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, as memory consumption and cost of computation become large factors for using LLM effectively, we will experience the challenge of achieving efficient inference with LLMs in the absence of adequate resources. Quantization, which refers to reducing numerical precision during the inference stage of the model to reduce memory usage, is a possible way to address these issues. Through quantization, model memory usage and cost efficiency will 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, this study presents a utility-aware, end-to-end evaluation framework for LLM Inference using quantization, where the framework operates on actual models, actual prompts, and actual hardware, capturing for each combination the latency, throughput, and the resulting impact on the quality of the response when quantized in various levels of precision. This study illustrates the usefulness and trade-off between latency and quality of throughput in LLM Inference by utilizing the framework to assess its own trial models against a set benchmark high-precision evaluation. The proposed UAQF (Utility-Aware Quantization Framework) utilizes a modern instruction-tuned LLM (Mistral-7B-Instruct-v0.2) and has been tested on FP16, 8-bit, and 4-bit quantization. The experimental results suggest that lower bit quantization increases throughput dramatically, with little or no effect on the ability to generate high-quality LLM outputs. The study further demonstrates that aggressive quantization of LLMs yields significantly greater overall utility than intermediate precision quantizations, underscoring the need for empirical and deployment-oriented methods for 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 best performance from LLM models. 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.

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 reduce the efficiency of large language models (LLMs), while maintaining the quality of large language models (LLMs) to the same degree. They discuss important considerations regarding trade-offs between the performance of LLMs and the speed of their continued development.

The previous quantitative studies have focused on how quantization impacts different classes of tasks and behaviours 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-based and smoothness-based approaches as effective methods for minimising the potential loss of accuracy resulting from quantization, and 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 standardised 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 real models, real prompts, and real hardware. This framework will allow for the 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 Our Contributions

The key contributions of this work are summarized as follows:

- An End-to-End Evaluation Pipeline for Quantizing LLMs: In this work, we have proposed a systematic, repeatable approach to evaluating the inference performance of quantised LLMs in terms of their use on real workloads and hardware. This approach avoids relying on synthetic benchmarks, which may not accurately represent quantised LLMs' performance under real-world usage conditions.
- Utility-based Comparison of Quantization Trade-offs: We have developed a utility-based analysis framework that allows us to evaluate the relative trade-offs between speed and quality degradation across a wide range of quantization levels, thus allowing for direct comparison between quantization levels and architectures.
- Characterising Low-Bit Inference Empirically: We conducted a series of controlled experiments that demonstrated the effects of different levels (FP16, 8-bit, and 4-bit) of quantization on latency, throughput, and output behaviour across a number of prompt categories.
- Engineering-driven Experimental Design: We built a resumable and chunked inference engine that supports deterministic decoding and structured logging, facilitating stability and repeatability under constrained conditions.

1.4 Research Objectives

This research is guided by the following objectives:

- To empirically measure the impact of different quantization levels on LLM inference latency and throughput under realistic conditions.
- To assess the extent to which low-bit quantization affects output quality using scalable proxy metrics.
- To establish a principled framework for comparing quantization strategies through a unified utility-based formulation.
- To provide deployment-relevant insights that inform precision selection in practical LLM systems.

1.5 Paper Organization

The structure of this paper is as follows: Section 3 reviews previous works related to LLM quantisation and inference optimization. Section 4 details the steps involved in the proposed UAQF (Utility-Aware Quantization Framework) with the algorithm. Section 5 discusses dataset construction, the experimental setup, the empirical results, and the comparative analysis. Section 6 concludes the work and discusses the benefits and pitfalls of our work along with the potential for future research in this area, followed by some closing comments.

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 Tssk 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. For example, the technique called SmoothQuant (Xiao et al., 2024) allowed for the near-FP16 accuracy at 8-Bit precision for various models through the combination of activation smoothing with Weight Quantization

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

Paper	Core Research Focus	Methods Used	Key Findings / Results	Main Limitations (Gap)
(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	No full end-to-end application benchmarks or user-level latency
(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	Evaluated mainly on kernel-level performance, limited system-level analysis
(Huang et al., 2025b)	Mixed-precision quantization	Salience detection with mixed-bit allocation	Mixed precision outperforms uniform low-bit quantization	Increased complexity; lacks real-world performance metrics
(Mekala et al., 2025)	Effect of quantization on long-context reasoning	PTQ across long-context benchmarks	Long-context tasks are more sensitive to quantization	No wall-clock latency or hardware-level analysis
(Shi & Ding, 2025)	Performance–energy–quality trade-offs	Controlled experiments on A100/H100 GPUs	Lower precision reduces energy; quality loss varies by task	Limited reproducibility outside controlled infrastructure
(Zhao et al., 2025)	Taxonomy and comparison of PTQ methods	Unified benchmarking framework	No universally optimal PTQ method	Focuses on algorithm ranking, not deployment realism
(Lang et al., 2024)	Large-scale quality evaluation	GPTQ, AWQ, FP8 across 13 benchmarks	Larger models tolerate quantization better	Ignores inference speed, energy, and kernel behavior
(Melin et al., 2024)	Quantization impact on code generation	Code benchmarks with human evaluation	Code generation highly sensitive to low-bit precision	Narrow task focus; no general inference metrics
(Xiao et al., 2024)	Accurate INT8 post-training quantization	Activation smoothing with weight quantization	Near-FP16 accuracy achievable for many models	Focuses on method design, not deployment utility
(Liu et al., 2024b)	Extreme low-bit PTQ	Vector-based post-training quantization	Maintains accuracy at very low bit-widths	No hardware-aware performance evaluation

Techniques. However, this study did not measure the real-world practical inference efficiency of the models tested. Another example is SliM-LLM (Huang et al., 2025b), who used the technique of mixed precision quantization by identifying the salient weights of a neural network and assigning them varying numbers of bits, thus outperforming even uniform, low-bit precision quantization models. However, this technique also adds additional complexity and does not include any analysis of real-world latencies or Throughput. Finally, 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 do not reflect 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 up to 405B (Lang et al., 2024) demonstrated that quantization allows for greater tolerance levels based on the larger model sizes (up to 405B) in comparison to smaller size models on average (i.e., for example, from 125M - 2800M), while ignoring other performance metrics such as inference speed/throughput, energy consumption, and kernel-level behaviours. In contrast, Systematic Characterization of LLM Quantization (Shi & Ding, 2025) provided

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

Paper	Core Research Focus	Methods Used	Key Findings / Results	Main Limitations (Gap)
(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	Lacks kernel-level or system deployment analysis
(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 unclear
(Lin et al., 2025)	Quantization of diffusion-based LLMs	PTQ adapted for diffusion architectures	Quantization feasible beyond autoregressive LLMs	Limited relevance to standard LLM inference pipelines
(Hasan, 2024)	Comparison of PTQ and QAT strategies	Empirical analysis across quantization regimes	QAT outperforms PTQ under aggressive compression	Mostly qualitative; lacks system metrics
(Sharify et al., 2024)	Novel numerical formats for PTQ	Microscaling-based low-bit representations	Better accuracy–compression trade-off than INT formats	No end-to-end inference benchmarking
(Liu et al., 2024a)	Kernel-aware post-training quantization	Reduced kernel size for precise compression	Improves accuracy at low bit-widths	Limited evaluation on modern LLM architectures
(Zhao et al., 2024)	Learnable post-training quantization	Optimization of quantization parameters	Improves robustness over fixed PTQ methods	Training overhead; limited system-level validation
(Li et al., 2024)	Systematic evaluation of quantized LLMs	PTQ across tasks and bit-widths	Quantization impact varies significantly across tasks	No latency, throughput, or hardware-aware evaluation
(Jin et al., 2024)	Comprehensive evaluation of quantization strategies	Multi-method, multi-task benchmarking	Confirms no single best quantization approach	Does not address hardware or throughput aspects
(Liu et al., 2023)	Data-free quantization-aware training	Synthetic calibration and QAT	Avoids reliance on proprietary datasets	Increased training complexity; limited inference analysis

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 emphasising that quality loss is determined largely by the task that is addressed. Limitations identified included the fact that many of the studies relied on stable infrastructure and experienced 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 and Specialist LLM + Knowledge Graph focus 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. Both of these methods improve the utility of their models in their respective domains, but they are primarily complementary to low-bit quantization, in that they typically do not consider inference efficiency or latency.

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. However, there is still a major gap between algorithmic approaches to quantization and the actual measures of latency, throughput, and energy efficiency for the deployment of quantized LLMs in practical, large-scale applications across multiple types of hardware and use cases. Remaining gaps need to be addressed if extensive deployments of quantized LLMs are to be accomplished.

3 Proposed Utility-Aware Quantization Framework (UAQF)

This section describes a 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 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). Unlike previous work on quantization that used either isolated microbenchmarks or theoretical estimates of FLOP reductions as evaluation metrics, the proposed UAQF evaluates quantized models from beginning to end (real models, real prompts, and real hardware).

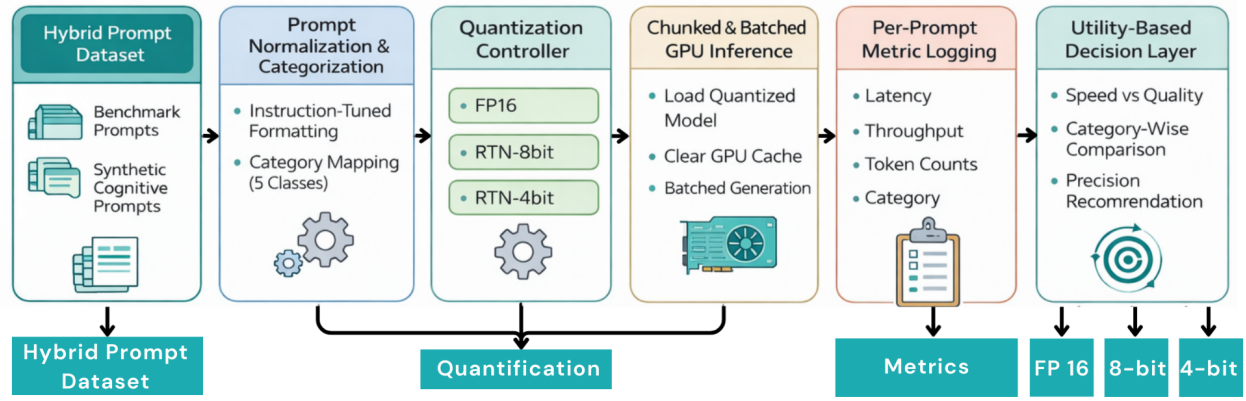

Figure 1: The steps of the proposed utility-aware quantization framework

The proposed framework UAQF follows a modular, pipeline-based design consisting of five sequential stages:

- Prompt and Dataset Preparation
- LLM Selection and Precision Configuration
- Quantized Inference Execution

- Performance and Quality Measurement
- Utility-Matching and Trade-off Analysis

This decomposition ensures reproducibility, extensibility to additional quantization methods, and isolation of quantization effects from architectural or dataset confounders. Table 3 summarizes quantization configuration of baseline and evaluated precision Settings.

Table 3: 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

3.1 Preprocessing and Prompt Engineering

3.1.1 Dataset Construction

To validate the robustness of quantization methods across a variety of cognitive functions, prompts were created via both data-driven and synthetic means. We began with a set of 13 conceptual task categories, such as commonsense reasoning, mathematical reasoning, following instructions, recalling known information, conversation with another participant, and several types of complex reasoning tasks (e.g., causal reasoning, ethical reasoning, and temporal reasoning).

During the evaluation, these conceptual task categories are collapsed into five operational task categories: Dialogue, Instruction Following, Knowledge/Commonsense, Math/Analytical, and Other using prefix-based heuristics embedded in the evaluation script. By collapsing the 13 conceptual categories into five operational task categories, we were able to maintain task diversity while reducing variation and increasing statistical stability within our analyses.

3.1.2 Prompt Normalization

All prompts are normalized to a unified instruction-tuned format compatible with chat-based LLMs. Deterministic decoding parameters (fixed temperature, top-p, and maximum tokens) are enforced across all precision settings to ensure fair and comparable evaluation.

3.2 LLM Selection and Quantization Configuration

3.2.1 Model Choice

The model chosen for the framework is Mistral-7B-Instruct-v0.2, an instruction-tuned LLM containing 7 billion parameters. This model struck a good balance between experimental feasibility and practical importance, being large enough that it will identify memory and bandwidth limitations but small enough where you can run the model on commodity GPUs such as NVIDIA A100, H100, or accelerators like those in Colab.

Having a fixed model across all experiments ensures that differences observed in the experiment were due to the quantization process only and not from any change in model architecture or other variable factors.

3.3 Quantization Regimes

The proposed solution evaluates three precision regimes:

- FP16/BF16: High-precision baseline
- RTN-8bit: Runtime-only 8-bit quantization
- RTN-4bit: Runtime-only 4-bit quantization

3.4 Quantized Inference and Execution Engine

The inference was performed using real GPUs with chunked execution to provide reliable execution during periods of limited memory. Prompts are processed in batches to maximize GPU utilization and to prevent out-of-memory errors.

Explicit memory cleanup is done between chunks to prevent fragmentation from happening; resumable inference capabilities were created by detecting previously processed prompts and appending them to both JSON and CSV logs. This capability allows for long-term experiments to continue running in the event of hardware failure without any data loss occurring.

3.5 Algorithm of the proposed framework

Algorithm 1 (Table 4) formalizes the proposed Utility-Aware Quantization Framework (UAQF). Table 5 summarizes key notations used in algorithm.

Table 4: Utility Evaluation Algorithm for Quantized LLM Inference

Stage	**Description**
Input	Prompt dataset $P = \{p_1, p_2, \ldots, p_n\}$; quantization set $Q = \{q_{\text{fp16}}, q_{\text{8b}}, q_{\text{4b}}\}$
Output	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;
	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, throughput s_i, input tokens x_i, output tokens y_i, and category $c_i \in C$; append $(t_i, s_i, x_i, y_i, c_i)$ to L_q
Step 3: Aggregation	Compute average latency $\bar{t}_q$, average throughput $\bar{s}_q$;
	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 $QL_q = \Delta\text{PP}_q$;
	Compute utility score $U_q = f(S_q, QL_q)$

Step 1: Prompt Preprocessing and Categorization The algorithm begins by initializing a fixed set of operational prompt categories, namely Dialogue, Instruction Following, Knowledge/Commonsense,

Table 5: Key Notations Used in the Quantization Utility Evaluation Algorithm

Notation	Description
P	Prompt dataset used for evaluation
p_i	i-th prompt in the dataset P
Q	Set of evaluated quantization levels
q	A specific quantization configuration, $q \in Q$
M	Base (full-precision) language model
M_q	Model M quantized under configuration q
C	Set of operational prompt categories
c_i	Category assigned to prompt p_i, $c_i \in C$
L_q	Metric log collected for quantization level q
t_i	End-to-end inference latency for prompt p_i
s_i	Token throughput (tokens/sec) for prompt p_i
x_i	Number of input tokens for prompt p_i
y_i	Number of output tokens generated for prompt p_i
$\bar{t}_q$	Average latency over all prompts under q
$\bar{s}_q$	Average throughput over all prompts under q
Δy_q	Relative change in output length under q
ΔPP_q	Perplexity change under q relative to FP16
S_q	Throughput speedup relative to FP16
QL_q	Quality loss metric for quantization level q
U_q	Final utility score for quantization level q
$f(\cdot)$	Utility function balancing speedup and quality loss

Math/Analytical, and Other. While many fine-grained conceptual task classes may exist in the source data, collapsing these into the operational prompt categories serves two important functions: First, aggregating fewer variances will increase statistical robustness; Second, quantization has a greater impact on the associated patterns of numerical computation and memory access rather than the meaning or identity of each individual task type. The Prompt Structure(s) for all applications are standardized into a uniform Instruction-Tuned format prior to any inference being performed. The intended outcome is that any detected differences in relative quality or performance at various levels of quantization will be due to a change in numerical precision and not to the way in which a given prompt was structured or decodes.

Step 2: Iteration Over Quantization Levels The primary function of the proposed UAQF is to evaluate different quantization configurations at which base model was quantized and subsequently loaded into the destination GPU for each configuration (FP16, 8-bit, and 4-bit). To ensure complete separation of the effects of quantization from one another (and therefore to limit inter-system contamination of metrics between precision regimes), algorithm 1 treated quantization as an outer iteration loop; thus, any metric logging structures initialized for a particular quantization level would not have any effect on the metrics of the next quantization level(s). This also allowed for a resumable workflow and the ability to analyse inference outputs after the fact without having to re-run inference. This feature is especially useful in environments where resources may be in short supply or may not be fully operational.

Step 3: Chunked and Batched Inference Execution Although the prompt set is divided into several smaller chunks, they are executed on the GPU sequentially, with each smaller chunk consisting of a set of prompts (or some subset of them) that can be run as a group (in other words, as batches). The batching of prompts within chunks maximises the potential for throughput while also minimising the chances of out-of-memory errors occurring. Each chunk is explicitly cleared from GPU cache and synchronised with the prior chunks to keep memory fragmentation from interfering with accurate timing measurement. Each time a prompt is executed, the algorithm records its end-to-end latency and throughput (tokens/second), the number of input tokens and the number of output tokens, and the prompt category. By logging data on a per-prompt basis, the model's inference operations and the subsequent aggregation of data related to those operations by category, precision and/or hardware configuration can be viewed in greater detail.

Step 4: Metric Aggregation and Quality Assessment After the inference process is completed for the quantization level defined by the user prompt, the method aggregates the latency throughputs created during all of the inferences processed. While average latency and throughput provide an approximate indication of the performance of the process (i.e. when considering coarse measurements of performance), the degree of stability in lengths of the output when obtaining the outputs through the inferences is an important indication of potential negative effects that exist due to premature termination or truncated outputs.. Quality of the generated outputs can also be assessed by measuring the difference in perplexity (i.e. a measure of how likely or likely different models will generate similar outputs) and comparing it to the baseline performance using FP16. Although perplexity will not provide a perfect signal of quality from a human perspective, it can provide a general means for identifying degradation in the quality of generated outputs caused by quantization effects and does not require extensive human evaluations to do so.

Step 5: Utility-Matching Analysis The method allows for this comparison between each quantization level and the respective utility of that configuration due to the introduction of the utility-matching principle, which is the primary conceptual foundation of the framework described. Instead of viewing speed and quality as separate metrics, the method combines them by calculating a utility score associated with each quantized configuration. The utility score for each quantized configuration is calculated based on the relative speedup from using FP16 as the baseline for comparison, which allows comparisons of quantized levels with deployment specific constraints on them - as opposed to simplifying the results by simply declaring a single "best" bit-width.

4 Results and Discussion

4.1 Dataset and Prompt Design Methodology

In this study, a hybrid dataset and the prompt design methodology are used to assess the robustness of quantized large language models for multiple cognitive behaviours. The dataset is deliberately constructed to balance standardized benchmarking with targeted stress testing, thereby ensuring both reproducibility and comprehensive coverage of higher-order reasoning abilities. To create the evaluation framework for the proposed approach, the study has integrated existing benchmark datasets as well as manual synthetic prompts based on cognitive behaviours that are typically underrepresented using standardised evaluation suites.

The dataset is organized into 13 different category types at its conceptual level. Task categories are further divided into two types of dataset categorization based on whether they rely on a synthetic cognitive approach or have a dataset-driven basis. The dataset-driven categories are used to evaluate the fundamental reasoning skills measured using standardised benchmarks. This enables comparison with prior work. These include:

- Commonsense Reasoning, sourced from CommonsenseQA
- Math / Analytical Reasoning, sourced from GSM8K
- Instruction Following, sourced from Alpaca
- Knowledge Recall, sourced from TriviaQA

In addition, nine synthetic cognitive categories have been manually developed to investigate complex, higher-order reasoning behaviours that cannot be easily measured using existing publicly available datasets and to which no single standard benchmark is currently available. These synthetic categories include:

- Dialogue / Conversational Fluency
- Code Reasoning & Symbolic Logic
- Ethical & Bias Sensitivity
- Creative Generation

- Causal & Counterfactual Reasoning
- Metacognition & Self-Correction
- Multilingual & Cross-Cultural Reasoning
- Abstract Relational Reasoning
- Temporal Reasoning & Planning

For evaluation and analysis, the 13 conceptual categories were operationally collapsed into five broader categories using prefix-based heuristics within the evaluation script. These operational categories are:

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

All nine synthetic cognitive categories were intentionally combined into a single category called "Other." The decision to create this category is justified on a methodological basis because the quantization process affects numerical computation, memory access patterns, and internal representations, but does not affect the fine-grained semantic task identity of each synthetic task. By creating a stress-test category for all of the cognitively different synthetic tasks, it increases the robustness of the results due to the reduced variability in evaluation results and helps to eliminate overinterpretation of semantic distinctions that quantization is not likely to have a direct effect on. This operational collapse does not eliminate any data; rather, it simply abstracts the labels of each of the multiple categories while continuing to maintain the full extent and intention of all of the original prompts.

4.2 Experimental Setup

Table 6: Experimental Setup

Model	mistralai/Mistral-7B-Instruct-v0.2
Methods	RTN-8b, RTN-4b, FP16 (baseline)
Datase	Eval$_{P}rompts_{H}ybrid_{V}3$
Colab GPU	NVIDIA A100-SXM4-80GB
Torch	2.2.2+cu121
Transformer	4.47.1
Bitsandbytes	0.44.1

All experiments were conducted under identical decoding parameters and execution conditions to isolate the effect of numerical precision from architectural or runtime confounders. The experimental settings are summarized in table 6.

4.3 Results of the proposed framework

4.3.1 Overall Speed–Quality Trade-off

Figure 2 illustrates the global relationship between inference speedup and quality degradation across precision regimes. FP16 serves as the reference point, achieving zero perplexity change at unit speed. RTN-8bit provides moderate throughput improvements but exhibits the highest relative perplexity degradation. In

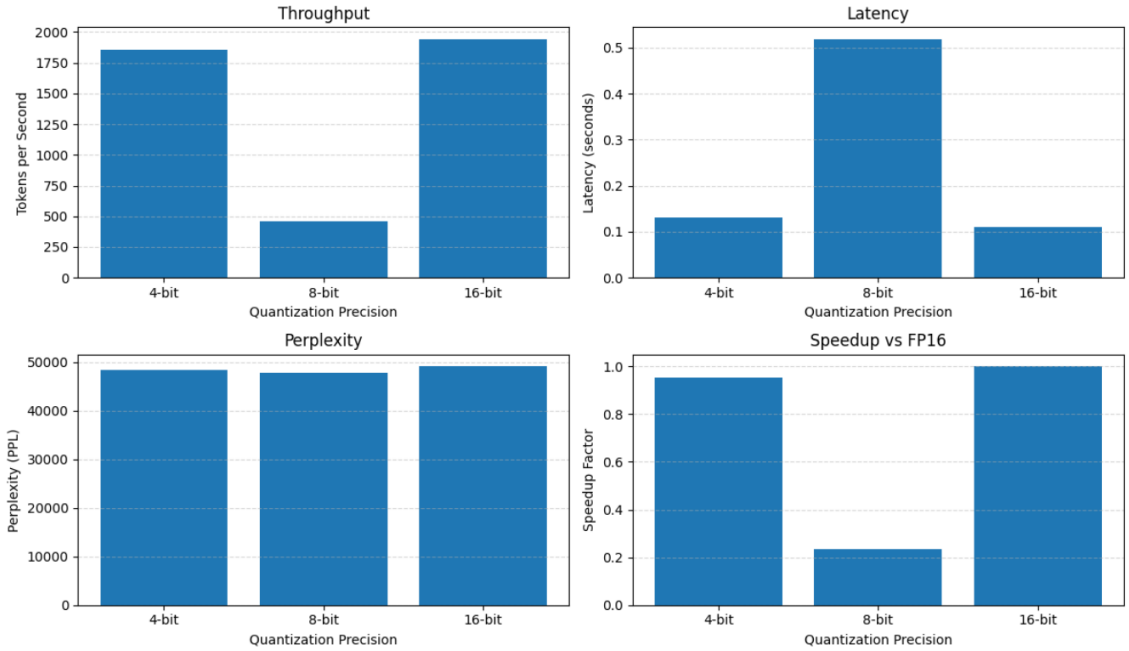

Figure 2: Raw Performance Metrics across Quantization Levels

contrast, RTN-4bit achieves comparable or higher inference speed while incurring substantially lower quality degradation than RTN-8bit.

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.

4.3.2 Category-Wise Throughput Analysis

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 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 the greatest sensitivity to reduced precision. Under RTN-4bit, these tasks experience pronounced throughput degradation, indicating that numerical computation amplifies quantization noise. As a result, higher precision remains preferable for analytically intensive workloads.

4.4 Comparison with existing framework

Table 7: Comparison with existing framework

Aspect	Existing	Proposed Framework
Metrics	Accuracy / PPL	Latency + Throughput + PPL
Hardware realism	Limited	Real GPU
Decision output	None	Deployment recommendation
Bit-width selection	Static	Utility-driven

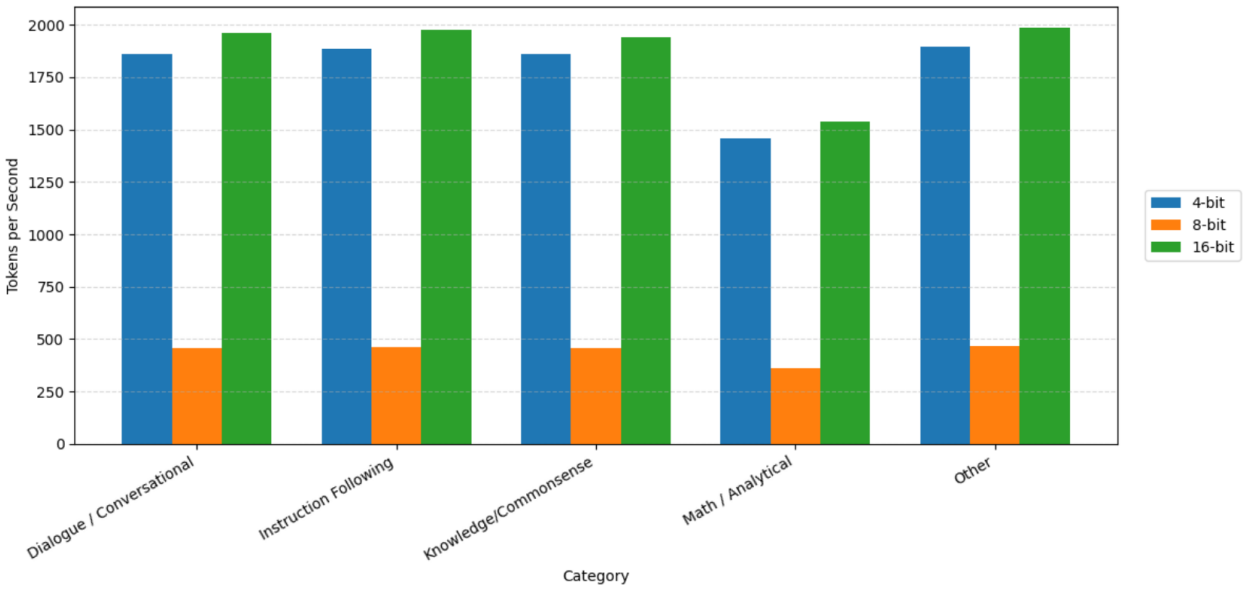

Figure 3: Category-wise Throughput Across Quantization Levels

The table 7 summarizes comparison. Unlike prior quantization studies that emphasize task-level accuracy or benchmark scores, the proposed framework evaluates quantization under full end-to-end inference conditions. Existing frameworks typically lack explicit consideration of latency, throughput, or deployment utility, and often rely on idealized or synthetic benchmarks. In contrast, the proposed approach integrates real hardware measurements and produces actionable precision recommendations through a unified utility formulation.

Furthermore, while many existing methods introduce calibration or architectural complexity, the present study establishes a clean runtime-only baseline, enabling transparent interpretation of quantization effects without confounding overheads.

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. Unlike many prior works 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.

The framework also demonstrates strong engineering robustness. Features such as chunked execution, resumable logging, and deterministic decoding ensure experimental stability even under constrained or unstable environments. Additionally, the abstraction of prompt categories into operational buckets reduces variance while preserving task diversity, strengthening the statistical reliability of aggregated metrics.

Finally, the focus on a single, well-chosen baseline quantization method (RTN) avoids methodological confounding 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.

Second, the experimental scope is limited to a single model architecture and size class. Although the chosen model is representative of modern open-weight LLMs, quantization behavior may differ for larger models, mixture-of-experts architectures, or models with different training objectives.

Third, 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.

5.3 Conclusion and final remarks

In summary, this work establishes a reproducible and deployment-oriented methodology for evaluating LLM quantization. By combining empirical measurement with utility-aware analysis, the proposed framework provides practical guidance for precision selection under real-world constraints. The results highlight the importance of empirical evaluation and challenge static assumptions about optimal quantization precision.

Acknowledgments

No authors have conflict of interest.

References

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.

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.

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*, 2024a. 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*, 2024b. 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.

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.