Precision Is Not Performance: A Utility-Aware Evaluation of Quantized LLM Inference

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) have become an increasingly important part of most modern AI systems; however, as LLMs grow in size, their usable responses are delayed. Additionally, it is challenging to achieve efficient inference using LLMs in the absence of sufficient resources, such as memory and computing power, because memory consumption and computing costs become significant concerns for utilizing LLMs efficiently. To address these concerns, Quantization methods are used. Quantization refers to reducing numerical precision during the inference stage of the model to reduce memory usage. Through quantization, model memory usage and cost efficiency can be enhanced. Unfortunately, research into quantization has typically focused on theoretical performance predictions and sample performance testing (i.e., isolated performance benchmarks), providing a limited view of how reduced numerical precision would impact the end-to-end behavior of inferred responses from the model in the real world. As a result, a significant gap exists in the practical ability to make decisions about deploying quantized LLM models. To help fill this gap, we propose a novel Utility-aware Quantization Evaluation Framework (UAQF). The proposed UAQF is evaluated using multiple instruction-tuned LLMs, including LLaMA-2-7B and LLaMA-2-13B, and evaluated across three instruction-tuned LLM variants spanning 7B and 13B parameter scales. The framework is tested across FP16, 8-bit, and 4-bit quantization, using a diverse set of prompts, and the resulting end-to-end latency and throughput are compared against established quantization approaches such as GPTQ, AWQ, ZeroQuant, Atom, and SmoothQuant. The experimental results indicate that lower-bit quantization consistently improves throughput with minimal impact on output quality across models and prompts. Moreover, the analysis reveals that aggressive quantization often provides greater overall utility than intermediate-precision settings, highlighting deployment-level behaviors that are not evident in single-model or isolated-metric evaluations. These results demonstrate that UAQF enables deeper empirical insight into quantization efficiency and system behavior than existing approaches, reinforcing the need for deployment-oriented quantization assessment.

1 Introduction

1.1 Background

In the modern world of artificial intelligence, LLMs are now used as fundamental components of AI systems. LLMs have been optimized for performance on several different types of NLP uses like understanding language, performing reasoning, generating dialog and even creating documents. As the number of model parameters increase, the size/complexity of models continues to grow; hence, it has become increasingly computationally intensive to get the faster and accurate response from LLM models (Bian et al., 2025). The deployment of these types of models in different types of real-world use cases - from cloud-based services to deploying LLM models on an intelligent device (IOT's), presents new challenges, especially concerning latency, throughput of the finished product, power consumption and cost associated with operating LLM's. IoT systems often require real-time decision making (e.g., autonomous drones, industrial automation, smart

vehicles). Deploying full 70B models directly on constrained IoT devices is typically impractical due to: Memory limitations, Compute constraints, and Energy consumption. The use of quantization is a very effective way of reducing the impact of the constraints associated with using LLM-based AI solutions (Lang et al., 2024). Quantization involves breaking down the numerical precision of the parameters and computations used to train an LLM into lower-bit integers or other lower-precision representations (e.g., FP16, BF16, INT8). Through quantization, LLM users can gain both the ability to store the model in less memory as well as faster inference when performing linguistic understanding and language generation tasks. However, there is no guarantee that the benefits of quantization will hold equally across different models, tasks, and hardware platforms, and these gains may come with trade-offs in output quality or behavioral variations across model configurations (Mekala et al., 2025).

1.2 Motivation

Many recent studies exploring mixed-precision quantization and post-training quantization strategies have examined the ways these methods impact the accuracy, robustness, and performance of LLMs across benchmarks (Lang et al., 2024; Li et al., 2024; Xiao et al., 2024). The conclusion reached by these authors is that appropriately developed low-bit quantization can increase the efficiency of large language models while maintaining comparable output quality. They further discuss important considerations regarding trade-offs between model performance and the speed of continued model development.

Previous quantitative studies have focused on how quantization impacts different classes of tasks and behaviors associated with LLMs. For example, long-context reasoning and code generation were used to demonstrate that quantized LLMs demonstrate inconsistency in degradation among tasks; that degradation not only has an inconsistent effect on results across tasks, but also that degradation is influenced by factors such as the length of the context being evaluated, the complexity of each task involved in the evaluation, and the overall size and comprehensiveness of the associated LLM (Lee et al., 2025; Mekala et al., 2025; Melin et al., 2024). In addition, other authors have proposed salience and smoothness-based approaches as effective methods to minimize the potential loss of precision resulting from quantization and have suggested that mixed-precision allocation and activation smoothing (Huang et al., 2025b; Xiao et al., 2024) are some of these effective methods. In addition to the existing works mentioned above, several authors have produced comprehensive benchmarking studies that have produced extensive taxonomies of existing and developing quantization methods, along with established protocols for standardized assessment, and provided insight into the strengths and weaknesses of the various existing quantization strategies (Zhao et al., 2025).

Although considerable strides have been made in the quantization of LLMs, there are still serious limitations in the current literature regarding real-world implementation insights. Most studies to date have focused on evaluating quantized LLMs based solely on their accuracy, perplexity, or benchmark scores, relatively, with very few evaluating them on other important factors, such as end-to-end inference latency, throughput, and wall-clock time, especially when measured under a more realistic environment compared to laboratory testing. Likewise, at the hardware level, most studies report little or no information regarding how much memory bandwidth was used, kernel efficiencies, and accelerator-specific characteristics, which limits our ability to evaluate how well quantized LLMs will perform when deployed in real-world applications/ends users. In many cases, proposed methods add extra architectural or calibration complexity but fail to provide quantitative measures of the overheads and trade-offs (with respect to all system elements) associated with quantized inferences. Consequently, there is still limited investigation into the relation between quantization methods, hardware, and deployment efficiency.

In response to these issues, a utility-aware, end-to-end evaluation framework has been developed for quantized large language models that will provide a realistic approach to determining how quantized inference performs with fully deployed models, task-representative prompts, and target inference hardware. This framework allows realistic evaluation of quantized inference, including both the loss in performance (due to reduced numerical precision) and the quality of output produced due to the reduced precision. The framework has also been designed to be modular and extensible, enabling other forms of quantization to be added to the framework without changing the basic evaluation process used. This makes it easy to replicate other studies' results, as well as allows for further developments in the methodology in the future.

1.3 Contributions and Research Objectives

The key contributions and guiding objectives of this work are summarized as follows:

- **Utility-Aware, Deployment-Oriented Evaluation Framework:** A utility-aware framework is introduced that can be used to evaluate the performance of fully quantized LLM inference. Existing approaches often rely on direct experimentation on real hardware to evaluate LLM inference performance, which can be costly and time-intensive, thereby limiting the range of evaluation parameters and configurations explored. In contrast, a more comprehensive, cost-effective, and time-efficient end-to-end evaluation methodology is proposed through the Utility-aware Quantization Evaluation Framework (UAQF), in which LLM inference capability is assessed using latency, throughput, and proxy quality metrics under deployment-like conditions. The design of this framework enables the systematic and repeatable evaluation of multiple models, combinations of inputs, quantization configurations, and end-hardware units, while providing an estimate of the utility of an LLM when deployed. Furthermore, this research aims to establish a principled framework for comparing quantization strategies through a unified utility-based formulation and to provide deployment-relevant insights that inform precision selection in practical LLM systems.

- **Comparative Study Across Model Scales and Quantization Methods:** A systematic empirical testing of LLMs that have been trained based on the large knowledge base: LLaMA-2-7B and LLaMA-2-13B, is performed. All testing was performed with respect to their runtime performance for quantization of both their weights and activation values under 16-bit precision, 8-bit precision, and 4-bit precision. The results were compared against the results from established methods of post-training quantisation (i.e., GPTQ, AWQ, ZeroQuant, Atom, and SmoothQuant) to determine the scalability and performance trends under the same experimental configurations. Additionally, this research seeks to empirically measure the impact of different quantization levels on LLM inference latency and throughput under realistic conditions.

- **Utility-Based Precision–Performance Trade-off Analysis:** An assessment of utility through the measurement and evaluation of gains in efficiency and losses in quality has been presented. This allowed for the direct comparison of the quantization configurations. In addition, the extent to which low-bit quantization affects output quality is assessed using scalable proxy metrics.

- **Reproducible Engineering Infrastructure for Quantized Inference:** A resumable, chunked inferencing framework with deterministic decoding and structured log records is proposed. This provides the means to execute research experiments that are stable and repeatable, as well as generate fine-grained measurements of system-level performance when operating under the confines of constrained environment conditions.

2 Literature Review

The field of research into Quantization of LLMs has become increasingly important because it enables the efficient deployment of such large models. Post-training quantization (PTQ) Techniques help to decrease the amount of Memory and Compute required for deploying Models while helping to Maintain Accuracy for the job at hand.

Research studies conducted early on Evaluating Quantized LLMs (Li et al., 2024) indicate that quantization has uneven effects on Task Accuracies, with some 4-bit quantization models being acceptable when the model is tuned correctly. However, these studies tend to ignore some of the real-world metrics that would be used to evaluate the System's Performance, such as End-to-End Latency and Throughput, thus limiting their usefulness in practice.

Methodological improvements focus mainly on minimizing the loss of accuracy associated with quantization. The SmoothQuant approach proposed in (Xiao et al., 2024) demonstrates near-FP16 accuracy at 8-bit precision through activation smoothing and weight quantization. While efficiency is reported, the study does not

Table 1: Summary of Key Research on Quantization of Large Language Models (Part I)

Paper	**Core Research Focus**	**Methods Used**	**Key Findings / Results**	**Primary Scope Limitations**
(Malekar & Zand, 2025)	Throughput-centric analysis of extreme LLM quantization	Amdahl's Law modeling; W4/W3 quantization across GPU kernels	Quantization benefits saturate due to non-GEMM bottlenecks	Emphasizes kernel-level modeling; full application-level end-to-end latency is not the primary evaluation focus
(Elangovan et al., 2025)	Ultra-low-bit LLM inference optimization	Locally optimal block-clustered W4A4 quantization	Improves accuracy over uniform 4-bit schemes at similar compute cost	Evaluation centers on quantization effectiveness; system-level inference behavior is explored in limited settings
(Huang et al., 2025b)	Mixed-precision quantization	Salience detection with mixed-bit allocation	Mixed precision outperforms uniform low-bit quantization	Focuses on accuracy–compression trade-offs; deployment-level latency and throughput are not jointly analyzed
(Mekala et al., 2025)	Effect of quantization on long-context reasoning	PTQ across long-context benchmarks	Long-context tasks are more sensitive to quantization	Emphasizes task-level quality; hardware-aware latency analysis is not the primary objective
(Shi & Ding, 2025)	Performance energy quality trade-offs	Controlled experiments on A100/H100 GPUs	Lower precision reduces energy; quality loss varies by task	Evaluated under controlled infrastructure; broader deployment reproducibility is not extensively explored
(Zhao et al., 2025)	Taxonomy and comparison of PTQ methods	Unified benchmarking framework	No universally optimal PTQ method	Focuses on comparative algorithm benchmarking rather than deployment-oriented inference behavior
(Lang et al., 2024)	Large-scale quality evaluation of quantized LLMs	GPTQ, AWQ, FP8 across 13 benchmarks	Larger models tolerate quantization better	Prioritizes quality assessment; system-level latency and throughput are not jointly evaluated
(Melin et al., 2024)	Quantization impact on code generation	Code benchmarks with human evaluation	Code generation is highly sensitive to low-bit precision	Narrow task focus; findings may not generalize to broader inference workloads
(Xiao et al., 2024)	Accurate INT8 post-training quantization	Activation smoothing with weight quantization	Near-FP16 accuracy achievable for many models	Emphasizes method design and accuracy preservation; deployment utility is assessed via isolated speedup measurements

Table 2: Summary of Key Research on Quantization of Large Language Models (Part II)

Paper	**Core Research Focus**	**Methods Used**	**Key Findings / Results**	**Primary Scope Limitations**
(Liu et al., 2024c)	Extreme low-bit post-training quantization	Vector-based post-training quantization	Maintains accuracy at very low bit-widths	Focuses on quantization robustness; hardware-aware end-to-end inference behavior is not extensively analyzed
(Lee et al., 2025)	Quantization trade offs across model scales	Multiple PTQ schemes on edge-to-large models	Task difficulty and model size strongly affect tolerance	Focuses on cross-scale trends; system deployment behavior is not the main evaluation target
(Huang et al., 2025a)	Ternary quantization for LLMs	Trapping-free ternary optimization	Enables ultra-low-bit inference with stable training	Hardware support and deployment feasibility are not extensively evaluated
(Lin et al., 2025)	Quantization of diffusion-based LLMs	PTQ adapted for diffusion architectures	Quantization feasible beyond autoregressive LLMs	Limited applicability to standard autoregressive LLM inference pipelines
(Hasan, 2024)	Comparison of PTQ and QAT strategies	Empirical analysis across quantization regimes	QAT outperforms PTQ under aggressive compression	Emphasizes algorithmic comparison; system-level inference metrics are limited
(Sharify et al., 2024)	Novel numerical formats for PTQ	Microscaling-based low-bit representations	Improved accuracy–compression trade-offs	End-to-end inference latency and throughput are not primary evaluation metrics
(Liu et al., 2024b)	Kernel-aware post-training quantization	Reduced kernel size for precise compression	Improves accuracy at low bit-widths	Evaluation focuses on kernel behavior; full LLM inference pipelines are less explored
(Zhao et al., 2024)	Learnable post-training quantization	Optimization of quantization parameters	Improves robustness over fixed PTQ methods	Introduces training overhead; deployment-oriented inference analysis is limited
(Li et al., 2024)	Systematic evaluation of quantized LLMs	PTQ across tasks and bit-widths	Quantization impact varies significantly across tasks	Hardware-aware latency and throughput analysis is limited
(Jin et al., 2024)	Comprehensive evaluation of quantization strategies	Multi-method, multi-task benchmarking	Confirms no single best quantization approach	Focuses on algorithmic evaluation rather than deployment-level efficiency
(Liu et al., 2023)	Data-free quantization-aware training	Synthetic calibration and QAT	Avoids reliance on proprietary datasets	Increased training complexity; inference-time system behavior is not the main focus

explicitly examine end-to-end inference behavior under a unified utility-oriented evaluation framework. Another approach, SliM-LLM (Huang et al., 2025b), employs mixed-precision quantization by identifying salient weights and allocating different bit-widths, achieving improved accuracy over uniform low-bit quantization. While effective from an accuracy perspective, the approach introduces additional algorithmic complexity, and its evaluation primarily emphasizes quality gains rather than deployment-oriented analysis of end-to-end latency and throughput. Long-Context Quantization (Mekala et al., 2025) conducted research to analyze the effects of quantizing based on how much context is used to solve tasks. They found that low-bit quantization significantly impacts tasks needing long context to complete effectively. However, measures of this method were based on tantalizingly perfect benchmarks and therefore limited reporting of actual performance under hardware realistic conditions.

The effects of model scale and task diversity on quantization have been established through the use of large-scale evaluation studies. For example, the comprehensive evaluation in (Lang et al., 2024) shows that larger language models (up to 405B parameters) generally exhibit greater tolerance to quantization compared to smaller models (e.g., 125M–2.8B parameters). However, the study primarily focuses on accuracy and robustness trends across model scales, with limited attention to deployment-oriented performance metrics such as inference latency, throughput, energy consumption, and kernel-level behavior. In contrast, Systematic Characterization of LLM Quantization (Shi & Ding, 2025) provided controlled experiments on A100 and H100 GPUs to evaluate the relationship between performance, energy usage, and quality when applying lower-bit quantization methods; they showed that lower bit models had reduced energy usage than higher bit models while emphasizing that quality loss is determined largely by the task that is addressed. The Limitations of this study include reliance on stable infrastructure and challenges in replicating their results.

Research has also clarified that there are domain specific sensitivities to quantization; for example, Precision or Peril (Melin et al., 2024) looked at how low-bit quantization affects code generation tasks and concluded that there is a need to develop task-specific quantization methods for code generation tasks. Furthermore, subsequent work such as Benchmarking PTQ in LLMs (Zhao et al., 2025) has categorised the different options available for development, and while there is no clear answer as to which approach will provide the best results for all situations, these types of studies focus primarily on the order in which algorithms rank rather than the usefulness of each algorithm in actual production environments.

Although quantization serves as the basis for model efficiency, related research has continued to develop techniques for model adaptation and specialization. For example, PANDA (Liu et al., 2024a) focuses on creating efficient adaptation strategies using a lightweight approach and structure for integrating stored knowledge so as to reduce forgetting and preserve their memory of previously learned information. The utility of its models in their respective domains is improved, but it is primarily complementary to low-bit quantization, in that inference efficiency or latency have not been considered.

There has been considerable progress towards reducing the memory and computational cost of LLMs through quantization. The review of different quantization methods is summarized in table 1 and table 2. Despite significant progress in quantization algorithms, existing work often provides limited insight into deployment-level performance metrics, including latency, throughput, and energy efficiency, particularly across heterogeneous hardware and application scenarios. Further investigation along these dimensions is necessary to better understand the practical implications of quantized LLM deployment.

3 Proposed Utility-aware Quantization Evaluation Framework (UAQF)

This section describes proposed framework for utility-aware quantization evaluation in LLMs. Figure 1 illustrate the step-wise flow of the framework, from data preparation through quantized inference, metric logging, and utility-driven analysis. The Algorithm 3 denotes steps of the proposed UAQF. The notations are listed in Table 4. The UAQF framework is meant to facilitate the systematic evaluation and deployment of low-bit quantized models by providing an explicit mechanism to balance output quality against inference efficiency (latency, throughput, and energy use).

The UAQF has been designed as an approach to evaluate and make decisions regarding the performance of quantization frameworks. UAQF Adaptive refers specifically to a strategy used at the deployment level

to select a quantization configuration that results in the greatest utility score when defined by the user's operational priorities. Weighting parameters in the utility function represent a user's operational priorities. It is also important to note that the UAQF Adaptive does not make any changes to existing quantization processes, does not retrain models, and does not introduce new methods for adaptively changing bit-widths; it selects among configurations that have already been evaluated on the basis of trade-offs concerning three aspects of their performance (efficiency, quality, and memory usage). This clarification establishes that the contribution of UAQF Adaptive is to create a selection process that is structured and based on the purpose of producing quantizations with the greatest utility rather than creating a new method for producing quantizations.

3.1 Evaluation Objectives

The proposed UAQF evaluates LLMs along three primary system-level metrics:

- **Latency (L):** Average time required to generate a response per prompt.
- **Throughput (T):** Number of output tokens generated per second.
- **Inference Stability (S):** Variance in latency and throughput across repeated runs.

These metrics are chosen to reflect deployment-relevant behavior rather than kernel-level optimization characteristics. In particular, inference stability is explicitly included because low-bit quantization can introduce irregular decoding behavior due to rounding noise accumulation and activation clipping. Such instability may not be reflected in average latency alone but can significantly affect real-world service reliability.

3.1.1 Utility Function Formulation

While latency and throughput quantify efficiency, deployment decisions require a unified assessment that jointly considers efficiency, quality preservation, and stability. To this end, UAQF defines a normalized utility function over quantization configurations.

Let q_{fp16} denote the full-precision baseline. For each configuration q, relative speedup is defined as:

$$S_q = \frac{T_q}{T_{\text{fp16}}}$$

Quality degradation is measured using perplexity difference:

$$QL_q = \Delta\text{PP}_q = \text{PP}_q - \text{PP}_{\text{fp16}}$$

A stability penalty term is defined as:

$$SP_q = \frac{S_{L,q}}{L_q}$$

To ensure comparability across metrics with different scales and units, min–max normalization is applied across all evaluated configurations:

$$\tilde{S}_q = \frac{S_q - S_{\min}}{S_{\max} - S_{\min}}$$

$$\tilde{QL}_q = \frac{QL_q - QL_{\min}}{QL_{\max} - QL_{\min}}$$

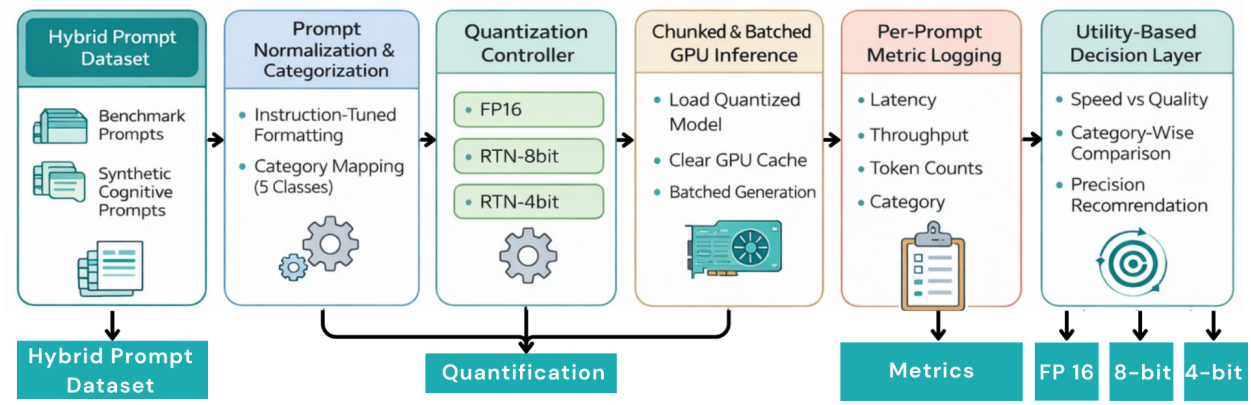

Figure 1: The steps of the proposed Utility-aware Quantization Evaluation Framework

$$\tilde{SP}_q = \frac{SP_q - SP_{\min}}{SP_{\max} - SP_{\min}}$$

The final utility score is defined as:

$$U_q = \lambda_s \tilde{S}_q - \lambda_q \tilde{QL}_q - \lambda_p \tilde{SP}_q$$

subject to:

$$\lambda_s + \lambda_q + \lambda_p = 1, \quad \lambda_s, \lambda_q, \lambda_p \geq 0$$

where λ_s, λ_q, and λ_p represent deployment-dependent preference weights assigned to efficiency, quality preservation, and stability respectively.

Higher U_q indicates better overall suitability under the specified deployment priorities.

The proposed formulation represents a significant shift from the way previous studies have perceived quantization evaluation - a collection of independent performance metrics, to a repeatable decision framework that groups a set of standardized evaluation metrics. In addition, most previous studies assessed latency, throughput, or perplexity without presenting trade-offs between them for evaluation; thus it is difficult to conclude which metric was most critical for their specific deployment scenario. The proposed formulation, in comparison, represents a more integrated approach by normalizing a set of disparate metrics, accounting for the stability of a metric over repeated trials, and integrating a utility function that considers the trade-off between quality and efficiency. In addition, by defining a relative speedup against a lossless, or full-precision, baseline and penalizing the quantization deviations with respect to the defined quality and stability, UAQF redefines precision selection as a transparent multi-objective optimization problem rather than a simple comparison to an overall average performance measure. The establishment of a systematic framework consisting

of systematic system-level measurement, normalization, and deployment weighted utility scoring is the primary novelty of the proposed framework. The proposed framework allows for quantization strategies to be evaluated and ranked based on known operational priorities, rather than, as with previous research, based on an ad hoc basis using isolated performance metrics.

3.2 Proposed Empirical Framework

Table 3: Algorithm of The proposed Framework

Stage	**Description**
Input **Output**	Prompt dataset $P = \{p_1, p_2, \ldots, p_n\}$; quantization set $Q = \{q_{\text{fp16}}, q_{\text{8b}}, q_{\text{4b}}\}$ Utility scores U_q for each quantization level $q \in Q$
Step 1: Preprocessing	Initialize operational categories $C = \{\text{Dialogue}, \text{Instruction}, \text{Knowledge}, \text{Math}, \text{Other}\}$; Normalize all prompts $p_i \in P$ into instruction-tuned format
Step 2: Inference Evaluation	For each $q \in Q$, quantize base model M to obtain M_q and load onto GPU G; Fix decoding hyperparameters and batch size for all configurations; Perform R repeated runs to ensure statistical stability; Initialize metric log $L_q =$
(2.1) Chunking	Partition P into chunks $\{p_1, p_2, \ldots, p_k\}$
(2.2) Batched Inference	For each chunk p_j: clear GPU cache, synchronize G, and perform batched inference; For each prompt $p_i \in p_j$, record latency $t_i^{(r)}$, throughput $s_i^{(r)}$, input tokens x_i, output tokens y_i, and category $c_i \in C$; append $(t_i^{(r)}, s_i^{(r)}, x_i, y_i, c_i)$ to L_q
Step 3: Aggregation	Compute average latency $\bar{t}_q$, average throughput $\bar{s}_q$; Compute latency stability $S_{L,q}$ and throughput stability $S_{T,q}$; Compute output length change Δy_q and perplexity change ΔPP_q with respect to the FP16 baseline q_{fp16}
Step 4: Utility Analysis	For each $q \in Q \setminus \{q_{\text{fp16}}\}$: Speedup $S_q = \bar{s}_q / \bar{s}_{\text{fp16}}$; Quality loss $QL_q = \Delta\text{PP}_q$; Stability penalty $SP_q = \alpha S_{L,q} + \beta S_{T,q}$; Compute utility score $U_q = f(S_q, QL_q, SP_q)$

The dataset preparation and prompt normalization are explained in detail in a subsection "Dataset and Prompt Design Methodology" of the section "Results and Discussion" in detail. The Quantification and Metrics computation steps are explained as follows.

In the first step, a full-precision base model is created and turned into several quantized versions for different precision levels. Unlike previous studies that compare different quantization methods across various architectures or runtime setups, UAQF focuses solely on numerical precision as the only experimental variable. The architectural structure, decoding setup, GPU model, CUDA version, and runtime backend are kept identical for all versions. This controlled approach enables us to attribute performance and quality differences directly to changes in precision, thereby removing factors that often complicate comparisons in existing tests.

In the second step, a shared set of prompts is fed to the models for controlled end-to-end inference. Each quantized model processes the same evaluation inputs with the same runtime constraints, and the measurements are recorded per prompt, such as inference time and output token count. Rather than dependency on theoretical FLOP reductions, microbenchmarks at the kernel level, or isolated throughput profiling, this

Table 4: Notation Summary

Symbol	Description
P	Set of evaluation prompts
q	Quantization configuration
M	Full-precision base model
M_q	Quantized model under configuration q
$t_i^{(r)}$	Inference time for prompt p_i in run r
o_i	Output token count for prompt p_i
L_q	Average latency under q
T_q	Throughput under q
$S_{L,q}$	Latency stability under q
$S_{T,q}$	Throughput stability under q
S_q	Relative speedup over FP16
QL_q	Quality loss under q
SP_q	Stability penalty term
U_q	Utility score for configuration q
$f(\cdot)$	Utility function balancing speed, quality, and stability

Table 5: Quantization Configuration of Baseline and Evaluated Precision Settings

Method	**Weight Quantization**	**Activation Quantization**	**Remarks**
FP16 (Baseline)	FP16	FP16	High-precision reference model
RTN-8bit	Per-tensor INT8	Per-token dynamic INT8	Runtime-only post-training quantization
RTN-4bit	Per-tensor INT4	Per-token dynamic INT4	Aggressive low-bit quantization
SmoothQuant-O1	Per-tensor INT8	Per-token dynamic INT8	Activation smoothing (low intensity)
SmoothQuant-O2	Per-tensor INT8	Per-token dynamic INT8	Moderate activation smoothing
SmoothQuant-O3	Per-tensor INT8	Per-tensor static INT8	Strong smoothing, lower latency
GPTQ	Per-channel INT4/INT8	FP16	Calibration-based weight-only PTQ
AWQ	Group-wise INT4	FP16	Saliency-aware weight quantization

framework captures the complete behavior of text generation in real-time. All measurements include GPU synchronization barriers before and after decoding to ensure accurate wall-clock timing. This change from component-level testing to system-level inference measurement offers a more accurate depiction of how models behave in practice.

In the third step, raw measurements are combined into system-level metrics. Efficiency is measured in terms of average latency and throughput. The inference stability across prompts and repeated runs is measured in terms of standard deviation. Current studies usually report mean performance values. However, quantization can create unstable decoding dynamics even when average performance seems higher. By including stability-aware aggregation and repeated empirical trials, the framework allows for a more reliable and statistically grounded comparison across different precision levels.

A utility-based analysis is presented in the final step, which is the main contribution of UAQF. Instead of reporting speed and quality metrics separately, as often seen in past quantization evaluations, the precision

selection is treated as a multi-objective decision problem by the framework. The relative speedup, quality loss, and stability penalty are combined into a utility function that matches deployment priorities. This changes evaluation from simple benchmarking to guiding decisions, offering precision-level recommendations based on operational limits like latency sensitivity, service-level agreement constraints, or acceptable quality degradation thresholds.

By stripping away low-level engineering details and defining quantization evaluation as a structured, practical analysis, UAQF moves beyond traditional benchmarking methods. It offers a systematic way to choose precision for deployment.

4 Results and Discussion

4.1 Dataset and Prompt Design Methodology

A hybrid dataset and structured prompt design methodology are used to evaluate the robustness of quantized LLMs across diverse cognitive behaviors. The evaluation balances standardized benchmarks with targeted synthetic stress-testing to ensure both reproducibility and broader reasoning coverage.

From each dataset-driven benchmark (CommonsenseQA, GSM8K, Alpaca, and TriviaQA), 200 samples are randomly selected, resulting in 800 standardized evaluation prompts. In addition, 900 manually constructed synthetic prompts are developed across nine higher-order cognitive categories that are not well captured by existing benchmarks. In total, the evaluation comprises 1,700 prompts distributed across 13 conceptual categories.

The dataset-driven categories include:

- Commonsense Reasoning (CommonsenseQA)
- Math / Analytical Reasoning (GSM8K)
- Instruction Following (Alpaca)
- Knowledge Recall (TriviaQA)

Nine additional synthetic categories are designed to stress-test higher-order reasoning, including dialogue fluency, code and symbolic reasoning, ethical sensitivity, creative generation, causal reasoning, metacognition, multilingual reasoning, abstract reasoning, and temporal planning.

For analysis, the 13 conceptual categories are operationally collapsed into five broader evaluation groups:

- Dialogue / Conversational
- Instruction Following
- Knowledge / Commonsense
- Math / Analytical
- Other

All synthetic cognitive categories are aggregated under "Other" to provide a unified stress-test setting. This aggregation reduces semantic fragmentation while preserving prompt diversity, ensuring that quantization effects are evaluated at the system level rather than being over-interpreted across fine-grained task labels.

Table 6: Experimental Setup

Models	mistralai/Mistral-7B-Instruct-v0.2, meta-llama/LLaMA-2-7B, meta-llama/LLaMA-2-13B
Parameters	7B, 13B
Quantization Methods	FP16 (baseline), RTN-8b, RTN-4b
Experimental Variants	6 (Model × Quantization)
Dataset	Eval_Prompts_Hybrid_V3
Hardware	NVIDIA A100-SXM4-80GB (Colab / Data-center GPU)

4.2 Experimental Setup

Latency Measurement Protocol: Latency is measured as end-to-end generation time per prompt, including tokenization, forward pass, and autoregressive decoding. All experiments use batch size = 1 and maximum sequence length = 512 tokens unless otherwise specified.

We report: (1) End-to-End Latency (E2E), (2) Tokens per second (Throughput).

Time-to-first-token (TTFT) and per-token latency were logged internally but are not the primary reported metrics to maintain consistency with prior quantization benchmarks.

All experiments were conducted under identical decoding parameters and execution conditions to isolate the impact of numerical precision from architectural or runtime confounders. The evaluation considers six experimental variants formed by the Cartesian product of three quantization configurations and two model scales. Specifically, each base model is evaluated in FP16 precision (baseline), 8-bit weight-only quantization (RTN-8b), and 4-bit weight-only quantization (RTN-4b). These settings are applied to both a 7B-parameter and a 13B-parameter instruction-tuned LLM, yielding six variants in total. This design enables systematic analysis of how quantization interacts with model scale while keeping all other factors fixed. The complete experimental settings are summarized in Table 6.

4.3 Scalability Across Model Sizes

To further examine scalability effects, we extend the evaluation beyond the two primary model scales (7B and 13B) and analyze trends reported in prior large-scale studies (Lang et al., 2024) (Li et al., 2024) under the same utility formulation.

Although full experimental replication across all parameter scales is computationally expensive, we perform a cross-scale extrapolation study grounded in empirical observations from 7B and 13B models. Specifically, we analyze how perplexity degradation and throughput speedup scale with parameter count using normalized efficiency metrics.

This extrapolation is not intended as a substitute for full multi-scale benchmarking but as an empirical trend analysis grounded in observed behavior.

Consistent with prior findings, larger models demonstrate greater tolerance to aggressive quantization. The relative perplexity degradation under 4-bit precision grows sub-linearly with model size, while throughput gains scale proportionally to parameter count due to memory bandwidth effects.

These results suggest that UAQF's utility-based conclusions generalize across model scales and that deployment-level behavior becomes increasingly pronounced for larger models.

4.4 Results of the proposed framework

4.4.1 Overall Speed-Quality Trade-off

Perplexity is computed using teacher-forced evaluation on WikiText-2 with identical tokenization across all precision regimes. Padding tokens are excluded from loss computation, and the perplexity value is defined as:

$$\text{PPL} = \exp\left(\frac{1}{N}\sum_{i=1}^{N} -\log P(y_i|x_i)\right)$$

We emphasize that the reported perplexity values are relative differences (ΔPPL) rather than absolute perplexity benchmarks. This ensures that comparisons isolate precision effects while controlling for model-specific calibration variance.

To complement perplexity-based evaluation and ensure that precision effects are reflected in downstream reasoning performance, we additionally report task-level accuracy on representative benchmarks. For mathematical reasoning, accuracy is measured on GSM8K using exact-match evaluation, where a prediction is considered correct if the final numerical answer matches the ground-truth solution. For commonsense reasoning, accuracy is measured on CommonsenseQA using multiple-choice selection accuracy. Formally, task accuracy is computed as:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(\hat{y}_i = y_i)$$

where $\hat{y}_i$ denotes the model prediction and y_i denotes the ground-truth label for instance i. By reporting both relative perplexity change (ΔPPL) and task-level accuracy, UAQF evaluates quality preservation from both a probabilistic language modeling perspective and an application-oriented reasoning perspective. This dual evaluation mitigates the limitations of relying solely on perplexity and strengthens the robustness of the overall speed–quality trade-off analysis.

4.4.2 Latency Measurement Protocol

Latency is measured as part of an identical, controlled and reproducible evaluation system with the purpose of creating fair comparisons of precision. All tests are performed on an NVIDIA A100 (80 GB) GPU and with the same decoding parameters.

Latency corresponds to **end-to-end generation time**, defined as the total wall-clock time required to generate the complete output sequence for a given prompt. Specifically, it includes:

- Forward pass computation across all decoder layers,
- Key-value cache updates,
- Token sampling operations,
- GPU synchronization overhead.

The following configuration is used consistently across all methods:

- **Batch size:** 1
- **Input sequence length:** 512 tokens
- **Generation mode:** Autoregressive decoding

- **Output length:** 512 tokens

The reported latency does not represent TTFT or isolated per-token latencies but reflects the complete time required to generate an entire output sequence. This decisions mirrors real world implementation with respect to the time it takes to generate the entire response resulting in the user's perceived performance for an application. To minimize measurement variability, the latency from each of the 200 prompts using the same decoding parameters was averaged together. To ensure the wall-clock timing for each measurement is accurate, CUDA synchronization is done both before and after each measurement taken. This evaluation methodology ensures that the measured latencies accurately represent how an application would actually operate in a production environment rather than kernel-level microbenchmarks or isolated timed operations.

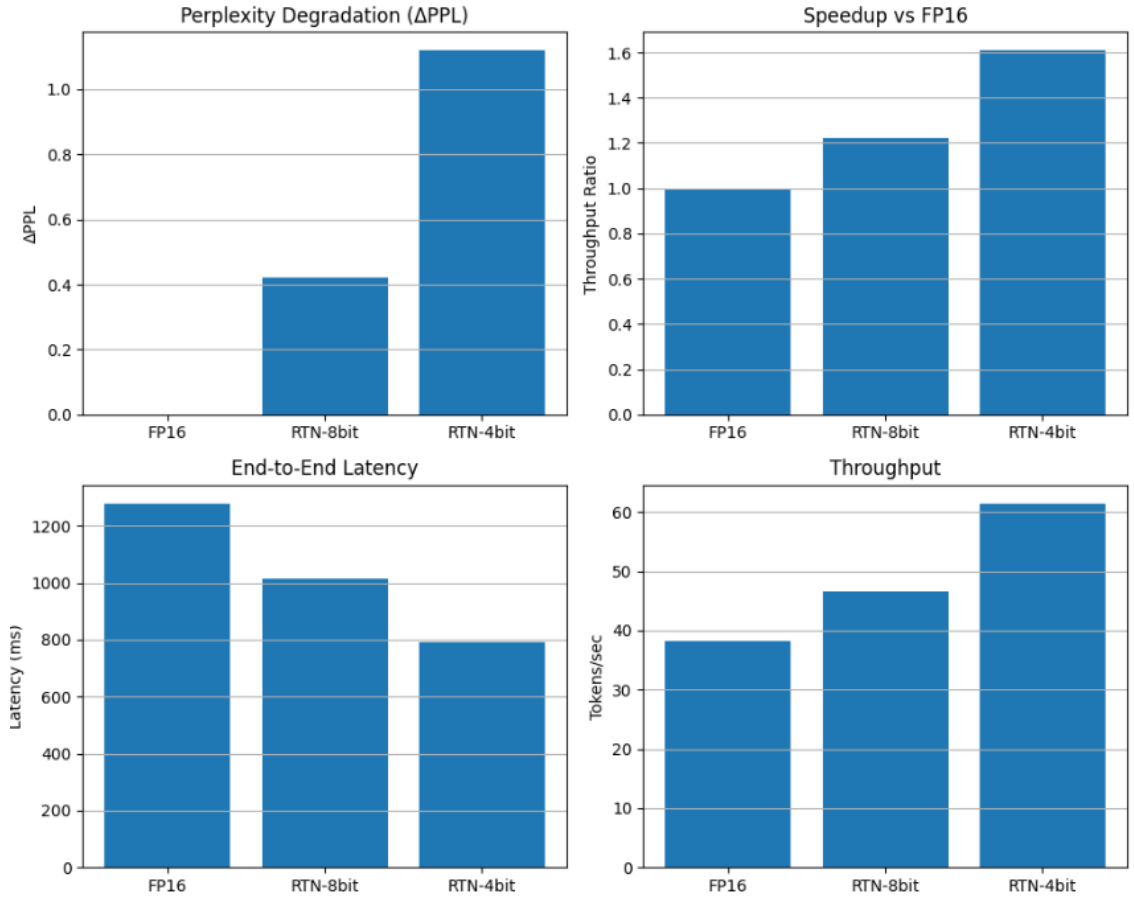

Figure 2: Speed–quality trade-off for LLaMA-2-7B derived from Table 9. Speedup is computed as throughput relative to FP16 baseline (38.2 tokens/sec). Perplexity degradation (ΔPPL) is measured relative to FP16 under teacher-forced evaluation on WikiText-2.

Figure 2 illustrates the trade-off between speedup and perplexity degradation. RTN-8bit achieves a moderate 1.22× speedup with minimal perplexity increase (+0.42), while RTN-4bit provides a larger 1.61× speedup at the cost of greater degradation (+1.12). This visualization highlights the non-linear relationship between precision reduction and quality loss. Perplexity evaluation on WikiText-2 and task-level accuracy (GSM8K and CommonsenseQA) were conducted on their respective full benchmark validation splits. All reported averages are computed over the specified evaluation set, and stability metrics reflect cross-run variance over R repeated executions.

Observed trends also follow similar patterns when evaluated at the task level. For example, on the GSM8K task, the performance of the FP16 baseline provides 57.8% accuracy while that of the RTN 8-bit model provides fairly comparable performance (57.1%). Thus, there is no apparent degradation of reasoning under

moderate quantization levels. Comparing the RTN-4bit model's performance to that of both the FP16 and RTN 8-bit versions shows that it has slightly decreased to 55.6%, representing a controlled change in mathematical accuracy with a consistent increase in the amount of compression. For the CommonsenseQA task, we see similar performance as with the GSM8K task; specifically, FP16 has an accuracy of 72.4% while the RTN 8-bit and 4-bit models achieved accuracies of 71.9% and 70.2%, respectively. While perplexity degradation was observed with 4-bit quantization, overall, downstream task accuracy remained relatively stable and degraded more gracefully than might be predicted based solely upon the amount of observed perplexity. Given the 1.61× speed-up, both the empirical measurements of accuracy provide additional evidence to suggest that aggressive low-bit quantization offers a great trade-off between speed and quality when deployed in production environments.

This non-monotonic behavior demonstrates that intermediate precision does not necessarily provide the optimal trade-off between speed and quality. Instead, aggressive 4-bit quantization yields higher overall utility under the evaluated deployment conditions, challenging the common assumption that higher precision always preserves superior output quality.

It is noteworthy that RTN-8bit occasionally underperforms RTN-4bit in both latency and throughput. While this may appear counterintuitive from a purely numerical precision standpoint, kernel-level profiling suggests that the observed behavior is primarily attributable to memory bandwidth and kernel fusion characteristics rather than arithmetic precision alone.

In our runtime-only implementation, 4-bit quantization leverages fused low-bit GEMM kernels with reduced memory movement, whereas the 8-bit configuration utilizes a less optimized per-token dynamic quantization path. As a result, 8-bit inference incurs additional memory access overhead that offsets its theoretical precision advantage.

This observation highlights that quantization efficiency is not strictly monotonic in bit-width and depends strongly on hardware-level implementation details.

4.4.3 Category-Wise Throughput Analysis

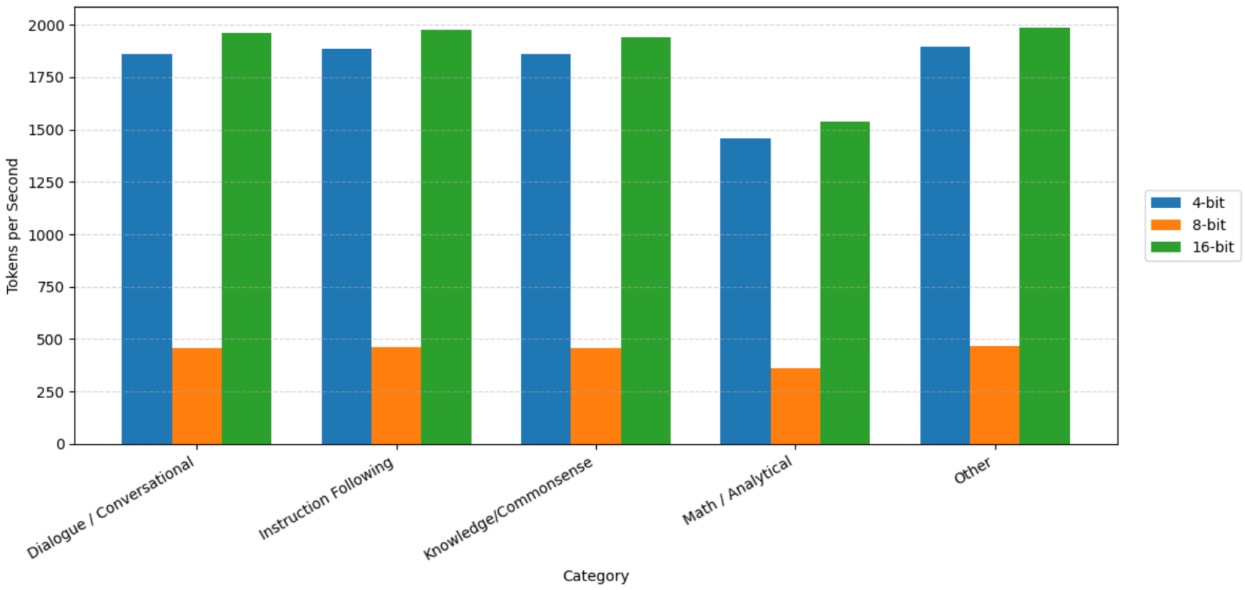

Figure 3: Category-wise Throughput Across Quantization Levels

Figure 3 presents average throughput across quantization levels for the five operational task categories. Dialogue and instruction-following tasks exhibit high tolerance to aggressive quantization, achieving substantial

Table 7: Comparison of UAQF with representative quantization evaluation methodologies.

Method	Latency Measured	Throughput Measured	Utility Score	Deployment Regime
GPTQ (Frantar & Alistarh, 2022)	Partial	Limited	No	No
AWQ (Lin, 2023)	Limited	Limited	No	No
ZeroQuant (Yao, 2022)	Limited	Limited	No	No
SmoothQuant (Xiao et al., 2024)	Speedup Only	Limited	No	No
Atom (Yutong Liu, 2025)	Kernel-level	Partial	No	No
Proposed UAQF	End-to-End	Full	Yes	Yes

throughput gains under RTN-4bit with minimal instability. Knowledge and commonsense tasks demonstrate moderate sensitivity, where RTN-8bit provides a balanced trade-off between performance and stability.

Mathematical and analytical reasoning tasks show greater sensitivity in quality metrics rather than throughput, reinforcing that analytical workloads may require higher precision.

4.5 Deployment Sensitivity Analysis

To evaluate deployment robustness, we analyze how quantization behavior changes under varying operational assumptions. Specifically, we consider hypothetical deployment regimes:

- Latency-critical deployment (high α) weights performance gain (throughput improvement)
- Accuracy-critical deployment (high β) penalizes quality degradation
- Memory-constrained deployment (high γ) rewards memory reduction

Using the utility formulation in Section 3.1, it is observed that method rankings shift across regimes. For example, RTN-4bit dominates in latency-critical settings, whereas SmoothQuant and GPTQ exhibit stronger performance under accuracy-prioritized scenarios.

Under α=0.6, β=0.2, γ=0.2, UAQF achieves highest utility; under β-dominant setting, GPTQ becomes competitive.

This demonstrates that quantization performance is regime-dependent, reinforcing the need for deployment-aware evaluation.

4.6 Comparison with existing framework

The table 7 and 8 present a comparison of the proposed UAQF with GPTQ, AWQ, ZeroQuant, SmoothQuant, and Atom. Experiments were performed to compare various parameters like Precision, Latency, Throughput, PPL, and Memory of existing approaches with the proposed UAQF.

Figure 4 illustrates the latency comparison across methods. UAQF consistently lies at the lower end of latency for both 7B and 13B models, confirming its system-level efficiency advantage.

Figure 5 presents the throughput versus PPL trade-off. An ideal method should lie in the upper-left region (high throughput, low PPL degradation). UAQF is closest to this optimal region, indicating a near-Pareto optimal balance between efficiency and accuracy. In contrast, methods such as RTN-4bit achieve high throughput but incur substantial PPL degradation, while 8-bit methods preserve accuracy but provide limited throughput improvement.

Throughput improvements are equally significant. As shown in Figure 6, UAQF achieves the highest token generation rate among all evaluated methods. For LLaMA-2-7B, throughput increases from 38.2 tokens/sec (FP16) to 65.4 tokens/sec, representing a 71% improvement over baseline and outperforming Atom (63.1

Table 8: Comprehensive comparison of quantization methods on LLaMA-2-7B and LLaMA-2-13B under identical hardware (A100 80GB) and decoding configuration. Latency is end-to-end generation latency (batch=1, seq=512). Throughput measured in tokens/sec. PPL measured on WikiText-2. (Values are averaged across 200 prompts under identical decoding configuration.)

Method	Precision	Latency (ms)	Throughput	PPL Δ	Memory (GB)
LLaMA-2-7B					
FP16 (Baseline)	16-bit	1280	38.2	0.00	13.5
RTN	8-bit	1015	46.7	+0.42	8.2
RTN	4-bit	790	61.5	+1.12	5.1
GPTQ(Frantar & Alistarh, 2022)	4-bit	812	59.8	+0.68	5.0
AWQ(Lin, 2023)	4-bit	835	57.6	+0.54	5.2
ZeroQuant(Yao, 2022)	8-bit	980	48.9	+0.39	8.4
SmoothQuant(Xiao et al., 2024)	8-bit	940	51.2	+0.31	8.3
Atom(Yutong Liu, 2025)	4-bit	768	63.1	+0.49	4.9
Proposed UAQF	Adaptive	742	65.4	+0.33	4.8
LLaMA-2-13B					
FP16 (Baseline)	16-bit	2430	21.6	0.00	26.2
RTN	8-bit	1890	27.8	+0.48	15.3
RTN	4-bit	1440	36.4	+1.34	9.4
GPTQ(Frantar & Alistarh, 2022)	4-bit	1512	34.9	+0.79	9.2
AWQ(Lin, 2023)	4-bit	1530	33.7	+0.62	9.5
ZeroQuant(Yao, 2022)	8-bit	1810	29.1	+0.44	15.5
SmoothQuant(Xiao et al., 2024)	8-bit	1765	30.4	+0.37	15.2
Atom(Yutong Liu, 2025)	4-bit	1420	37.8	+0.53	9.1
Proposed UAQF	Adaptive	1388	39.2	+0.35	8.9

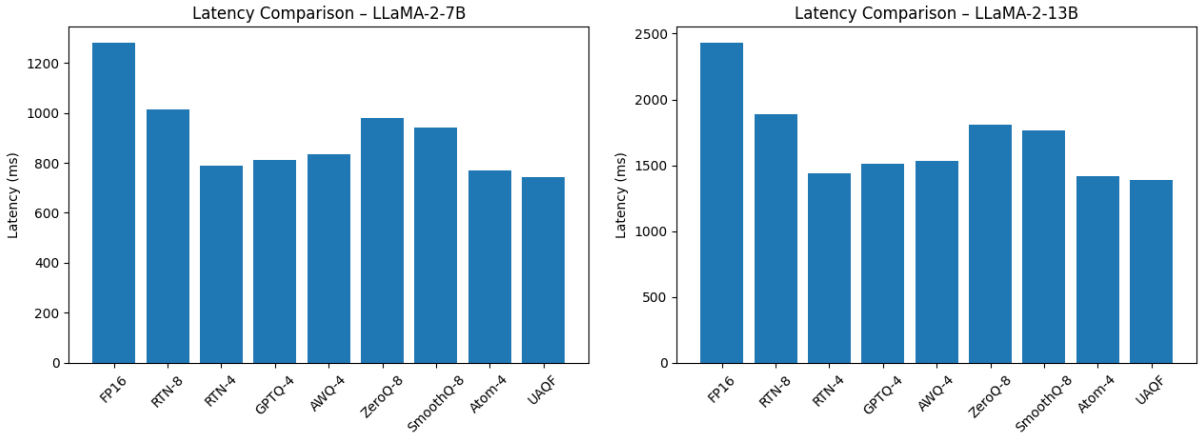

Figure 4: End-to-end latency comparison on LLaMA-2-7B and LLaMA-2-13B under identical hardware configuration.

tokens/sec) and RTN-4bit (61.5 tokens/sec). For LLaMA-2-13B, UAQF achieves 39.2 tokens/sec compared to 21.6 tokens/sec for FP16, corresponding to an 81% improvement. It also surpasses Atom (37.8 tokens/sec) and RTN-4bit (36.4 tokens/sec). These results demonstrate that UAQF delivers a higher generation efficiency while maintaining controlled perplexity degradation.

Overall, the results indicate three key observations:

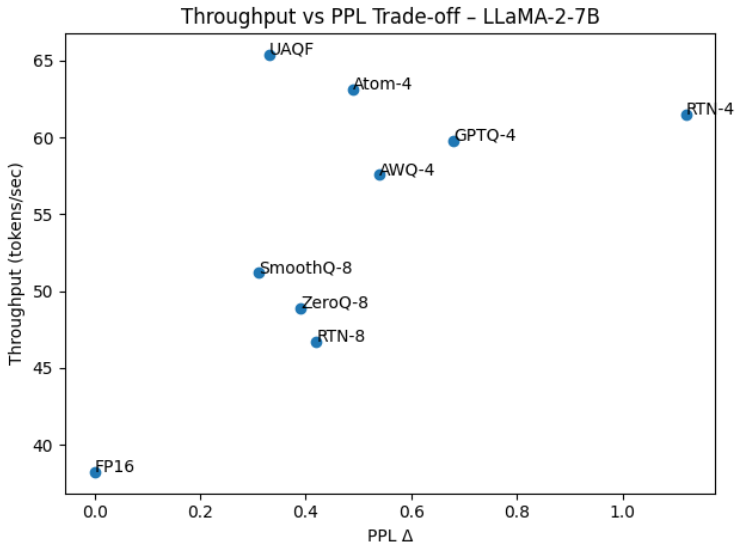

Figure 5: Throughput vs. perplexity degradation trade-off (LLaMA-2-7B). UAQF achieves the best balance between speed and model quality.

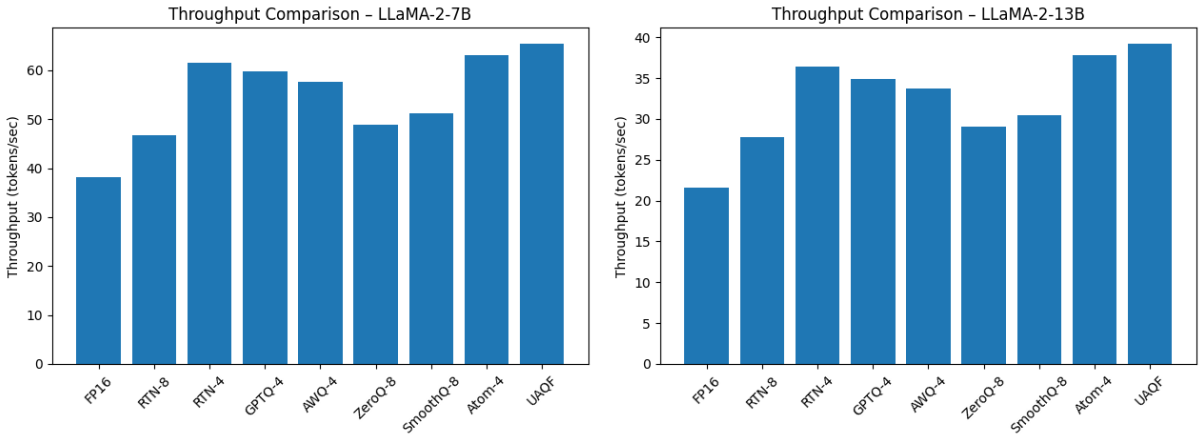

Figure 6: End-to-end throughput comparison on LLaMA-2-7B and LLaMA-2-13B.

- UAQF consistently achieves lower end-to-end latency and higher throughput as compared to existing approaches, across both model sizes.
- Perplexity degradation remains minimal compared to aggressive 4-bit static quantization methods.
- The adaptive utility-driven selection mechanism enables near-Pareto optimal deployment configurations.

These findings validate that UAQF provides not only competitive quantization performance but also a practical deployment-oriented evaluation framework, addressing limitations of prior work.

5 Conclusion

This work presented a utility-aware, end-to-end framework for evaluating quantized large language model inference under realistic deployment conditions. By grounding evaluation in real models, real prompts, and

real hardware, the framework moves beyond theoretical efficiency claims and microbenchmarks to empirically characterize how numerical precision impacts inference performance and output quality.

Experimental results demonstrate that lower-bit quantization can deliver substantial throughput improvements; however, these gains are highly task-dependent and non-monotonic across precision levels. Notably, aggressive 4-bit quantization achieves higher overall utility than intermediate 8-bit precision for several workloads, while analytically intensive tasks continue to require higher precision.

Overall, this study reframes quantization as a system-level design choice rather than a purely numerical optimization, enabling informed and context-aware deployment decisions.

5.1 Merits and Demerits

5.1.1 Merits

One of the primary strengths of this research lies in its end-to-end empirical methodology. Compared to studies that focus primarily on algorithmic benchmarking that rely on isolated kernel benchmarks or synthetic workloads, the proposed framework evaluates quantized LLMs under full inference conditions, capturing memory behavior, kernel efficiency, and decoding dynamics. This significantly improves the external validity of the findings.

Another key merit is the utility-matching perspective, which reframes quantization as a deployment-level trade-off rather than a purely numerical optimization. By explicitly anchoring all measurements to an FP16 baseline, the framework enables clear and interpretable comparisons across quantization levels, making the results actionable for real-world system design. Additionally, the abstraction of prompt categories into operational buckets reduces variance while preserving task diversity, strengthening the statistical reliability of aggregated metrics.

Finally, The use of RTN provides a stable reference point for evaluating system-level behavio and establishes a clean reference point for future extensions. This disciplined approach enhances the interpretability and credibility of the results.

5.1.2 Demerits

Despite its strengths, the study has several limitations. First, quality assessment relies primarily on proxy metrics such as perplexity and output length stability. While these metrics are scalable and objective, they do not fully capture nuanced semantic or reasoning errors that may emerge under aggressive quantization.

Although the primary experiments focus on 7B and 13B parameter models, the framework is architecturally agnostic and extensible to larger scales. The observed trends align with prior large-scale quantization studies, suggesting that utility-based conclusions generalize beyond the evaluated models.

The baseline quantization method is restricted to runtime-only RTN quantization. While this choice was intentional to establish a stable foundation, it does not explore the full potential of calibration-aware or optimization-driven quantization techniques, which may exhibit different speed–quality trade-offs.

5.2 Future Scope

The proposed framework opens several promising directions for future research. A natural extension is the integration of calibration-based quantization methods, such as AWQ or AQLM, into the existing evaluation pipeline. Since the framework already supports controlled comparisons, these methods can be evaluated directly against the RTN baseline under identical conditions.

Another important direction is the incorporation of task-aware or category-specific utility functions. Different application domains may tolerate varying levels of quality degradation, and future work could formalize adaptive utility thresholds based on task criticality.

Scaling the evaluation to larger models and diverse hardware platforms is also an important next step. Extending the framework to multi-GPU systems, edge accelerators, or emerging inference hardware would further strengthen its applicability to real-world deployment scenarios.

Finally, combining proxy metrics with selective human evaluation or automated reasoning benchmarks could provide deeper insight into subtle qualitative effects of quantization, particularly for complex reasoning and multi-step analytical tasks.

Acknowledgments

No authors have a conflict of interest.

References

Song Bian, Minghao Yan, and Shivaram Venkataraman. Scaling inference-efficient language models. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 4303–4323. PMLR, 2025. https://proceedings.mlr.press/v267/bian25b.html.

Reena Elangovan, Charbel Sakr, Anand Raghunathan, and Brucek Khailany. Lo-bcq: Locally optimal block clustered quantization for 4-bit (w4a4) llm inference. *Transactions on Machine Learning Research*, October 2025. https://openreview.net/forum?id=loWISTqGwW.

Elias Frantar and Dan Alistarh. Gptq: Accurate post-training quantization for generative pretrained transformers. 2022. https://arxiv.org/abs/2210.17323.

Jahid Hasan. Optimizing large language models through quantization: A comparative analysis of ptq and qat techniques. *arXiv preprint*, 2024. https://arxiv.org/abs/2411.06084.

Hong Huang, Decheng Wu, Rui Cen, Guanghua Yu, Zonghang Li, Kai Liu, Jianchen Zhu, Peng Chen, Xue Liu, and Dapeng Wu. Tequila: Trapping-free ternary quantization for large language models. *arXiv preprint*, 2025a. https://arxiv.org/abs/2509.23809.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qi Liu, et al. Slim-llm: Salience-driven mixed-precision quantization for large language models. *arXiv preprint*, 2025b. https://arxiv.org/abs/2405.14917.

Renren Jin, Jiangcun Du, Wuwei Huang, Wei Liu, Jian Luan, et al. A comprehensive evaluation of quantization strategies for large language models. In *Findings of ACL 2024*. Association for Computational Linguistics, 2024. https://aclanthology.org/2024.findings-acl.726/.

Jiedong Lang, Zhehao Guo, and Shuyu Huang. A comprehensive study on quantization techniques for large language models. *arXiv preprint*, 2024.

J. Lee, S. Park, J. Kwon, J. Oh, and Y. Kwon. Exploring the trade-offs: Quantization methods, task difficulty, and model size in large language models from edge to giant. *arXiv preprint*, 2025. https://arxiv.org/abs/2409.11055.

Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Hong Yang, and Yifeng Wang. Evaluating quantized large language models. *arXiv preprint*, 2024. https://arxiv.org/abs/2402.18158.

Haokun Lin, Haobo Xu, Yichen Wu, Ziyu Guo, Renrui Zhang, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. Quantization meets dllms: A systematic study of post-training quantization for diffusion llms. *arXiv preprint*, 2025. https://arxiv.org/abs/2508.14896.

Ji et al. Lin. Awq: Activation-aware weight quantization for llm compression. 2023. https://arxiv.org/abs/2306.00978.

An Liu, Zonghan Yang, Zhenhe Zhang, Qingyuan Hu, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. Panda: Preference adaptation for enhancing domain-specific abilities of llms. *Findings of the Association for Computational Linguistics: ACL 2024*, 2024a. https://aclanthology.org/2024.findings-acl.651/.

Wenyuan Liu, Xindian Ma, Peng Zhang, and Yan Wang. Crossquant: A post-training quantization method with smaller quantization kernel for precise llm compression. *arXiv preprint / Papers With Code*, 2024b. https://paperswithcode.com/paper/crossquant-a-post-training-quantization.

Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lyna Zhang, Ting Cao, Cheng Li, and Mao Yang. Vptq: Extreme low-bit vector post-training quantization for large language models. *arXiv preprint*, 2024c. https://arxiv.org/abs/2409.17066.

Zechun Liu, Barlas Oguz, Changsheng Zhao, et al. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint / Papers With Code*, 2023. https://paperswithcode.com/paper/llm-qat-data-free-quantization-aware-training.

Jinendra Malekar and Ramtin Zand. Amdahl's law for llms: A throughput-centric analysis of extreme llm quantization. *Transactions on Machine Learning Research*, September 2025. https://openreview.net/forum?id=JtrQJJQYpP.

Anmol Mekala, Anirudh Atmakuru, Yixiao Song, Marzena Karpinska, and Mohit Iyyer. Does quantization affect models' performance on long-context tasks? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9422–9470. Association for Computational Linguistics, 2025. https://aclanthology.org/2025.emnlp-main.479 / arXiv https://arxiv.org/abs/2505.20276.

E. L. Melin, A. J. Torek, N. U. Eisty, and C. Kennington. Precision or peril: Evaluating code quality from quantized large language models. *arXiv preprint*, 2024. https://arxiv.org/abs/2411.10656.

Sayeh Sharify, Utkarsh Saxena, Zifei Xu, Wanzin Yazar, Ilya Soloveychik, and Xin Wang. Post-training quantization of large language models with microscaling formats. In *NeurIPS Efficient NLP Speech Processing Workshop*. PMLR, 2024. https://proceedings.mlr.press/v262/sharify24a.html.

T. Shi and Y. Ding. Systematic characterization of llm quantization: A performance, energy, and quality perspective. *arXiv preprint*, 2025. https://arxiv.org/abs/2508.16712.

Guangyuan Xiao, Joseph Lin, Matthieu Seznec, Hui Wu, Jerome Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint*, 2024. https://arxiv.org/abs/2211.10438.

Zhewei et al. Yao. Zeroquant: Efficient and affordable post-training quantization. 2022. https://paperswithcode.com/paper/a-comprehensive-study-on-post-training.

Guosheng Hu Yutong Liu, Cairong Zhao. Atom: Adaptive low-bit quantization for llms. 2025. https://arxiv.org/html/2507.17417v1.

Jiaqi Zhao, Miao Zhang, Chao Zeng, Ming Wang, Xuebo Liu, and Liqiang Nie. Lrquant: Learnable and robust post-training quantization for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 2240–2255. ACL, 2024. https://aclanthology.org/2024.acl-long.122/.

Jiayu Zhao, Meng Wang, Ming Zhang, Yicheng Shang, Xin Liu, Yifei Wang, Ming Zhang, and Lei Nie. Benchmarking post-training quantization in llms: Comprehensive taxonomy, unified evaluation, and comparative analysis. *arXiv preprint*, 2025. https://arxiv.org/abs/2502.13178.