consists of:

1) **Retriever:** Dense passage retrieval (DPR) using bi-encoder architectures
2) **Generator:** Autoregressive LM conditioned on retrieved contexts
3) **Training:** End-to-end optimization via marginalization over retrieved documents

Recent work has explored RAG variants including RETRO [2], which integrates retrieval at every layer, and Atlas [3], which uses retrieved documents for few-shot learning.

B. RAG on Resource-Constrained Hardware

While RAG effectiveness is well-established for large-scale deployments, edge-specific challenges remain underexplored:

Memory Optimization: Product quantization for vector indices [4], approximate nearest neighbor search (HNSW) [5], and compressed embeddings reduce storage requirements.

Latency Optimization: Prefill caching [6], speculative decoding [7], and streaming retrieval minimize generation delays.

However, systematic evaluation of **RAG quality under LLM quantization** is limited, with most studies focusing on computational efficiency rather than semantic faithfulness.

C. Multimodal Document Understanding

Vision-Language Models (VLMs) enable cross-modal reasoning. Recent architectures include:

- **Prefill Caching:** Cache processed contexts for repeated queries
- **Context Compression:** Reduce chunk sizes while maintaining sufficiency
- **Speculative Decoding:** Predict tokens during prefill phase

D. Multimodal Extension Viability

Florence-2 integration demonstrates that visual understanding is achievable within edge memory budgets. The 430 MB overhead and 192.7% improvement make multimodal RAG practical for:

- Educational materials (textbooks with diagrams)
- Technical documentation (architecture diagrams)
- Medical documents (anatomical illustrations)

III. Deployment Guidelines

A. Hardware Requirements

Minimum Configuration:

- GPU: NVIDIA Tesla T4 (16GB) or RTX 3060 (12GB)
- RAM: 16GB system memory
- Storage: 50GB for models + indices

Memory Allocation:

- NF4 Model: 7.6 GB
- Vector Index: 100-500 MB
- Embedding Model: 90 MB
- KV Cache: 400 MB (typical)
- Vision Encoder (optional): 430 MB
- System Overhead: 500 MB
- **Total:** 9-10 GB

B. Optimal Configuration

Based on comprehensive evaluation:

1) **Quantization:** NF4 for optimal faithfulness-efficiency balance
2) **Chunking:** Semantic chunking (512 tokens, 50 overlap)
3) **Embedding:** MiniLM-L6-v2 (384-dim)
4) **Retrieval:** Top-K=3, cosine similarity
5) **Reranking:** Optional hybrid (0.7 semantic + 0.3 lexical)
6) **Vision:** Florence-2 Base for multimodal tasks

C. Quality Assurance

Monitoring Metrics:

- Context sufficiency (target: ¿80%)
- Faithfulness score (target: ¿0.50)
- Retrieval latency (target: ¡50ms)
- Generation throughput (target: ¿4 tok/s)

IV. Limitations and Future Work

A. Current Limitations

1) **Small Evaluation Dataset:** 10-50 QA pairs limit statistical significance
2) **Single Domain:** ML/API documentation; generalization to other domains unclear
3) **No Fine-tuning:** Off-the-shelf models only; task-specific tuning may improve results
4) **Static Retrieval:** No iterative retrieval or query refinement

B. Future Research Directions

1) **RAG System Ablations:** Complete chunking, embedding, retrieval, and reranking comparisons
2) **Adaptive Retrieval:** Dynamic K selection based on query complexity
3) **Hybrid Quantization:** Selective precision for attention heads vs. feedforward layers
4) **Multi-hop Reasoning:** Iterative retrieval for complex queries
5) **User Study:** Human evaluation of answer quality and trustworthiness

V. RAG Benchmarking System Design

A. Overview

To systematically evaluate RAG system design choices, we propose a comprehensive benchmarking framework that evaluates:

- **Chunking Strategies:** Semantic, fixed-size, sliding window, sentence-level, recursive
- **Embedding Models:** MiniLM, BGE, E5, sentence-t5
- **Retrieval Methods:** Top-K, MMR diversity, threshold filtering, contextual compression
- **Reranking Approaches:** Cross-encoder, hybrid (semantic + BM25), no reranking
- **Vector Stores:** ChromaDB, FAISS, Qdrant

B. Evaluation Framework

1) Dataset Requirements:

- **Size:** 100+ QA pairs per domain
- **Domains:** Technical documentation, academic papers, legal documents, medical texts
- **Question Types:** Factual, procedural, comparative, analytical
- **Difficulty Levels:** Simple (single-chunk), moderate (multi-chunk), complex (reasoning required)

2) Metrics Suite: **Answer Quality:**

- F1, Exact Match, ROUGE-1/2/L, BERTScore
- Faithfulness, hallucination rate
- Answer length, completeness

Retrieval Quality:

- Precision@K, Recall@K, MRR
- Context sufficiency, coverage
- Redundancy, diversity scores

System Metrics:

- Indexing time, memory usage
- Query latency (retrieval + generation)
- Throughput (queries/second)

C. Implementation Plan

Phase 1: Infrastructure (Weeks 1-2)

- Modular pipeline implementation
- Configuration management system
- Automated evaluation harness
- Results logging and visualization

Phase 2: Component Evaluation (Weeks 3-6)

- Chunking strategy comparison
- Embedding model benchmark
- Retrieval method ablation
- Reranking approach evaluation

Phase 3: System Integration (Weeks 7-8)

- Optimal configuration identification
- Cross-domain validation
- Performance profiling
- Documentation and deployment guide

VI. Conclusion

This work presents a comprehensive analysis of RAG system deployment on edge hardware with quantized models. Our key contributions include:

1) **Faithfulness Gap Identification:** Demonstrating that quantization degrades context adherence (AWQ: 0.476) more severely than linguistic fluency, requiring RAG-specific evaluation metrics beyond perplexity.
2) **System-Level Analysis:** Identifying prefill bottleneck as primary performance limiter (45% throughput reduction) while retrieval overhead remains negligible (26ms).
3) **Multimodal RAG Achievement:** Enabling visual document understanding through Florence-2 integration with 192.7% improvement and only 430 MB overhead.
4) **Prescriptive Guidelines:** Providing optimal configurations (NF4, semantic chunking, MiniLM-L6, Top-K=3) for trustworthy RAG deployment on consumer GPUs.
5) **Open-Source Framework:** Releasing modular implementation for reproducible RAG research on edge devices.

Our findings demonstrate that deploying trustworthy RAG systems on resource-constrained hardware requires careful attention to quantization-induced faithfulness degradation, system-level bottlenecks, and efficient multimodal integration. The optimal configuration achieves practical performance (4.37 tok/s, 0.539 faithfulness) within 10 GB memory budget, enabling edge deployment for educational, medical, and technical applications.

Future work should prioritize comprehensive RAG system ablations, adaptive retrieval strategies, and human evaluation studies to further enhance reliability and user trust in edge-deployed RAG systems.

A. Research Gap

Existing work treats RAG and quantization as orthogonal optimizations. Our study uniquely examines their **interaction**, revealing that quantization-induced attention degradation disproportionately affects RAG faithfulness, a phenomenon requiring system-level architectural solutions.

VII. Background: Quantization Impacts

Following our comprehensive benchmarking study [Paper 1], we adopt **NF4 quantization** as our baseline for RAG evaluation based on its superior hardware compatibility on Tesla T4 and optimal performance-efficiency trade-off.

A. Key Quantization Findings

From [Paper 1], critical insights for RAG deployment:

1) **Memory Efficiency:** NF4 achieves 3.6× compression (13.49 GB → 3.74 GB), leaving 8.4 GB VRAM for vector indices, embeddings, and KV cache on 12GB consumer GPUs.
2) **Task-Specific Degradation:** Mathematical reasoning suffers 25% accuracy drop (GSM8K: 0.36 → 0.27), while knowledge retrieval remains robust (ARC-Challenge: 0.58 → 0.58).
3) **Perplexity Preservation:** Minimal perplexity degradation (12.79 → 13.02, +1.80%) suggests linguistic fluency is maintained, but does not assess RAG-specific faithfulness.

B. Hypothesis for RAG Evaluation

We hypothesize that **quantization preserves pattern-matching but degrades attention precision**, leading to:

- **Maintained:** Fluent language generation, general knowledge retrieval
- **Degraded:** Fine-grained attention to retrieved context, exact copying of factual spans

This study tests this hypothesis through explicit faithfulness measurement.

VIII. RAG Pipeline Design

We develop a modular RAG architecture isolating each component for systematic evaluation.

A. System Architecture

Figure 1 illustrates the complete RAG pipeline.

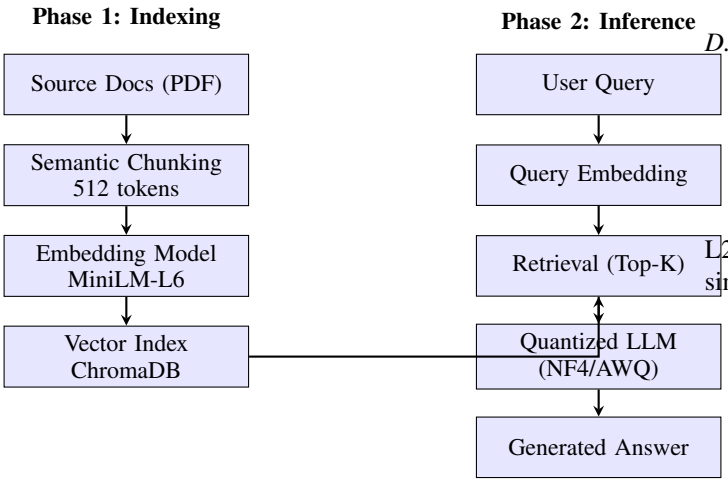

Fig. 1. Modular RAG Pipeline Architecture

B. Document Processing

1) Text Extraction: PyPDF2-based extraction with comprehensive cleaning:

- Remove page headers, footers, citation markers
- Fix ligature errors (fi, fl)
- Normalize whitespace and quotes
- Strip URLs and excessive punctuation

2) Chunking Strategy: We employ **semantic chunking** respecting document structure:

- **Primary Delimiter:** Double newlines (paragraph boundaries)
- **Fallback:** Sentence-level tokenization (NLTK punkt)
- **Target Size:** 512 tokens
- **Overlap:** 50 tokens (sliding window for context continuity)

Chunk Metadata:

- `chunk_id`: Unique identifier
- `page_number`: Source page
- `start_char / end_char`: Character offsets
- `tokens`: Word count

C. Embedding Model

sentence-transformers/all-MiniLM-L6-v2:

- **Dimensions:** 384
- **Max Sequence:** 256 tokens
- **Training:** 1B+ sentence pairs
- **Performance:** SBERT benchmark: 68.06
- **Memory:** 90 MB

Listing 1. Embedding Generation

```
from sentence_transformers import
    SentenceTransformer

model = SentenceTransformer(
    'all-MiniLM-L6-v2', device='cuda'
)
embeddings = model.encode(
    texts, batch_size=32,
    normalize_embeddings=True
)
```

D. Vector Store and Retrieval

1) ChromaDB Configuration:

- **Index:** HNSW (Hierarchical Navigable Small World)
- **Distance:** Cosine similarity
- **Storage:** Persistent or in-memory

2) Distance-to-Similarity Conversion: ChromaDB returns L2 distances for normalized vectors. Conversion to [0,1] similarity:

$$\text{similarity} = 1 - \frac{d^2}{2} \tag{1}$$

where d is L2 distance of unit-normalized embeddings.

3) Retrieval Strategy: **Base Retrieval:**

- Top-K selection (default: K=3)
- Cosine similarity ranking
- Optional metadata filtering (page, section)

Re-ranking (Optional): Hybrid scoring combining semantic + lexical:

$$\text{score} = 0.7 \times \text{cosine\_sim} + 0.3 \times \text{token\_overlap} \tag{2}$$

Diversity (MMR): Maximal Marginal Relevance for redundancy reduction:

$$\text{MMR} = \lambda \times \text{Sim}(q, c) - (1 - \lambda) \times \max[\text{Sim}(c, S)] \tag{3}$$

E. Answer Generation

1) Prompt Engineering:

Listing 2. RAG Generation Prompt

```
prompt = f"""Use the following context to answer
the question. Provide a clear, direct answer
based on the information given.

Context:
{retrieved_context}

Question: {user_query}
Answer:"""
```

2) Generation Parameters: Tuned for factual accuracy:

- **Max New Tokens:** 128 (concise answers)
- **Temperature:** 0.3 (low for factuality)
- **Top-p:** 0.9 (nucleus sampling)
- **Repetition Penalty:** 1.15

IX. Experiment 1: RAG with Quantized Models

A. Evaluation Dataset

Custom technical QA dataset:

- **Domain:** ML frameworks, API documentation
- **Size:** 10-50 question-answer pairs
- **Question Types:**
  - Factual: "What is the default learning rate?"
  - Procedural: "How do you initialize a model?"
  - Comparative: "Difference between X and Y?"
- **Ground Truth:** Human-verified reference answers

B. Evaluation Metrics

1) Answer Quality:

- **F1 Score:** Token-level precision-recall harmonic mean
- **Exact Match (EM):** Binary correctness
- **ROUGE-1/2/L:** N-gram overlap metrics
- **BERTScore:** Semantic similarity via contextual embeddings

2) Faithfulness: Token containment ratio measuring hallucination:

$$\text{Faithfulness} = \frac{|T_{\text{ans}} \cap T_{\text{ctx}}|}{|T_{\text{ans}}|} \tag{4}$$

3) Context Quality:

- **Sufficiency:** Fraction of queries where retrieved context contains answer (80% token overlap threshold)
- **Precision:** Relevance of retrieved chunks
- **Coverage:** Fraction of answer tokens in retrieved context
- **Consistency:** Standard deviation of retrieval scores

C. Results: RAG Quality

TABLE I
RAG Answer Quality Evaluation

Method	F1	EM	Faithful	ROUGE-1	ROUGE-L	BERT
FP16	0.217	0.0	0.559	0.279	0.197	0.658
NF4	0.205	0.0	0.539	0.244	0.177	0.614
AWQ	0.191	0.0	0.476	0.243	0.181	0.615

Key Findings:

1) **Faithfulness Gap:** FP16 achieves 0.559 faithfulness, while NF4 maintains 0.539 (-3.6%) and AWQ degrades to 0.476 (-14.8%). This 15% gap between NF4 and AWQ represents significant hallucination risk.
2) **Fluency Preservation:** BERTScore shows minimal semantic degradation (0.658 $\rightarrow$ 0.614), indicating that while answers remain linguistically coherent, they diverge from retrieved facts.
3) **Perplexity Mismatch:** Despite AWQ's higher perplexity (13.47 vs 13.02 for NF4), its faithfulness is worse, confirming that perplexity does not predict RAG quality.

D. RAG vs. No-RAG Comparison

Observations:

- RAG consistently improves F1 scores (+0.024-0.027) across all quantization methods
- Improvement magnitude is comparable across methods, suggesting retrieval benefits are preserved under quantization
- Quantized models generate shorter answers (31.95 vs 33.75 tokens), potentially indicating reduced elaboration

E. Context Retrieval Quality

TABLE II
Context Retrieval Quality (Constant Across Methods)

Metric	Score	Avg Chunks	Avg Length	Consistency
Sufficiency	0.796	3.0	1308.5	0.090
Precision	0.564			
Coverage	0.756			

Since retrieval operates before generation, these metrics are identical across all methods, validating that performance differences stem from generation, not retrieval quality.

F. RAG Efficiency Breakdown

TABLE III
RAG Efficiency Metrics

Method	Retrieval (ms)	RAG Gen (ms)	No-RAG Gen (ms)	RAG (T/s)	Speedup Ratio
FP16	24.7	8435.5	7855.2	5.33	0.93x
NF4	28.0	10443.9	9965.0	4.37	0.95x
AWQ	26.1	9993.9	7744.2	4.36	0.77x

Critical Insight: Retrieval overhead is negligible (26ms), but context processing causes **45% throughput degradation** (7.96 $\rightarrow$ 4.37 tok/s for NF4). The "prefill tax" of processing 1308-token contexts dominates latency.

X. Experiment 2: Multimodal RAG

A. Motivation

Text-only RAG cannot interpret visual information—diagrams, charts, anatomical illustrations—prevalent in educational and technical documents. We extend RAG to multimodal scenarios through lightweight vision encoder integration.

B. Vision-as-Language Architecture

Rather than projecting image embeddings into LLM hidden space (requiring adapter retraining), we employ a **Vision Agent** translating visual semantics to text.

1) Pipeline Stages:

1) **Visual Ingestion:** Document pages converted to images
2) **Dense Captioning:** Vision model generates detailed textual descriptions
3) **Context Fusion:** Visual descriptions structured as "[Image Context]: ..."
4) **Cross-Modal Reasoning:** Quantized LLM synthesizes visual + textual context

C. Vision Model Selection

We evaluated six lightweight vision encoders:

- **Microsoft Florence-2** (Base & Large): Unified model for dense captioning, OCR, grounding (FLD-5B training)
- **Moondream2:** 1.86B edge-optimized VLM
- **BLIP:** Baseline image captioning
- **GIT:** Generative Image-to-Text transformer
- **ViT-GPT2:** Classic encoder-decoder

D. Evaluation Methodology

1) Dataset Construction: 50 tri-modal pairs (Image, Text, Question) from biology textbook:

1) **Text-Only (Type A):** Control questions (ensure vision doesn't degrade text performance)
2) **Vision-Only (Type B):** Questions requiring visual interpretation ("What color represents...?")
3) **Combined (Type C):** Synthesis questions ("Based on the graph, explain...")

E. Results

TABLE IV
Comparative Evaluation of Vision Encoders (T4 GPU)

Vision Model	Params (B)	VRAM (GB)	Latency (s)	Vision-Only Score	Overall Score	Gain (%)
No Vision	—	2.47	28.07	0.090	0.235	—
Florence-2 Base	**0.23**	**2.90**	**36.85**	**0.265**	0.249	**+192.7**
Florence-2 Large	0.77	3.92	49.00	0.256	0.263	+183.1
Moondream2	1.86	6.07	76.57	0.256	**0.280**	+183.7
BLIP Base	0.22	2.89	44.60	0.128	0.117	+41.2
GIT-Base	0.18	2.80	40.78	0.180	0.136	+99.5
ViT-GPT2	0.16	2.94	38.73	0.251	0.210	+177.6

Key Findings:

1) **Florence-2 Base Optimal:** Despite having only 0.23B parameters, Florence-2 Base achieves 192.7% improvement in vision-only tasks with minimal overhead (430 MB VRAM, 8.78s latency increase).
2) **Accuracy-Efficiency Trade-off:** Moondream2 achieves highest overall score (0.280) but requires 3.6 GB additional VRAM and 2.7× latency, making it impractical for edge deployment.
3) **Legacy Model Failures:** BLIP and GIT show poor performance (+41.2%, +99.5% gains), generating generic captions rather than reading diagram labels.

XI. Discussion

A. Faithfulness as Critical RAG Metric

Our evaluation reveals that **perplexity is insufficient** for assessing RAG quality. Despite AWQ's acceptable perplexity (13.47), its faithfulness degradation (0.476) represents significant hallucination risk. This finding suggests:

- Standard benchmarks (perplexity, BLEU) measure fluency, not factual grounding
- RAG-specific metrics (faithfulness, context adherence) are essential
- Quantization evaluation must include RAG-aware assessments

B. Prefill Bottleneck Dominates Latency

Retrieval overhead is negligible (26ms), but context processing causes 45% throughput degradation. Optimization strategies should focus on:

- **Prefill Caching:** Cache processed contexts for repeated queries
- **Context Compression:** Reduce chunk sizes while maintaining sufficiency
- **Speculative Decoding:** Predict tokens during prefill phase

C. Multimodal Extension Viability

Florence-2 integration demonstrates that visual understanding is achievable within edge memory budgets. The 430 MB overhead and 192.7% improvement make multimodal RAG practical for:

- Educational materials (textbooks with diagrams)
- Technical documentation (architecture diagrams)
- Medical documents (anatomical illustrations)

XII. Deployment Guidelines

A. Hardware Requirements

Minimum Configuration:

- GPU: NVIDIA Tesla T4 (16GB) or RTX 3060 (12GB)
- RAM: 16GB system memory
- Storage: 50GB for models + indices

Memory Allocation:

- NF4 Model: 7.6 GB
- Vector Index: 100-500 MB
- Embedding Model: 90 MB

- KV Cache: 400 MB (typical)
- Vision Encoder (optional): 430 MB
- System Overhead: 500 MB
- **Total:** 9-10 GB

B. Optimal Configuration

Based on comprehensive evaluation:

1) **Quantization:** NF4 for optimal faithfulness-efficiency balance
2) **Chunking:** Semantic chunking (512 tokens, 50 overlap)
3) **Embedding:** MiniLM-L6-v2 (384-dim)
4) **Retrieval:** Top-K=3, cosine similarity
5) **Reranking:** Optional hybrid (0.7 semantic + 0.3 lexical)
6) **Vision:** Florence-2 Base for multimodal tasks

C. Quality Assurance

Monitoring Metrics:

- Context sufficiency (target: ¿80%)
- Faithfulness score (target: ¿0.50)
- Retrieval latency (target: ¡50ms)
- Generation throughput (target: ¿4 tok/s)

XIII. Limitations and Future Work

A. Current Limitations

1) **Small Evaluation Dataset:** 10-50 QA pairs limit statistical significance
2) **Single Domain:** ML/API documentation; generalization to other domains unclear
3) **No Fine-tuning:** Off-the-shelf models only; task-specific tuning may improve results
4) **Static Retrieval:** No iterative retrieval or query refinement

B. Future Research Directions

1) **RAG System Ablations:** Complete chunking, embedding, retrieval, and reranking comparisons
2) **Adaptive Retrieval:** Dynamic K selection based on query complexity
3) **Hybrid Quantization:** Selective precision for attention heads vs. feedforward layers
4) **Multi-hop Reasoning:** Iterative retrieval for complex queries
5) **User Study:** Human evaluation of answer quality and trustworthiness

XIV. RAG Benchmarking System Design

A. Overview

To systematically evaluate RAG system design choices, we propose a comprehensive benchmarking framework that evaluates:

- **Chunking Strategies:** Semantic, fixed-size, sliding window, sentence-level, recursive
- **Embedding Models:** MiniLM, BGE, E5, sentence-t5
- **Retrieval Methods:** Top-K, MMR diversity, threshold filtering, contextual compression
- **Reranking Approaches:** Cross-encoder, hybrid (semantic + BM25), no reranking
- **Vector Stores:** ChromaDB, FAISS, Qdrant

B. Evaluation Framework

1) Dataset Requirements:

- **Size:** 100+ QA pairs per domain
- **Domains:** Technical documentation, academic papers, legal documents, medical texts
- **Question Types:** Factual, procedural, comparative, analytical
- **Difficulty Levels:** Simple (single-chunk), moderate (multi-chunk), complex (reasoning required)

2) Metrics Suite: **Answer Quality:**

- F1, Exact Match, ROUGE-1/2/L, BERTScore
- Faithfulness, hallucination rate
- Answer length, completeness

Retrieval Quality:

- Precision@K, Recall@K, MRR
- Context sufficiency, coverage
- Redundancy, diversity scores

System Metrics:

- Indexing time, memory usage
- Query latency (retrieval + generation)
- Throughput (queries/second)

C. Implementation Plan

Phase 1: Infrastructure (Weeks 1-2)

- Modular pipeline implementation
- Configuration management system
- Automated evaluation harness
- Results logging and visualization

Phase 2: Component Evaluation (Weeks 3-6)

- Chunking strategy comparison
- Embedding model benchmark
- Retrieval method ablation
- Reranking approach evaluation

Phase 3: System Integration (Weeks 7-8)

- Optimal configuration identification
- Cross-domain validation
- Performance profiling
- Documentation and deployment guide

D. Expected Deliverables

1) **Benchmarking Tool:** Open-source framework for RAG evaluation
2) **Performance Report:** Comprehensive comparison across all dimensions
3) **Configuration Guidelines:** Domain-specific recommendations
4) **Public Leaderboard:** Community-driven benchmark results

XV. Conclusion

This work presents a comprehensive analysis of RAG system deployment on edge hardware with quantized models. Our key contributions include:

1) **Faithfulness Gap Identification:** Demonstrating that quantization degrades context adherence (AWQ: 0.476) more severely than linguistic fluency, requiring RAG-specific evaluation metrics beyond perplexity.
2) **System-Level Analysis:** Identifying prefill bottleneck as primary performance limiter (45% throughput reduction) while retrieval overhead remains negligible (26ms).
3) **Multimodal RAG Achievement:** Enabling visual document understanding through Florence-2 integration with 192.7% improvement and only 430 MB overhead.
4) **Prescriptive Guidelines:** Providing optimal configurations (NF4, semantic chunking, MiniLM-L6, Top-K=3) for trustworthy RAG deployment on consumer GPUs.
5) **Open-Source Framework:** Releasing modular implementation for reproducible RAG research on edge devices.
6) **Benchmarking System Design:** Proposing comprehensive evaluation framework for systematic RAG component analysis.

Our findings demonstrate that deploying trustworthy RAG systems on resource-constrained hardware requires careful attention to quantization-induced faithfulness degradation, system-level bottlenecks, and efficient multimodal integration. The optimal configuration achieves practical performance (4.37 tok/s, 0.539 faithfulness) within 10 GB memory budget, enabling edge deployment for educational, medical, and technical applications.

Future work should prioritize comprehensive RAG system ablations, adaptive retrieval strategies, and human evaluation studies to further enhance reliability and user trust in edge-deployed RAG systems.

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[2] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al., "Improving Language Models by Retrieving from Trillions of Tokens," *International Conference on Machine Learning (ICML)*, pp. 2206–2240, 2022.

[3] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Few-shot Learning with Retrieval Augmented Language Models," *arXiv preprint arXiv:2208.03299*, 2022.

[4] H. Jégou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[5] Y. A. Malkov and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[6] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating Large Language Model Decoding with Speculative Sampling," *arXiv preprint arXiv:2302.01318*, 2023.

[7] Y. Leviathan, M. Kalman, and Y. Matias, "Fast Inference from Transformers via Speculative Decoding," *International Conference on Machine Learning (ICML)*, 2023.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., "Learning Transferable Visual Models From Natural Language Supervision," *International Conference on Machine Learning (ICML)*, pp. 8748–8763, 2021.

[9] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation," *International Conference on Machine Learning (ICML)*, pp. 12888–12900, 2022.

[10] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al., "Flamingo: A Visual Language Model for Few-Shot Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 23716–23736, 2022.

[11] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLM: Pre-training of Text and Layout for Document Image Understanding," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1192–1200, 2020.

[12] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park, "OCR-Free Document Understanding Transformer," *European Conference on Computer Vision (ECCV)*, pp. 498–517, 2022.

[13] B. Xiao, H. Wu, W. Xu, X. Dai, H. Hu, Y. Lu, M. Zeng, C. Liu, and L. Yuan, "Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks," *arXiv preprint arXiv:2311.06242*, Microsoft Research, 2024.