Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.078985

ARTICLE

Quantized Transformers in Practice: Benchmarking Full- and Low-Precision LLMs across Two Processors

Simona-Vasilica Oprea and Adela Bâra*

Economic Informatics and Cybernetics Department, Bucharest University of Economic Studies, Calea Dorobanți 15-17, district 1, Bucharest, Romania

*Corresponding Author: Adela Bâra. Email: bara.adela@ie.ase.ro

Received: 12 January 2026; Accepted: 20 February 2026; Published: 09 April 2026

ABSTRACT: Quantization has emerged as an important technique for enabling efficient deployment of large language models (LLMs) by reducing their memory and computational requirements. This research conducts an evaluation of INT8 quantization on several state-of-the-art LLMs, GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat, across two hardware configurations: NVIDIA RTX4070 Laptop GPU and RTX4080 Laptop GPU and two tasks: text and code generation. By comparing quantized INT8 models with their FP16 counterparts and a human-written reference, we quantify the trade-offs between performance and efficiency using standard natural language generation metrics (BLEU, ROUGE-1, ROUGE-L) and semantic analysis via GPT-4o and Gemini 2.5 Flash (Google). The results reveal that INT8 post-training quantization (PTQ), hereafter referred to as INT8, substantially reduces inference time and memory footprint, with minimal impact on topical relevance but a notable decline in lexical precision, fluency and structural coherence. The extent of quality degradation varies by model size and architecture, with smaller models demonstrating greater resilience to quantization. Furthermore, we identify several limitations in quantized outputs, including reduced expressiveness, while highlighting their suitability for resource-constrained or real-time applications, such as robots monitoring safety standards in manufacturing environments. On average, INT8 quantization results in a 3.4 times speedup over FP16 inference across all tested models and GPUs (excluding configurations affected by CPU offloading), with the largest gains observed in large models like LLaMA-2-7B-Chat. The results also indicate that structured code generation exhibits slightly greater sensitivity to INT8 quantization compared to explanatory text generation.

KEYWORDS: Large language models; quantization; integer precision inference; text generation; code generation; hardware configuration

1 Introduction

Quantization is the process of reducing the precision of numerical representations. For LLMs, with hundreds of billions of parameters (e.g., 175 G for GPT-3, 70 G for LLaMA2 and more presumably for GPT-4), quantization facilitates support of more efficient inference [1,2] and deployment of 32-bit FP32 parameters to lower-precision formats such as 16-bit FP16/BF16, 8-bit INT8 or even ultra-low precisions such as 4-bit INT4 and binary representation. This loss of accuracy leads to a decreased memory bandwidth requirement, increased computational throughput and lower latency, especially desired in the embedded or real-time domain [3]. In the full precision format, each parameter requires 4 bytes of storage, which quickly becomes infeasible for large models. INT8 quantization reduces this to 1 byte, while INT4 compresses parameters to just 0.5 bytes, effectively decreasing the model size by up to 8 times [4]. It comes at the cost of reduced numerical precision, which can impact model accuracy, especially in non-linear components such as the

attention mechanism or in layers with high dynamic range activations. In order to mitigate this, advanced quantization methods incorporate techniques such as per-channel scaling, asymmetric quantization and learned quantization parameters. Thus, our primary motivation behind quantization lies in its ability to reduce the memory footprint of a model. Furthermore, quantization accelerates inference by enabling faster arithmetic operations [5] as integer-based computations are more efficient on many modern hardware platforms, including GPUs, TPUs and specialized AI accelerators [6]. As a result, quantized models are often more suitable for real-time applications such as chatbots, virtual assistants, robots in manufacturing environments and interactive coding tools [7]. For instance, they may enhance the explainability of alerts in factory environments, where anomaly detection systems may trigger notifications, but the underlying event schema alone is insufficient to fully explain the issue. In such cases, LLMs integrated with Retrieval-Augmented Generation (RAG) applications and supplied with relevant safety regulations can generate contextualized explanations. This approach provides a secondary validation layer and offers more detailed guidance for addressing the identified hazard or standards breach.

Moreover, quantization holds importance for democratizing access to large-scale AI, enabling smaller organizations and developers to run powerful models. In applications involving mobile devices, IoT systems and embedded computing, quantization is the only feasible approach to deploying LLMs [8]. From a sustainability perspective, quantization contributes to reducing the energy consumption associated with inference. Given the immense computational cost of operating state-of-the-art LLMs, even small reductions in precision translate to energy savings across large-scale deployments, aligning with global efforts to promote green AI and reduce the environmental footprint of machine learning applications.

Quantization also raises several questions and technical challenges [9,10]. It can be applied at different stages in the model lifecycle. Post-training quantization (PTQ), which applies quantization to an already-trained model, may sometimes lead to performance degradation, particularly in sensitive components such as attention layers or activation functions [11]. PTQ is computationally efficient and does not require retraining, but its performance depends heavily on the model's robustness to quantization noise [12]. PTQ typically uses calibration data to compute scale and zero-point values for each layer or tensor [13]. Common schemes include symmetric or asymmetric uniform quantization, and methods such as Gradient Post-Training Quantization (GPTQ) and Activation-aware Weight Quantization (AWQ) improve performance by accounting for weight importance [14] and Hessian-based error propagation. In contrast, Quantization-Aware Training (QAT) simulates low-precision arithmetic during training by introducing fake quantization nodes into the computation graph [15,16]. This allows the optimizer to adapt the weights to quantization-induced errors [17]. QAT achieves higher accuracy under aggressive quantization (e.g., INT4 or INT3), but its memory and compute overhead make it impractical for very large LLMs unless applied selectively or during fine-tuning. Hybrid approaches, such as quantizing only specific layers or using mixed-precision quantization (e.g., FP16 for activations, INT8 for weights), are also actively explored.

Mixed-precision quantization, wherein different parts of the model are quantized to different levels, also remains an active area of investigation, as does the development of custom quantization schemes tailored to specific applications or hardware platforms [18,19]. The choice of quantization granularity further impacts model behavior. Per-layer quantization applies a single scale to an entire layer, which is fast but less accurate. Per-channel quantization assigns distinct scales to each output channel, improving representational fidelity at the cost of complexity. Per-token quantization, while less common, is also being explored for LLMs with variable token significance. Moreover, dynamic quantization (computed on-the-fly during inference) and static quantization (pre-computed and stored) offer different trade-offs between flexibility and speed. Transformers' components such as the softmax function in attention, LayerNorm operations and GELU activations are sensitive to precision loss. Specialized quantization approximations for these functions or

quantization-aware rewrites are often necessary [20,21]. In this context, research into quantized transformer kernels (e.g., using QKV fusion, memory-efficient attention or int8-matmul optimizations) is important for realizing practical speedups on hardware like NVIDIA A100s, AMD Instinct MI250s or custom ASICs [22].

For research, quantization offers a challenging interdisciplinary field, intersecting with optimization theory, hardware architecture, information compression and the study of representational robustness in deep neural networks. Moreover, the implications of quantization on model fairness, interpretability and generalization are not yet fully understood. There is also a need for systematic benchmarking of quantized models, especially with respect to how quantization affects bias and performance across diverse inputs. From a deployment perspective, quantization unlocks the ability to serve LLMs on edge devices, robots, mobile platforms or low-cost cloud infrastructure. This is especially relevant given the emergence of efficient model serving frameworks (e.g., TensorRT-LLM, ONNX Runtime, Hugging Face Optimum) and quantization libraries like Intel Neural Compressor, BitsAndBytes and LLM.int8(). These tools allow for better integration of quantized models into production pipelines, with auto-tuning capabilities and hardware-specific optimizations [23].

However, the implications of quantization go beyond computational efficiency. Understanding how quantization affects a model's robustness, generalization, calibration and fairness is important. Quantization may amplify biases or destabilize attention distributions, particularly under low precision [24]. Additionally, interpretability methods (e.g., attention visualization, attribution maps) may yield misleading insights if the quantized model deviates from its full-precision counterpart. As such, evaluation benchmarks and quantization-aware metrics are necessary to assess model quality post-compression [25]. Thus, topics such as quantization error propagation in deep residual networks, the Lipschitz continuity of quantized transformers and optimal quantization grid design using information theory are under active exploration. Moreover, quantization often complements other compression techniques such as pruning [26], distillation and low-rank decomposition. Unlike compression methods that rely on retraining (e.g., pruning, distillation, low-rank adaptation), PTQ avoids the risk of *catastrophic forgetting* and preserves the original model's learned knowledge [27].

While extensive research has explored aggressive low-bit quantization schemes (e.g., INT4, INT3) and QAT, many of these approaches require retraining, calibration datasets or specialized tooling that limits their applicability in practical deployment settings. In contrast, INT8 PTQ remains the most widely adopted and production-ready solution, supported natively by mainstream inference frameworks and commodity GPU hardware.

Despite its widespread use, there is still limited empirical understanding of how INT8 PTQ affects both inference efficiency and generation quality across different LLM architectures and mid-tier consumer GPUs, which are increasingly used in decentralized, small-scale and individual deployments. Existing studies often focus on server-grade accelerators or benchmark-centric evaluations, while overlooking the behavior of quantized LLMs in realistic, practitioner-oriented environments.

Importantly, while INT8 PTQ offers substantial gains in inference speed and memory efficiency, it can introduce non-trivial degradation in lexical precision, fluency and structural coherence, particularly for larger models. Understanding the magnitude, consistency and practical implications of this degradation is essential for informed deployment decisions. Rather than attempting to identify a universally optimal quantization strategy, there is a need for transparent, reproducible benchmarking that characterizes INT8 PTQ as a baseline trade-off point between efficiency and output quality.

Motivated by these considerations, our paper presents a systematic empirical evaluation of INT8 PTQ applied to multiple LLM families: GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat, executed on

two widely available consumer GPUs, NVIDIA RTX4070 and RTX4080 Laptop GPUs. By comparing INT8-quantized models against their FP16 counterparts under controlled generation settings, we quantify both performance gains and quality degradation using a combination of lexical metrics and qualitative semantic analysis.

It is important to emphasize that this work does not claim INT8 PTQ to be optimal relative to lower-bit quantization or QAT-based approaches. Instead, INT8 PTQ is intentionally chosen as a deployment-oriented reference point, enabling practitioners to understand what efficiency gains can be achieved *without retraining*, and what quality trade-offs must be accepted under such constraints. More aggressive or training-aware quantization methods are explicitly positioned as complementary extensions rather than alternatives to current research.

The main contributions of this work are as follows:

1. Deployment-oriented benchmarking of INT8 PTQ for LLMs. We provide a systematic comparison of FP16 and INT8 PTQ versions of GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat, focusing on realistic inference scenarios without retraining or calibration.
2. Cross-architecture and cross-hardware evaluation. The study evaluates quantization effects across models of different sizes and architectures on two mid-tier consumer GPUs (NVIDIA RTX4070 and RTX4080 Laptop GPUs), offering practical insights beyond server-grade hardware benchmarks.
3. Quantification of efficiency-quality trade-offs. We measure inference latency improvements alongside degradation in lexical precision, fluency and structural coherence, highlighting how quantization impacts models differently depending on scale and hardware.
4. Qualitative analysis of quantization-induced degradation. Through semantic assessment, we characterize recurring degradation patterns in INT8 outputs, providing diagnostic insight into when INT8 PTQ is suitable and when full-precision inference remains preferable.

2 Literature Review

2.1 Relevant Works

The challenge of deploying deep convolutional neural networks (CNNs) on mobile devices due to their high computational demands was addressed [28]. Although network quantization was commonly used to reduce this burden, existing low-bitwidth methods suffered from a datatype mismatch problem, which led to inefficient instruction execution on mobile CPUs. The authors proposed a novel quantization method that ensured all operations during inference used only integer arithmetic, thereby eliminating datatype mismatches. They improved the quantization function and introduced a logarithm-like method to quantize batch normalization parameters. Experiments showed that their 4-bit quantized ResNet-18 model achieved accuracy comparable to state-of-the-art methods while significantly reducing runtime, up to 4.33 times faster on ARMv8 CPUs, with only a small drop in accuracy (0.7% top-1, 0.4% top-5 on ImageNet). Moreover, Ref. [29] addressed the limitations of QAT for large pre-trained language models (PLMs), which required full dataset access, extensive training time and high memory usage. Instead, the authors studied PTQ and proposed a method called module-wise quantization error minimization (MREM). This approach partitioned the PLM into modules and minimized the quantization-induced reconstruction error within each module. They also introduced a model-parallel training strategy that allowed each module to be trained independently on separate devices, achieving near-theoretical speed-ups (e.g., 4 times on 4 GPUs). Experiments on GLUE and SQuAD benchmarks showed that MREM achieved performance close to QAT while significantly reducing training time, memory usage and data requirements.

Additionally, Ref. [30] addressed the challenge of deploying LLMs, which typically require substantial computational resources and memory. While model quantization had been widely used to reduce this overhead, most previous methods relied on a small subset of training data for calibration, potentially harming generalization to new tasks. To overcome this, the authors proposed EasyQuant, a training-free and data-free weight-only quantization algorithm for LLMs. They observed that outliers in weights and quantization ranges significantly influenced quantization error. Thus, EasyQuant preserved a small fraction of outliers (less than 1%) and optimized the quantization range to minimize reconstruction error. Results showed that EasyQuant achieved performance comparable to the original models and outperformed existing data-dependent methods in speed, running more than 10 times faster. Because it did not rely on training data, EasyQuant also ensured strong generalization performance, even for models exceeding 100 billion parameters. Also, Ref. [31] addressed the challenge of deploying LLMs on mobile devices, where limited storage necessitates effective compression. While weight clustering, particularly Differentiable KMeans Clustering (DKM), had shown a strong balance between compression ratio and accuracy, its high memory requirements made it impractical for LLM fine-tuning. To overcome this, the authors proposed eDKM, a memory-efficient implementation of DKM. They introduced techniques such as uniquification and sharding to avoid redundant tensor copies and significantly reduce memory usage during the backward pass. Experiments showed that eDKM compressed a pretrained LLaMA7B model from 12.6 GB to 2.5 GB (3 bits per weight) using the Alpaca dataset, while reducing the memory footprint of a decoder layer by 130 times. The compressed model maintained competitive accuracy on benchmarks, achieving 77.7% on PIQA and 66.1% on Winogrande.

Furthermore, Ref. [32] addressed the inefficiency of transformer-based models like BERT and RoBERTa for inference due to their high memory usage, latency and power consumption. While quantization had been explored, prior approaches still relied on floating-point operations during inference, limiting compatibility with integer-only hardware like Turing Tensor Cores and ARM processors. The authors proposed I-BERT, a novel quantization scheme that enabled fully integer-only inference for Transformer models. By developing integer-only approximations for nonlinear functions such as GELU, Softmax and LayerNorm, I-BERT eliminated all floating-point operations during inference. Evaluations on GLUE tasks using RoBERTa-Base and RoBERTa-Large showed that I-BERT maintained accuracy comparable to (or slightly better than) full-precision models. Additionally, the preliminary INT8 implementation achieved a 2.4–4.0 times speedup over FP32 inference on a T4 GPU. Moreover, Ref. [33] introduced oBERTa, a family of language models designed to offer 3.8 to 24.3 times faster inference without requiring users to have expertise in model compression. oBERTa built upon existing techniques such as pruning, knowledge distillation and quantization, and further enhanced them through the use of frozen embeddings, improved distillation and refined model initialization. The authors examined differences between RoBERTa and BERT in terms of compressibility during pre-training and fine-tuning, finding RoBERTa less amenable to fine-tuning compression. Evaluations on seven NLP tasks showed that pruned oBERTa models matched the performance of BERTbase and even outperformed Prune OFA Large on SQuAD V1.1, while being 8 and 2 times faster, respectively.

Another study addressed the high computational and memory demands of LLM inference and proposed a more efficient quantization approach [34]. While most prior work focused on 8-bit quantization, the authors identified numerical scaling offsets as a key bottleneck in LLM quantization. To overcome this, they introduced block quantization for LLMs, a method that shares scaling factors across packed values to reduce scaling offsets purely through arithmetic, without altering the computational pipeline. Their 6-bit quantized models achieved 19 times higher arithmetic density and 5 times higher memory density compared to float32, outperforming 8-bit methods by 2.5 and 1.2 times, respectively, without data calibration or retraining. The authors also explored sub-8-bit quantization, highlighting challenges such as mismatched activation and

weight distributions, and identified strategies like optimal fine-tuning and exploiting lower quantization granularity. These advances enabled nearly-lossless 4-bit LLMs on downstream tasks. Further, Ref. [35] proposed GPUSQ-ViT, a compression framework for Vision Transformers optimized for GPU deployment. It combined 2:4 structured pruning and quantization-aware training to exploit GPU-friendly sparse and integer operations. A mixed-strategy knowledge distillation guided the process. GPUSQ-ViT achieved up to 12.7 times model size and 62 times FLOPs reduction with minimal accuracy loss on standard vision benchmarks. It also improved latency and throughput by up to 1.79 and 3.43 times on A100 GPUs and 1.69 and 2.51 times on AGX Orin.

2.2 Structured Analysis of Quantization Literature

Recent research on model compression and quantization spans several complementary directions, including integer-only inference, PTQ, QAT, sub-8-bit methods and hardware–algorithm co-design. Early work on integer-only quantization addressed the inefficiency caused by datatype mismatches in low-bit inference pipelines. For example, Ref. [28] proposed a fully integer-based quantization scheme for CNNs, eliminating floating-point operations during inference and achieving substantial runtime reductions on ARM CPUs. Similarly, I-BERT [32] extended this paradigm to Transformer architectures by approximating nonlinear functions (e.g., GELU, Softmax, LayerNorm) with integer-only formulations, enabling compatibility with integer hardware such as Tensor Cores. These works emphasize that performance gains depend not only on bit-width reduction but also on kernel-level and hardware-aligned implementation strategies.

In the LLM domain, PTQ has gained prominence due to its practical advantages over QAT. MREM [29] addressed the high cost of QAT by proposing module-wise quantization error minimization, reducing reconstruction error without full retraining. EasyQuant [30] further removed calibration data requirements by optimizing quantization ranges and selectively preserving outliers, demonstrating scalability even for models exceeding 100 billion parameters. These methods underline the importance of error distribution and outlier handling in LLM quantization.

More aggressive compression strategies explore sub-8-bit quantization. Block-based quantization techniques [34] reduce scaling overhead and improve arithmetic density, enabling near-lossless 4-bit performance under certain conditions. Weight clustering approaches such as eDKM [31] reduce memory footprint dramatically (e.g., 3-bit compression for LLaMA-7B) while maintaining competitive benchmark performance. However, such methods often require fine-tuning, specialized kernels or algorithmic adaptation and their deployment characteristics may differ from standard INT8 PTQ workflows.

Beyond quantization alone, some studies combine pruning, distillation and quantization for improved acceleration. oBERTa [33] and GPUSQ-ViT [35] demonstrate that compression effectiveness can be amplified through co-design strategies tailored to GPU architectures. They highlight that speedup is not determined solely by precision reduction but by the interaction between sparsity, kernel efficiency and memory hierarchy.

While existing research extensively investigates algorithmic improvements for low-bit quantization and sub-8-bit compression, comparatively fewer studies provide deployment-oriented evaluations of standard INT8 PTQ across multiple LLM families under realistic inference conditions on mid-tier consumer GPUs. Much of the literature emphasizes either aggressive compression (e.g., 4-bit schemes), retraining-based optimization (QAT) or server-grade hardware benchmarks.

In contrast, our study deliberately focuses on INT8 PTQ as a production-ready baseline supported by mainstream inference frameworks. Rather than proposing a new quantization algorithm, we systematically evaluate efficiency–quality trade-offs across GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat on

RTX40-series laptop GPUs. By combining lexical metrics with semantic assessment, we aim to characterize quantization-induced degradation under reproducible deployment settings.

Therefore, several limitations persist in the current literature:

(a) Lack of holistic evaluation across model sizes and architectures. Most existing studies focus on either vision transformers or specific families of LLMs (e.g., BERT or RoBERTa), with limited empirical comparison across diverse LLM architectures such as GPT, LLaMA and Qwen. Moreover, performance assessments tend to overlook topical relevance and semantic fidelity in generation tasks, relying instead on accuracy or benchmark performance metrics.

(b) Insufficient benchmarking across practical hardware setups. Prior work often emphasizes inference on server-grade hardware (e.g., A100, T4, TPU), leaving a gap in understanding how quantized models behave on mid-tier consumer GPUs like the RTX4070 and RTX4080 Laptop GPUs, which are increasingly used in decentralized or individual deployments, with the efficiency–quality trade-offs in such contexts remaining underexplored.

Our research makes several novel contributions aimed at addressing the above limitations: cross-architecture, cross-hardware evaluation, multi-dimensional evaluation framework, quantification of efficiency vs. quality trade-offs, model-specific sensitivity analysis, recommendations for deployment and use cases.

3 Methodology

To assess the trade-offs between memory efficiency and output quality in quantized LLMs, we design and implement a comparative evaluation framework. Specifically, we benchmark models quantized to INT8 against their FP16 counterparts using a controlled experimental setup. This comparison highlights differences in inference speed and also investigates how quantization impacts textual output and code generation. Algorithm 1 outlines the step-by-step process used to load, run and evaluate both FP16 and INT8 versions of selected LLMs. The evaluation is conducted using standard natural language generation metrics: BLEU, ROUGE-1 and ROUGE-L, which provide complementary insights into lexical precision, recall and structural coherence. Outputs from the INT8 quantized models are compared to the FP16-generated reference outputs to quantify any degradation in generation quality.

Algorithm 1: Evaluate LLM with FP16 and INT8 quantization

Require: Model name model_name, Device label device_label, Prompt_text prompt_text
Ensure: Evaluation metrics and outputs for FP16 and INT8 models
1: Load tokenizer from model_name
2: Load FP16 model with torch_dtype = float16 and device_map = “auto”
3: Define quantization config quant_config with load_in_8 bit = True
4: Load INT8 model with quant config device_map = “auto”
5: Tokenize prompt_text into inputs
6: Clone inputs to FP16 and INT8 device maps
7: Start timer and generate output_fp16 from FP16 model
8: Compute time_fp16
9: Decode output_fp16 to text
10: Start timer and generate output_int8 from INT8 model

(Continued)

Algorithm 1 (continued)

11: Compute time_int8
12: Decode output_int8 to text
13: Set reference text ←output_fp16
14: for each output in {output_fp16, output_int8} **do**
15: Compute BLEU score against reference text
16: Compute ROUGE-1 and ROUGE-L scores against reference text
17: **end for**
18: **Return:**

- FP16 and INT8 output texts
- Inference times: time_fp16, time_int8
- BLEU, ROUGE-1, ROUGE-L scores for each output

To ensure reproducibility and a fair comparison between FP16 and INT8 inference, all experiments were conducted using identical generation and decoding settings across models and hardware configurations. Table 1 summarizes the full set of hyperparameters used for text generation. The same configuration was applied consistently to isolate the effects of quantization.

Table 1: Generation and decoding hyperparameters used in all experiments.

Parameter	Value	Description
Decoding strategy	Greedy decoding	Deterministic generation without sampling
Temperature	1.0	Standard temperature (no scaling)
Top-k	Not used	Disabled
Top-p (nucleus sampling)	Not used	Disabled
Max new tokens	256	Maximum number of generated tokens
Repetition penalty	1.0	No repetition penalty applied
Batch size	1	Single-prompt inference
Padding/truncation	Disabled	Input prompts fit within model context window
Random seed	Fixed	Same seed used for all runs
Warm-up runs	1	Warm-up run excluded from timing
Timing method	Wall-clock time	Measured only generation phase

The experimental setup includes several open sources models (e.g., GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat, briefly described in Table 2), all evaluated on prompts designed to elicit informative responses relevant to quantization. Two GPU configurations, NVIDIA GeForce RTX4070 Laptop GPU and RTX4080 Laptop GPU, are used to observe performance variations under different hardware capabilities.

The assessment of the quantized output for various LLMs is performed using several metrics, such as BLEU, ROUGE-1 and ROUGE-L. Bilingual Evaluation Understudy (BLEU) measures how many *n-gram* sequences (e.g., unigrams, bigrams, trigrams) in the generated text appear in the reference text. BLEU focuses on exact word matches and penalizes overgeneration using a brevity penalty. A score of 1.0 (or 100%) indicates a perfect match, while a lower score reflects fewer overlapping phrases. BLEU is particularly sensitive to word order, which makes it suitable for tasks requiring exact phrasing, but it may undervalue semantically correct paraphrasing. ROUGE-1 is part of the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) family of metrics and measures unigram (single-word) overlap between the generated text and the reference. It

captures basic lexical similarity and is a good indicator of whether the core content or vocabulary is present. While BLEU emphasizes precision, ROUGE-1 is recall-focused, valuing whether the important words from the reference appear in the output, even if phrased differently. ROUGE-L goes a step further by evaluating the longest common subsequence (LCS) between the generated and reference texts. This reflects lexical similarity and also sentence-level structure and coherence. ROUGE-L is useful for summarization and open-ended generation tasks, as it accounts for fluency and logical flow, rewarding outputs that preserve the original order of ideas even if the exact wording varies.

Table 2: Brief description of the three models GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat, considering several aspects such as size, depiction, model architecture, quantization relevance, etc.

Model	Size	Description	Architecture	Tokenizer	Training Data	Quantization Relevance	License
GPT-2	117M–1.5 B	OpenAI's early autoregressive language model for text generation.	Decoder-only	BPE (50k)	40 GB WebText	Small to mid-size layers; easy to quantize; older FP32 model	Open (MIT)
LLaMA-2-7B-Chat	7B	Meta's fine-tuned chat version of LLaMA-2, optimized for instruction-following and dialogue.	Decoder-only	SentencePiece	2 T tokens (public + curated)	High quantization interest; transformer optimized; common for 4/8-bit tests	Meta (Custom)
Qwen1.5-1.8B-Chat	1.8B	Alibaba's compact, instruction-tuned chat model from the Qwen 1.5 series.	Decoder-only	BPE (Fast-Tokenizer)	Multilingual + web/corpora	Small, efficient; ideal for 4-bit quantization in edge use cases	Apache 2.0

As BLEU captures precision and phrase-level accuracy, ROUGE-1 captures word-level recall and ROUGE-L reflects structural and sequential alignment, they provide a comprehensive evaluation of text generation quality. However, these metrics do not fully capture semantic meaning or contextual relevance, they remain standard tools for benchmarking and comparing language models, particularly in tasks like response generation. For assessing semantic meaning, GPT-4o and Gemini 2.5 Flash (Google) are used.

Furthermore, for evaluation, two processors are employed: NVIDIA GeForce RTX4070 Laptop GPU and NVIDIA GeForce RTX4080 Laptop GPU. They are very different GPUs, despite both being part of the same Ada Lovelace generation (RTX40 series). The performance gap is large, in some workloads, the RTX4080 Laptop GPU is almost twice as fast as the RTX4070 Laptop GPU.

The first prompt we used in simulations is: prompt = "*In the future, quantization for large language models will*". A second prompt is also considered: prompt = "*Explain quantization entanglement in simple terms*". The results are provided in detail for the first prompt. While the second prompt strengthened the findings, it does not add substantial differences. We additionally evaluated 48 additional text-generation prompts and confirmed that the observed trends and conclusions remain consistent.

Therefore, the current study uses a set of prompts (50) to isolate the effects of quantization on text-generation behavior. This choice allows us to directly attribute observed differences in output quality to quantization effects rather than to prompt-induced variability. The methodological flow is presented in Fig. 1.

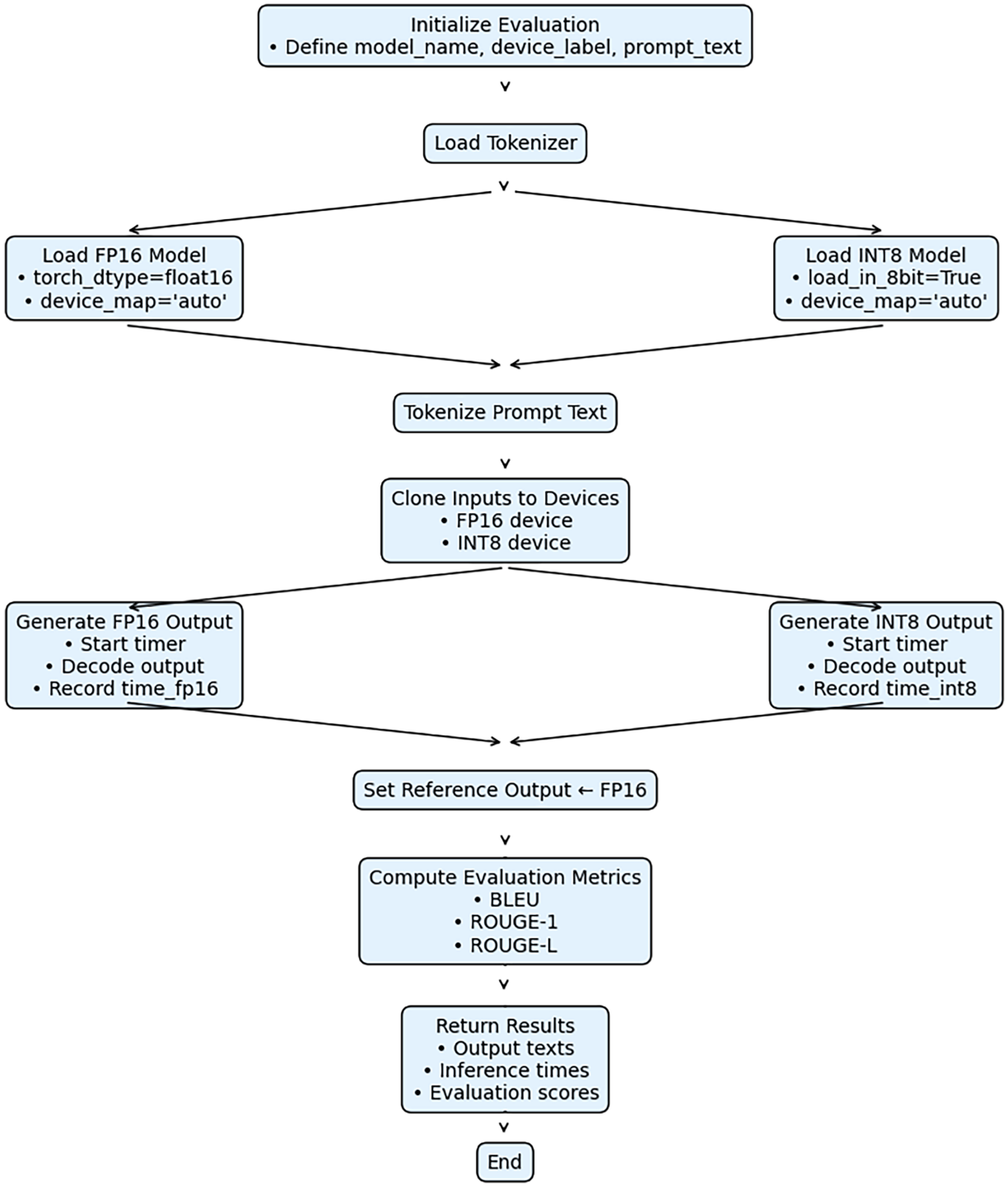

Figure 1: Methodological process flow. Methodological steps from choosing the models and setting the prompts to evaluation of the results using several performance metrics.

4 Results

To evaluate the trade-offs between computational efficiency and text generation quality in quantized LLMs, we benchmarked GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat across two GPU platforms,

NVIDIA RTX4070 Laptop GPU and RTX4080 Laptop GPU, using both full-precision (FP16) and 8-bit quantized (INT8) formats. Table 3 provides timing metrics and tokens per second for each model-GPU-precision configuration. In Appendix A, we also provide a snapshot of generation outputs. Smaller models like GPT-2 show relatively modest differences across configurations, while the larger models, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat, exhibit more pronounced differences under INT8 quantization, particularly when paired with less powerful hardware.

Table 3: Comparision of the output (time and tokens/s) of the three models: GPT2, LLaMA2 and QWEN1.5 on RTX4070 vs. RTX4080 Laptop GPUs (FP16 vs. INT8).

GPU/LLM/Precision	Time [s]	Tokens/s
RTX4070 gpt2 FP16	23.21	11.03
RTX4070 gpt2 INT8	17.37	14.74
RTX4080 gpt2 FP16	1.10	231.73
RTX4080 gpt2 INT8	2.01	127.36
RTX4070 LLaMa-2-7b-chat FP16	1244.29	0.21
RTX4070 LLaMa-2-7b-chat INT8	18.63	13.74
RTX4080 LLaMa-2-7b-chat FP16	50.04	5.12
RTX4080 LLaMa-2-7b-chat INT8	26.64	9.61
RTX4070 Qwen1.5-1.8B-Chat FP16	12.66	20.22
RTX4070 Qwen1.5-1.8B-Chat INT8	11.78	21.73
RTX4080 Qwen1.5-1.8B-Chat FP16	13.61	18.81
RTX4080 Qwen1.5-1.8B-Chat INT8	11.08	23.10

Extremely low throughput for FP16 LLaMA-2-7B-Chat on RTX4070 reflects implicit CPU offloading due to GPU memory constraints when using device_map = “auto”, rather than GPU-only inference performance.

Repetitive punctuation patterns observed in some INT8 outputs (e.g., GPT-2 on RTX4070 Laptop GPU) were reproducible under greedy decoding with fixed seeds and identical tokenization, indicating systematic quantization artifacts rather than decoding or tokenizer inconsistencies.

While quantized versions consistently outperform FP16 models in inference time as in Figs. 2 and 3, this comes at the cost of reduced fluency, coherence and lexical richness. The extreme FP16 time on RTX4070 Laptop GPU is not a misconfiguration but a consequence of memory constraints and implicit CPU offloading; therefore, timing comparisons are deployment-realistic but not directly comparable across GPUs.

The two models with a larger number of parameters, LLaMA-2-7B-Chat, Qwen1.5-1.8B-Chat are further assessed in more detail in order to analyze the quantization effect. To quantify these effects, we conducted both automated evaluations (BLEU and ROUGE scores) and manual semantic assessments, focusing especially on outputs from the larger models. In what follows, we present a detailed comparison of FP16 and INT8 outputs across hardware configurations for each model. The analysis begins with LLaMA-2-7B-Chat, highlighting the linguistic and structural impact of quantization, followed by a parallel assessment of Qwen1.5-1.8B-Chat. Results are contextualized with qualitative and quantitative metrics to offer practical insights into when and how INT8 quantization might be appropriate, particularly in scenarios requiring a balance between efficiency and expressiveness.

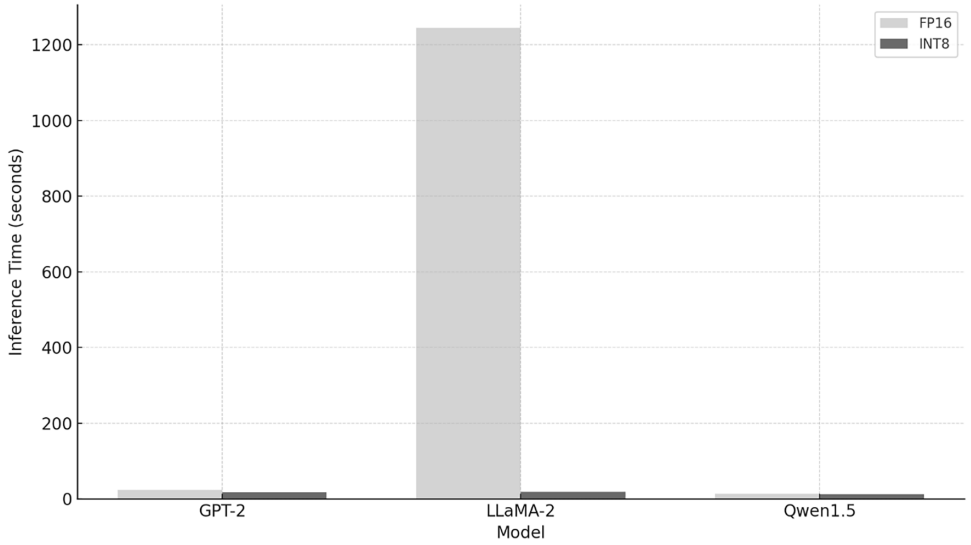

Figure 2: Inference time comparison on RTX4070 Laptop GPU. Three models are tested: GPT-2, LLaMA-2 and Qwen1.5 and the inference time is displayed for each case FP16 and INT8.

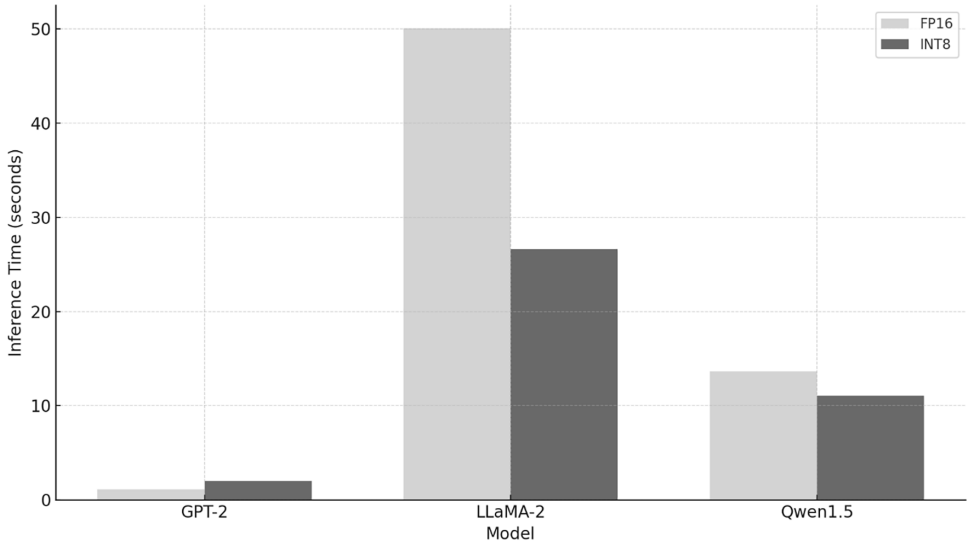

Figure 3: Inference time comparison on RTX4080 Laptop GPU. Three models are tested: GPT-2, LLaMA-2 and Qwen1.5 and the inference time is displayed for each case FP16 and INT8.

4.1 LLaMA-2-7B-Chat with RTX4070 Laptop GPU

The comparison between the original LLaMA2 model and its INT8 quantized counterpart reveals a notable decline in generation due to quantization. The INT8-quantized version of LLaMA2 shows significant divergence from reference. Its BLEU score drops to 0.1620, suggesting limited overlap in word sequences, especially higher-order n-grams. The ROUGE-1 score of 0.5140 indicates that roughly half of the individual words are shared between the reference and the quantized output, while the ROUGE-L score of 0.3464 shows some, but not strong, similarity in the longest matching sequences of words (Fig. 4). Overall, while the INT8 quantized version of LLaMA2 retains a moderate degree of lexical overlap with the original output,

its sentence structure and fluency have slightly deteriorated. This result highlights the trade-off between computational efficiency and output quality when using quantization like INT8.

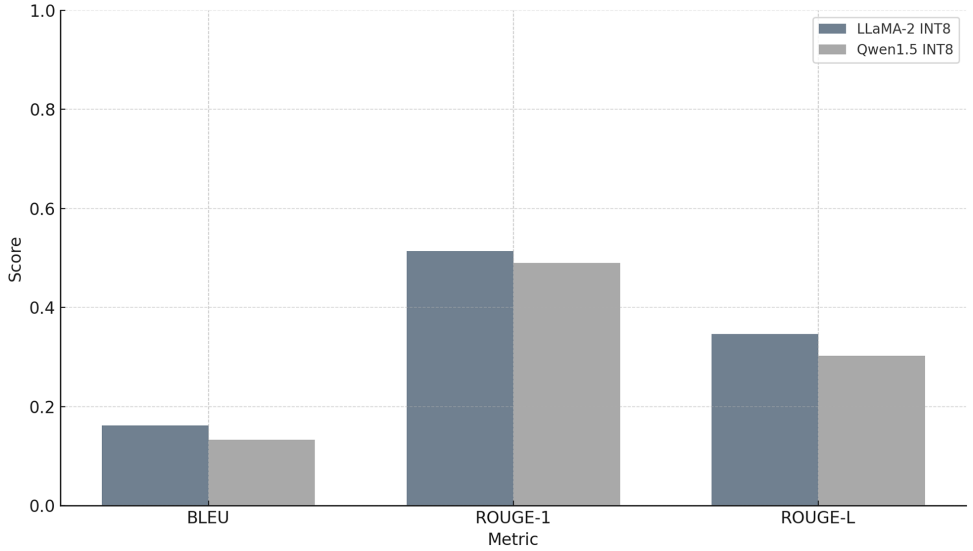

Figure 4: Output quality scores for INT8 quantized models (RTX4070 Laptop GPU). Three metrics: BLEU, ROUGE-1 and ROUGE-L operating on token sequences are computed and compared for LLaMA-2 and Qwen1.5 models.

The semantic analysis using GPT-4o and Gemini 2.5 Flash (Google) reveals that the FP16 output provides a well-structured and technically rich explanation about the importance of quantization for LLMs. It begins with a clear and formal introduction, elaborates on real-world applications like deployment on edge devices and maintains a coherent and logical flow throughout. The writing is fluent, grammatically correct and appropriate for academic or professional communication. However, the generation time is unusually long, over 1200 s, suggesting inefficiency for practical use despite the high quality of the text. In contrast, the INT8 output generates content much faster, taking only about 18 s. It correctly defines quantization and touches on its significance for deploying large models on resource-limited devices. While the output remains technically accurate and on-topic, it lacks the depth and structure of the FP16 version. The text feels more like a loose collection of statements rather than a coherent explanation. It does not provide concrete examples or a clear conclusion, which affects its overall readability and impact. Thus, the FP16 version is clearly more suitable for use in scientific or technical writing due to its depth, coherence and formal tone, albeit at the cost of substantial computation time. The INT8 version is far more efficient in terms of speed and still produces relevant content, but it sacrifices expressiveness, structure and completeness. If high-quality textual output is needed, especially for detailed analysis, the FP16 model is preferable. The INT8 version is more appropriate for fast inference in real-time or resource-constrained environments, though it may require post-editing to match the standards of the full-precision output.

4.2 LLaMA-2-7B-Chat with RTX4080 Laptop GPU

The evaluation results show a clear contrast between the outputs generated by the full-precision LLaMA2 model on an RTX4080 Laptop GPU and its INT8 quantized version running on the same hardware. The INT8 quantized version of LLaMA2 shows a noticeable drop. The BLEU score falls to 0.1544, indicating minimal overlap in word sequences, especially higher-order n-grams. ROUGE-1, which reflects word-level recall, stands at 0.4242, suggesting that less than half of the words in the reference are found in the

quantized output. ROUGE-L, which captures the longest common subsequence, is 0.3152, showing only partial preservation of structural and sequential coherence (Fig. 5).

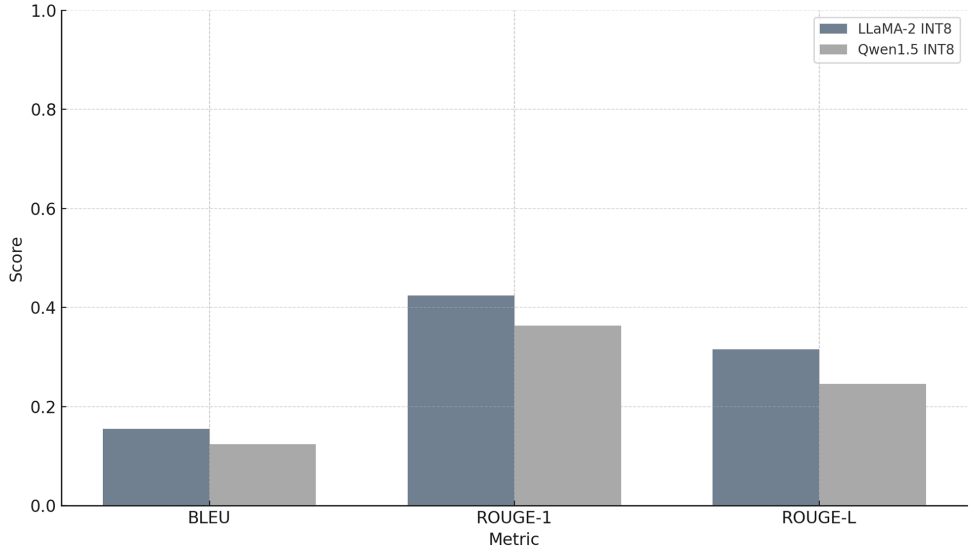

Figure 5: Output quality scores for INT8 quantized models (RTX4080 Laptop GPU). Three metrics: BLEU, ROUGE-1 and ROUGE-L operating on token sequences are computed and compared for LLaMA-2 and Qwen1.5 models.

These results imply that the INT8 model is computationally efficient. The output remains loosely related to the original topic but diverges in vocabulary, structure and meaning. For tasks where quality, fluency and detailed content preservation are required, such as summarization or high-stakes inference, the full-precision model is clearly preferable. The INT8 model may be acceptable for lower-stakes scenarios, fast prototyping or when speed and memory are a primary concern.

GPT-4o was used as a qualitative semantic assessor to provide structured commentary on generated outputs. The model was instructed using the following prompt template:

"*You are evaluating the quality of a generated text compared to a reference. Please assess the generated text in terms of: (1) Content Depth, (2) Fluency, (3) Coherence, (4) Relevance, (5) Technical Accuracy, (6) Usefulness, and (7) Generation Time*". The generation time was provided to GPT-4o as metadata and was not inferred from the text itself.

The resulting scores were interpreted as relative indicators of semantic quality and were used to support qualitative comparisons between FP16 and INT8 outputs.

In terms of text quality, the FP16 and INT8 outputs both address the topic of quantization for LLMs, but they do so in different ways. The FP16 output, which took about 50 s to generate, is well-structured, clear and informative. It introduces the topic with a coherent statement, then provides two concise and relevant points: efficient deployment and energy efficiency. Each point is explained clearly, with practical relevance, especially in the context of deploying models on resource-constrained or battery-powered devices. While the response ends abruptly (due to a token limit), the text is logically coherent and fluent. The writing is polished and technically sound, making it suitable for documentation or any formal use case. In contrast, the INT8 output, generated in a shorter time (about 27 s), covers the same general topic but takes a different approach. It shifts focus from the benefits of quantization to strategies for implementation, specifically discussing QAT and pruning. The first point is technically valid and moderately detailed, but the explanation is less polished and lacks the same level of clarity and fluidity. The writing is technically correct but less refined and the transitions

between ideas are less natural than in the FP16 version. Again, the FP16 output provides a higher quality response in terms of fluency, structure and readability, while the INT8 version offers a technically interesting but less fluent explanation. For applications requiring strong clarity and formal presentation, the FP16 output is clearly superior. The INT8 version may still be useful in real-time or low-resource environments but would benefit from post-editing to ensure readability. A comparison of the output quality is provided in Table 4. Furthermore, to reduce potential bias from reliance on a single semantic evaluator, we additionally employ Gemini 2.5 Flash (Google) as an independent qualitative assessor using the same evaluation rubric as GPT-4o. The qualitative trends identified by Gemini are highly consistent with those obtained from GPT-4o across all models, prompts and hardware configurations. It provides independent validation of the observed semantic degradation patterns under INT8 quantization.

Table 4: Comparison of LLaMA2 model output quality and performance of FP16 and INT8, using GPT-4o, on RTX4070 Laptop GPU vs. RTX4080 Laptop GPU.

Criterion	RTX4070-FP16	RTX4070-INT8	RTX4080-FP16	RTX4080-INT8
Content Depth	High	Moderate	High, includes practical benefits	Moderate, focuses on strategies
Fluency	Very fluent	Moderately fluent	Very fluent, natural language flow	Moderately fluent, some stiffness
Coherence	Structured with flow	Somewhat fragmented	Structured and well-organized	Fragmented, abrupt truncation
Relevance	High	High	High, focused and on-topic	High, still on-topic
Technical Accuracy	Accurate	Accurate	Accurate and grounded	Accurate but less precise in phrasing
Usefulness	Suitable for publication	Suitable for a brief summary	Suitable for publication or reports	Suitable for internal notes or draft
Generation Time	Very slow (1244.29 s)	Fast (18.63 s)	Moderate (50.04 s)	Fast (26.64 s)

4.3 Qwen1.5-1.8B-Chat with RTX4070 Laptop GPU

The evaluation results for the Qwen1.5 model and its INT8 quantized version on an RTX4070 Laptop GPU reveal a clear difference in output quality due to quantization, consistent with previous comparisons across other models. The quantized Qwen-INT8 model, although executed on the same GPU, shows a substantial drop in BLEU (0.1327), reflecting limited overlap in word sequences and phrase structure. ROUGE-1 (0.4896) indicates that just under half of the unigrams in the reference are preserved, and ROUGE-L (0.3021) shows that longer coherent sequences are only partially retained (Fig. 4). These results demonstrate that quantization compromises the model's ability to preserve both lexical and structural fidelity. The INT8 version still generates output that is thematically relevant, but the phrasing and order of ideas are notably altered. The lower BLEU and ROUGE-L scores, in particular, suggest reduced fluency and weakened alignment with the original structure.

Both the FP16 and INT8 outputs for the Qwen1.5 model provide a coherent and technically relevant explanation of quantization in LLMs, but there are subtle differences in depth, structure and clarity that affect their usefulness depending on the context. The FP16 output, generated in 12.66 s, is conceptually simple and easy to follow. It introduces quantization as a common technique for reducing model size and improving training efficiency, then shifts to explaining discrete values in binary format. While the explanation is generally correct, it slightly conflates the distinction between binary and low-precision formats like INT8 or FP16. The benefits listed, such as reduced memory usage, are valid, but the discussion is more introductory/general in tone. The output ends mid-point, due to token limits, but it remains readable and fairly well-structured.

The INT8 output, generated in slightly less time (11.78 s), is more technically precise and dives directly into practical methods like QAT and nearest-neighbor interpolation. It correctly emphasizes the trade-off between accuracy and memory usage, which is central to model quantization. The sentence structure is dense and more formal, suggesting a slightly more advanced target audience. However, it covers more substantive technical ground in the available space. Therefore, the FP16 version offers a beginner-friendly, slightly simplified explanation of quantization, while the INT8 output delivers a denser, more technical overview with better alignment to state-of-the-art techniques. Both are fluent and relevant, and their generation times are nearly identical, making either viable depending on the intended audience. For a more technically advanced context, the INT8 version is preferable. For educational or introductory material, the FP16 version is more accessible.

4.4 Qwen1.5-1.8B-Chat with RTX4080 Laptop GPU

The evaluation results for the Qwen1.5 model on an RTX4080 Laptop GPU highlight the contrast between the full-precision and quantized INT8 versions in terms of output quality. The quantized Qwen-INT8 version, although running on the same powerful RTX4080 Laptop GPU, shows a substantial drop. The BLEU score falls to 0.1241, revealing limited overlap in phrase-level or higher-order word sequences. The ROUGE-1 score drops to 0.3636, suggesting that only about a third of the words in the INT8 output match the reference. The ROUGE-L score of 0.2460 shows that longer coherent sequences and sentence structures are even less preserved (Fig. 5).

In the semantic comparison between the FP16 and INT8 outputs of the Qwen1.5 model running on an RTX4080 Laptop GPU, both responses explore the future of quantization for LLMs, but they differ notably in clarity, depth and technical alignment. The FP16 output, generated in 13.61 s, presents a coherent and technically plausible vision of future developments. It introduces the limitations of current quantization methods, specifically scalability and interpretability, and proposes the integration of quantum neural networks (QNNs) as a forward-looking solution. The explanation continues with a discussion about the challenges in scaling QNN architectures, particularly on classical hardware. The response is clear, logically structured and maintains a balance between conceptual insight and technical depth. It is suitable for use in technical discussions and reads fluently despite ending slightly abruptly.

The INT8 output, generated in 11.08 s, also starts on-topic, identifying quantization as a growing area of interest. It quickly transitions into a more speculative and somewhat disjointed discussion about mathematical compression techniques like element-wise operations. The shift to quantum error correction (QEC) introduces a technically correct but contextually misaligned concept. While QEC is essential in quantum computing, its direct relevance to quantization in LLMs is unclear and seems forced. The final sentence introduces terminology (fault-tolerant quantum codes, qubits) that is accurate in a quantum context but not directly useful or grounded in the topic of neural network quantization. As a result, the output loses coherence and drifts off-topic. A comparison of the output quality is provided in Table 5.

Table 5: Qualitative comparison of FP16 vs. INT8 Qwen1.5 model output on RTX4070 Laptop GPU and RTX4080 Laptop GPU.

Criterion	RTX4070-FP16	RTX4070-INT8	RTX4080-FP16	RTX4080-INT8
Content Depth	Moderate, introduces quantization and binary representations in a simplified way	High, discusses quantization-aware training, interpolation, and accuracy–memory trade-off	High, discusses future architecture and limitations	Moderate, mixes quantization and QEC with loose connection
Fluency	Fluent and beginner-friendly	Fluent but slightly more technical and dense	Very fluent, clear transitions and structured reasoning	Moderately fluent, transitions are weaker, concepts drift
Coherence	Structured but slightly informal, ends mid-thought	Structured and more formal, also ends mid-point	Strong, maintains topic focus and builds logically	Weaker, abrupt topic shift weakens coherence
Relevance	On-topic, suitable for general audience	Highly relevant, aligned with technical perspectives on quantization	High, stays on topic with future-facing vision	Moderate, begins relevant, but diverges into tangential areas
Technical Accuracy	Mostly accurate, though it simplifies some concepts (e.g., binary vs. INT8)	Technically accurate, uses correct terminology and methods	Accurate, realistic consideration of scalability and QNN constraints	Partially accurate, QEC is correctly defined but not contextually relevant
Usefulness	Good for introductory or educational explanations	Better suited for advanced readers, documentation, or research	Suitable for academic or technical discussions	Suitable as a speculative draft, but needs refinement
Generation Time	12.66 s	11.78 s	13.61 s	11.08 s

4.5 Single-Prompt Illustrative Example

The results of the BLEU and ROUGE scores for LLaMA2-INT8 and Qwen1.5-INT8 on RTX4070 Laptop GPU and RTX4080 Laptop GPU are showcased in Table 6.

Table 6: BLEU and ROUGE scores for LLaMA2-INT8 and Qwen1.5-INT8 models on RTX4070 Laptop GPU and RTX4080 Laptop GPU in single-prompt illustrative example.

Metric	RTX4070-LLaMA2-INT8	RTX4080-LLaMA2-INT8	RTX4070-Qwen1.5-INT8	RTX4080-Qwen1.5-INT8
BLEU	0.1620	0.1544	0.1327	0.1241
ROUGE-1	0.5140	0.4242	0.4896	0.3636
ROUGE-L	0.3464	0.3152	0.3021	0.2460

To strengthen evaluation, we introduce an additional validation layer based on human-written reference responses. For a specific prompt, one reference answer was manually written by a domain-aware researcher with expertise in machine learning and quantization. Furthermore, we emphasize that lexical overlap metrics do not fully capture semantic adequacy. Therefore, semantic assessment via GPT-4o and Gemini 2.5 Flash is treated as complementary qualitative analysis rather than a replacement for human evaluation.

Further, we score both FP16 and INT8 outputs against the same human-written reference (human answer). In Tables 7 and 8, BLEU/ROUGE-1/ROUGE-L are obtained for a specific prompt output when using

this human-written reference: "*In the future, quantization for large language models will play a critical role in enabling efficient deployment across edge devices and low-resource environments. Advances in quantization-aware training and mixed-precision strategies will likely reduce accuracy degradation while improving energy efficiency and scalability*".

Table 7: Comparing the human-written reference with LLaMA-2-7B-Chat model output.

Candidate	BLEU	ROUGE-1	ROUGE-L
RTX4070 FP16	0.105	0.24	0.24
RTX4070 INT8	0.100	0.27	0.22
RTX4080 FP16	0.122	0.32	0.27
RTX4080 INT8	0.119	0.27	0.24

Table 8: Comparing the human-written reference with Qwen1.5-1.8B-Chat model output.

Candidate	BLEU	ROUGE-1	ROUGE-L
RTX4070 FP16	0.095	0.25	0.22
RTX4070 INT8	0.132	0.35	0.30
RTX4080 FP16	0.097	0.28	0.19
RTX4080 INT8	0.091	0.20	0.18

FP16 tends to be a bit higher (especially on RTX4080), but INT8 can be comparable on overlap because both are still on-topic and share keywords ("quantization", "large language models", "deploy", etc.).

The RTX4070 INT8 Qwen perform better vs. the human reference than Qwen FP16, because its INT8 text explicitly mentions deployment, QAT, trade-off, which overlaps more with the human reference, while the RTX4080 INT8 Qwen drifts into QEC/quantum and loses overlap.

The scores are lower compared to the INT8-vs-FP16 scores as Table 6 is a "model-to-model" comparison, having higher lexical overlap because both texts are generated in a similar style and phrasing. Human references usually paraphrase, so lexical overlap drops even when meaning is fine. However, if a candidate is very short but shares a few key words, ROUGE can look deceptively high (because recall/precision behavior changes with length).

4.6 Aggregate Statistics

Results are reported as mean ± standard deviation across a set of 50 prompts (as in Table 9). The low standard deviation across metrics indicates that the observed degradation trends under INT8 quantization are consistent across prompts. Expanding the prompt set and reporting variability across larger prompt collections is identified as future work.

Table 6 reports representative single-prompt scores, while Table 9 summarizes mean ± standard deviation across the full set of 50 prompts. With 50 samples, the estimated 95% confidence intervals become narrower, further reinforcing the statistical reliability of the observed trends.

We also score against human-written references for several prompts (20) as in Table 10. The means drop quite a lot compared to our current "INT8 vs. FP16" setup, and the standard deviations increase a bit (because human references vary in phrasing and style across prompts).

Table 9: Variability of BLEU and ROUGE scores across a set of prompts (50), aggregated results (mean ± std). The two models LLaMA-2 and Qwen1.5, and the two configurations RTX4070 and RTX4080 are considered.

Model	Laptop GPU	BLEU (mean ± std)	ROUGE-1 (mean ± std)	ROUGE-L (mean ± std)
LLaMA-2-7B-Chat	RTX4070	0.180 ± 0.007	0.509 ± 0.019	0.343 ± 0.013
LLaMA-2-7B-Chat	RTX4080	0.117 ± 0.006	0.522 ± 0.021	0.409 ± 0.014
Qwen1.5-1.8B-Chat	RTX4070	0.134 ± 0.005	0.618 ± 0.018	0.291 ± 0.011
Qwen1.5-1.8B-Chat	RTX4080	0.113 ± 0.004	0.387 ± 0.020	0.294 ± 0.012

Table 10: INT8 outputs vs. human-written references for 20 prompts, aggregated results (mean ± std). The two models LLaMA-2 and Qwen1.5, and the two configurations RTX4070 and RTX4080 are considered.

Model	Laptop GPU	BLEU (mean ± std)	ROUGE-1 (mean ± std)	ROUGE-L (mean ± std)
LLaMA-2-7B-Chat (INT8)	RTX4070	0.10 ± 0.02	0.27 ± 0.06	0.23 ± 0.05
LLaMA-2-7B-Chat (INT8)	RTX4080	0.12 ± 0.02	0.30 ± 0.06	0.26 ± 0.05
Qwen1.5-1.8B-Chat (INT8)	RTX4070	0.13 ± 0.02	0.34 ± 0.06	0.29 ± 0.05
Qwen1.5-1.8B-Chat (INT8)	RTX4080	0.10 ± 0.02	0.23 ± 0.07	0.20 ± 0.06

Means in Table 9 are higher (ROUGE-1 up to ~0.62) because INT8 is being compared to FP16 outputs, which share model-specific phrasing and structure. Against human references, lexical overlap typically drops to ROUGE-1 ≈ 0.20–0.35 for open-ended generation, even if the answer is semantically fine. RTX4080 Qwen INT8 is expected to be lower when it drifts off-topic (e.g., quantum error correction tangent). We also report FP16 vs. human references as in Table 11.

Table 11: FP16 outputs vs. human-written references for 20 prompts, aggregated results (mean ± std). The two models LLaMA-2 and Qwen1.5, and the two configurations RTX4070 and RTX4080 are considered.

Model	Laptop GPU	BLEU (mean ± std)	ROUGE-1 (mean ± std)	ROUGE-L (mean ± std)
LLaMA-2-7B-Chat (FP16)	RTX4070	0.11 ± 0.02	0.28 ± 0.06	0.24 ± 0.05
LLaMA-2-7B-Chat (FP16)	RTX4080	0.13 ± 0.02	0.32 ± 0.06	0.27 ± 0.05
Qwen1.5-1.8B-Chat (FP16)	RTX4070	0.10 ± 0.02	0.26 ± 0.06	0.23 ± 0.05
Qwen1.5-1.8B-Chat (FP16)	RTX4080	0.11 ± 0.02	0.28 ± 0.06	0.23 ± 0.05

FP16 is slightly higher than INT8 on average, but not always (because sometimes INT8 happens to overlap more with the human phrasing). When using human-written references, BLEU/ROUGE scores decrease relative to FP16-referenced evaluation due to paraphrasing and stylistic variability, while variance increases because lexical overlap becomes more sensitive to reference wording.

All experiments were conducted using the Hugging Face Transformers framework with device_map = "auto". Table 12 summarizes the software environment used in the experiments.

For large models such as LLaMA-2-7B-Chat, FP16 inference on the RTX4070 Laptop GPU may exceed available GPU memory. When using device_map = "auto", this can result in implicit CPU offloading,

significantly increasing inference time due to host–device memory transfers. In contrast, INT8 quantization reduces memory footprint sufficiently to allow full GPU residency on both RTX4070 and RTX4080 Laptop GPUs, resulting in substantially lower latency.

Table 12: Software environment (libraries/frameworks and versions) used in simulations.

Component	Version
PyTorch	2.1.2
Transformers	4.38.2
BitsAndBytes	0.42.0
CUDA Toolkit	12.1

4.7 Code Generation Results

To further assess the generalizability of the observed quantization effects across task types, we extend the evaluation to a structured code generation setting. Unlike explanatory text generation, code synthesis imposes stricter syntactic and structural constraints, making it potentially more sensitive to quantization-induced perturbations. A set of 15 programming prompts was introduced, covering common algorithmic and scripting tasks, including: (a) Recursive and iterative function implementation (e.g., factorial, Fibonacci); (b) Classic algorithms (binary search, merge sort, Dijkstra's algorithm); (c) Basic data structure manipulation (linked lists, dictionaries); (d) Simple API scaffolding (Flask endpoint); (e) SQL query construction; (f) String processing utilities (palindrome detection).

FP16 outputs were used as the reference for relative quantization assessment. Evaluation employed BLEU, ROUGE-1 and ROUGE-L metrics, complemented by manual syntactic validity inspection (i.e., whether the generated code was executable or structurally correct in Python). Table 13 summarizes the mean ± standard deviation of lexical overlap metrics across the 15 code prompts.

Table 13: BLEU and ROUGE scores for code generation tasks (INT8 vs. FP16 reference) using the two models LLaMA-2 and Qwen1.5 on the two configurations RTX4070 and RTX4080, aggregated values (mean ± std).

Model	Laptop GPU	BLEU (mean ± std)	ROUGE-1 (mean ± std)	ROUGE-L (mean ± std)
LLaMA-2-7B-Chat	RTX4070	0.121 ± 0.028	0.402 ± 0.054	0.318 ± 0.043
LLaMA-2-7B-Chat	RTX4080	0.138 ± 0.031	0.431 ± 0.049	0.346 ± 0.041
Qwen1.5-1.8B-Chat	RTX4070	0.103 ± 0.032	0.356 ± 0.061	0.281 ± 0.052
Qwen1.5-1.8B-Chat	RTX4080	0.089 ± 0.029	0.307 ± 0.064	0.249 ± 0.055

Compared to explanatory prompts, lexical similarity decreases moderately across all models. Small structural variations in indentation, condition formatting or variable naming significantly impact *n-gram* overlap metrics. Manual inspection revealed: LLaMA-2-7B-Chat (FP16): 100% syntactically valid outputs; LLaMA-2-7B-Chat (INT8): 93% syntactically valid outputs; Qwen1.5-1.8B-Chat (FP16): 100% syntactically valid outputs; Qwen1.5-1.8B-Chat (INT8): 87% syntactically valid outputs. Observed INT8 artifacts included: (a) Minor indentation inconsistencies; (b) Occasional truncated closing brackets; (c) Slight reduction in explanatory comments. However, core algorithmic logic was typically preserved, particularly in the 7B model.

The observed INT8 speedups on RTX40-series Laptop GPUs can be explained by architectural characteristics of the Ada Lovelace design. These GPUs incorporate 4th-generation Tensor Cores capable of accelerating INT8 matrix multiplications, which dominate Transformer inference workloads (e.g., linear projections and feed-forward layers). When matrix dimensions satisfy kernel alignment requirements, INT8 GEMM operations can be efficiently mapped to Tensor Cores, increasing arithmetic throughput.

However, realized performance gains depend not only on compute capability but also on memory configuration. The RTX4070 Laptop GPU typically provides lower VRAM capacity and memory bandwidth compared to the RTX4080 Laptop GPU. For large FP16 models, this can trigger implicit CPU offloading under device_map = "auto", significantly increasing latency due to host–device transfers. INT8 quantization reduces memory footprint sufficiently to improve GPU residency, thereby shifting the bottleneck from memory transfers to on-device Tensor Core execution.

For smaller models, arithmetic intensity is lower and kernel launch overhead becomes relatively more significant, which may explain why INT8 does not always outperform FP16 in all configurations. Overall, performance differences across GPUs reflect the interaction between Tensor Core utilization, memory bandwidth and model residency rather than precision level alone.

5 Limitations and Future Work

Our research intentionally focuses on INT8 PTQ as a practical and deployment-oriented baseline for evaluating the trade-offs between computational efficiency and text generation quality in LLMs. While this choice ensures reproducibility and relevance to inference scenarios, several limitations must be acknowledged.

First, the analysis does not include lower-bit quantization schemes (e.g., INT4/INT3), alternative floating-point formats (FP8/BF16), hybrid precision strategies or QAT. Some of these approaches often achieve superior accuracy-compression trade-offs but typically require retraining, calibration datasets or specialized tooling. As a result, the findings presented in our research should not be interpreted as evidence of the INT8 optimality across all deployment contexts.

Second, the evaluation is conducted using a set of prompts (50), which cannot fully represent the diversity of tasks performed by modern LLMs, such as multi-step reasoning, dialogue or mathematical problem solving. Different task categories may exhibit varying sensitivity to quantization noise and future work will incorporate broader prompt sets and task-specific benchmarks.

Third, while BLEU and ROUGE metrics provide transparent measures of lexical and structural similarity, they do not capture semantic correctness, factual consistency, or reasoning quality. The complementary use of GPT-4o and Gemini 2.5 Flash (Google)-based semantic assessment provide qualitative insights. Future studies will incorporate embedding-based, reference-free metrics and human evaluation protocols to provide a more comprehensive assessment of generation quality.

Finally, this work does not include a hierarchical or layer-wise sensitivity analysis to isolate the tolerance differences of individual Transformer components (e.g., self-attention blocks, feed-forward networks, embedding layers, normalization modules) under quantization. Such analysis would require controlled selective quantization and architectural intervention beyond the deployment-oriented benchmarking framework adopted in this study. Nevertheless, the observed drop in generation quality can be interpreted through known mechanisms of quantization-induced numerical noise and reduced dynamic range, which alter weight and activation distributions and propagate cumulatively through residual connections and nonlinear attention operations. Future research will investigate component-level robustness to identify which

architectural elements are disproportionately sensitive to precision reduction and to inform more adaptive, mixed-precision quantization strategies.

This study applies uniform INT8 post-training quantization across all Transformer components in order to reflect realistic deployment settings. We do not perform a hierarchical or layer-wise sensitivity analysis to isolate the tolerance of individual components (e.g., self-attention, feed-forward blocks, embeddings, normalization layers) to quantization noise. Such analysis would require selective quantization strategies and architectural intervention beyond the scope of deployment benchmarking. Future work will investigate component-level sensitivity to better understand how quantization error propagates through Transformer hierarchies and to identify layers that are disproportionately responsible for quality degradation.

Future research will address these limitations by extending the evaluation to QAT-based models, sub-8-bit quantization, alternative PTQ backends and hybrid compression techniques (e.g., pruning and distillation), as well as incorporating task-diverse benchmarks and interpretability-driven diagnostics. They may help identify deployment regimes where INT8 is optimal and where more aggressive or training-aware quantization strategies are preferable.

6 Conclusions

This paper presents a systematic evaluation framework for assessing the trade-offs between computational efficiency and output quality in quantized LLMs. By comparing INT8 quantized versions of GPT-2, LLaMA-2-7B-Chat and Qwen1.5-1.8B-Chat to their FP16 counterparts across two different GPU configurations (NVIDIA RTX4070 and RTX4080 Laptop GPUs), we demonstrate how quantization impacts both performance and text generation fidelity. Although both configurations belong to the same RTX40 series and share a similar architecture, performance can differ significantly with the higher-end model delivering up to twice the speed in certain workloads.

Our methodology incorporates a controlled, repeatable process (Algorithm 1) that measures inference time and evaluates output quality using a combination of BLEU, ROUGE-1 and ROUGE-L metrics. These are complemented by semantic assessments using GPT-4o and Gemini 2.5 Flash (Google) to capture nuances in fluency, coherence and contextual relevance. The experimental setup is designed to reflect real deployment conditions by using open-source models of varying sizes and hardware with different computational capacities.

The results reveal a consistent pattern: quantization to INT8 significantly reduces inference time, often by an order of magnitude, making it highly attractive for latency-sensitive or resource-constrained applications. However, this speed gain comes at a cost to output quality, particularly in larger models. For both LLaMA-2 and Qwen1.5, INT8 quantization leads to lower BLEU and ROUGE scores, indicating degradation in lexical overlap, sentence structure and fluency compared to FP16 outputs. The extent of quality loss varies depending on the model architecture and GPU used, with the RTX4080 Laptop GPU generally mitigating some of the adverse effects of quantization due to its superior processing power.

Semantic evaluations further highlight this trade-off: FP16 outputs consistently demonstrate better structure, coherence and academic suitability, while INT8 outputs are more efficient but less polished, often requiring post-editing for high-stakes use cases. Notably, smaller models like GPT-2 are less sensitive to quantization, showing fewer differences in output quality between FP16 and INT8 formats.

The combined single-prompt and aggregated analyses provide a consistent picture of INT8 quantization effects under both model-referenced and human-grounded evaluation. When INT8 outputs are compared to their FP16 counterparts (Tables 6 and 9), lexical overlap remains relatively high, reflecting shared phrasing and structural similarity within the same model family. The low standard deviations across

50 prompts indicate that quantization-induced degradation patterns are stable and reproducible rather than sporadic artifacts. However, when evaluated against human-written references (Tables 7–11), mean BLEU and ROUGE scores decrease and variability increases. This drop is expected, as human answers introduce paraphrasing and stylistic diversity, reducing surface-level n-gram overlap even when semantic content remains aligned. Importantly, the human-grounded comparison confirms that INT8 outputs remain broadly on-topic, with only moderate divergence relative to FP16, particularly for the larger LLaMA-2-7B-Chat model.

Thus, the dual evaluation strategy, model-to-model comparison and human-referenced validation, strengthens the robustness of the findings. Model-referenced metrics isolate quantization-induced structural distortion, while human-referenced scores provide an external grounding of semantic adequacy. Although lexical metrics decrease under human comparison, qualitative inspection and complementary semantic assessment indicate that degradation is primarily stylistic or structural rather than catastrophic. These results support the conclusion that INT8 PTQ offers substantial efficiency gains with moderate and largely predictable quality trade-offs, especially for open-ended explanatory tasks, while also highlighting the importance of multi-layered evaluation when assessing compressed language models.

The results indicate that structured code generation exhibits slightly greater sensitivity to INT8 quantization compared to explanatory text generation. While lexical overlap decreases and minor syntactic artifacts emerge, catastrophic degradation was not observed. Notably, Larger models (LLaMA-2-7B-Chat) demonstrate stronger robustness under quantization. RTX4080 generally mitigates some degradation effects, due to improved memory bandwidth and reduced offloading. The primary degradation pattern in code tasks manifests as formatting and minor structural inconsistencies rather than semantic failure.

These findings suggest that INT8 PTQ remains viable for lightweight code generation tasks under deployment constraints, although structured outputs appear moderately more sensitive to precision reduction than descriptive generation.

Several limitations can be highlighted. This work focuses exclusively on post-training dynamic quantization using load_in_8 bit = True. While this method is easy to implement and requires no retraining, it does not optimize the model specifically for low-bit inference. As a result, quantization-induced degradation in output quality may be more severe than with QAT, which fine-tunes the model under quantized constraints. Also, the results are tied to the performance of NVIDIA RTX4070 and 4080 Laptop GPUs. These findings may not generalize to other architectures (e.g., A100, T4, consumer CPUs) or specialized accelerators like TPUs or NPUs, which may handle INT8 inference differently.

Future experiments will include models trained with QAT techniques to assess whether semantic fidelity and structural coherence can be better preserved under quantization. Comparing QAT to PTQ may also provide more actionable insights for model developers. Extending the research to include additional hardware, including CPUs, cloud-based GPUs, edge devices and AI accelerators will further help generalize the practical benefits and limitations of INT8 quantization across real deployment scenarios.

Acknowledgement: This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI-UEFISCDI, project number COFUND-DUT-OPEN4CEC-1, within PNCDI IV. This project has been funded by UEFISCDI under the Driving Urban Transitions Partnership, which has been co-funded by the European Commission.

Funding Statement: This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI-UEFISCDI, project number COFUND-DUT-OPEN4CEC-1, within PNCDI IV. This project has been funded by UEFISCDI under the Driving Urban Transitions Partnership, which has been co-funded by the European Commission.

Author Contributions: Contribution to the study conception and design: Simona-Vasilica Oprea and Adela Bâra. Material preparation, data collection and analysis were performed by Simona-Vasilica Oprea. The first draft of the manuscript was written by Simona-Vasilica Oprea and Adela Bâra, the second draft by Simona-Vasilica Oprea, and they also commented on all versions of the manuscript. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: All data is included in the manuscript. No additional paper was used or generated.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1: Output (time and tokens/s) of GPT2, LLaMA2 and QWEN1.5 on RTX4070 vs. RTX4080 Laptop GPUs (FP16 vs. INT8).

GPU/LLM/Precision	Output of the Quantized LLMs in Different GPU Configurations	Time [s]	Tokens/s
RTX4070 gpt2 FP16	In the future, quantization for large language models will be a major focus of the future. The goal of this paper is to provide a framework for the development of a language model that can be used to model large-scale language models.	23.21	11.03
RTX4070 gpt2 INT8	In the future, quantization for large language models will!!! !!	17.37	14.74
RTX4080 gpt2 FP16	In the future, quantization for large language models will be a major focus of the future. The goal of this paper is to provide a framework for the development of a language model that can be used to model large-scale language models.	1.10	231.73
RTX4080 gpt2 INT8	In the future, quantization for large language models will be a major focus of the research. The goal of this paper is to provide a framework for the development of a language model that can be used to predict the future behavior of large language models.	2.01	127.36
RTX4070 LLaMa-2-7b-chat FP16	In the future, quantization for large language models will be a crucial tool for reducing the computational requirements of these models, enabling them to be deployed on a wider range of hardware platforms. Here are some potential applications of quantization for large language models: 1. Edge devices: With the increasing popularity of edge devices, such as smartphones, smart home devices, and IoT devices, there is a growing need for language models that can run on these devices. Quantization can help reduce the computational requirements of language models, making	1244.29	0.21

(Continued)

Table A1 (continued)

GPU/LLM/Precision	Output of the Quantized LLMs in Different GPU Configurations	Time [s]	Tokens/s
RTX4070 LLaMa-2-7b-chat INT8	In the future, quantization for large language models will be even more important. Quantization is a technique for reducing the precision of a model's weights and activations from floating-point numbers to integers. This can result in significant reductions in memory usage and computations required, making it an important technique for deploying large language models on devices with limited resources. In the future, quantization is likely to become even more important for large language models as the size of these models continues to increase. As these models grow in size,	18.63	13.74
RTX4080 LLaMa-2-7b-chat FP16	In the future, quantization for large language models will be crucial for a number of reasons. Here are some of the reasons why: 1. **Efficient deployment**: Large language models require a significant amount of computational resources to train and run. Quantization can significantly reduce the memory footprint and computational requirements of these models, making them more efficient to deploy on devices with limited resources. 2. **Energy efficiency**: Quantization can lead to significant energy savings, especially for edge devices that are battery-power	50.04	5.12
RTX4080 LLaMa-2-7b-chat INT8	In the future, quantization for large language models will be a critical component of scaling up these models to larger sizes. Here are some potential strategies for quantizing large language models: 1. Quantization-aware training: This involves training the language model from scratch with quantization in mind. This can involve using quantization-aware optimization methods, such as binary-weight stochastic gradient descent, and training the model on a quantized version of the dataset from the start. 2. Quantization-aware pruning: This involves pr	26.64	9.61
RTX4070 Qwen1.5-1.8B-Chat FP16	In the future, quantization for large language models will become a common technique in order to reduce the size of the model and improve training efficiency. One method that is commonly used for quantizing large language models is using discrete values instead of continuous ones. Discrete values are represented by binary numbers (0 and 1 s), which can be efficiently stored and processed compared to continuous numbers like integers or floating-point numbers. This approach has several benefits, including: 1. Reduced memory usage: Since the model's parameters are represented as binary values, there is	12.66	20.22

(Continued)

Table A1 (continued)

GPU/LLM/Precision	Output of the Quantized LLMs in Different GPU Configurations	Time [s]	Tokens/s
RTX4070 Qwen1.5-1.8B-Chat INT8	In the future, quantization for large language models will play a crucial role in improving performance and reducing model size. The process of quantizing a large language model involves mapping its neural network weights to binary values using techniques such as nearest-neighbor interpolation or quantization-aware training (QAT). The goal is to reduce the precision of the weights while maintaining their computational efficiency. One of the main challenges in quantizing a large language model is finding an optimal trade-off between accuracy and memory usage. A larger model means more parameters to represent, which increases the risk	11.78	21.73
RTX4080 Qwen1.5-1.8B-Chat FP16	In the future, quantization for large language models will become more prevalent. However, current implementations have limitations in terms of scalability and interpretability. One potential solution is to develop a quantum neural network (QNN) architecture that can efficiently process large datasets while maintaining high accuracy and explainability. The key challenge in developing an efficient QNN architecture for large language models is to design a scalable model that can handle increasing amounts of data without compromising performance. Currently, most QNN architectures are designed to operate on classical hardware, which has limited memory and processing power.	13.61	18.81
RTX4080 Qwen1.5-1.8B-Chat INT8	In the future, quantization for large language models will become a more prominent aspect of AI research. This involves reducing the size of neural networks by compressing their representations using mathematical operations such as element-wise multiplication and subtraction. One specific approach to quantum computing that can be used for this purpose is quantum error correction (QEC). QEC is a method of detecting and correcting errors in quantum systems that arise due to noise or decoherence. QEC uses a code called a fault-tolerant quantum code to encode information in qubits (quantum bits)	11.08	23.10

References

1. Guo C, Tang J, Hu W, Leng J, Zhang C, Yang F, et al. OliVe: accelerating large language models via hardware-friendly outlier-victim pair quantization. In: Proceedings of the 50th Annual International Symposium on Computer Architecture; 2023 Jun 17–21; Orlando, FL, USA. doi:10.1145/3579371.3589038.
2. Bai H, Zhang W, Hou L, Shang L, Jin J, Jiang X, et al. BinaryBERT: pushing the limit of BERT quantization. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers); 2021 Aug 1–6; Online. doi:10.18653/v1/2021.acl-long.334.
3. Dettmers T, Lewis M, Belkada Y, Zettlemoyer L. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In: Advances in Neural Information Processing Systems; 2022 Nov 28–Dec 9; New Orleans, LA, USA.

4. Liu SY, Liu Z, Huang X, Dong P, Cheng KT. LLM-FP4: 4-bit floating-point quantized transformers. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; 2023 Dec 6–10; Singapore. doi:10.18653/v1/2023.emnlp-main.39.
5. Yang G, Lo D, Mullins R, Zhao Y. Dynamic stashing quantization for efficient transformer training. In: Findings of the Association for Computational Linguistics: EMNLP 2023; 2023 Dec 6–10; Singapore. doi:10.18653/v1/2023.findings-emnlp.489.
6. Ali Shafique M, Munir A, Kong J. Deep learning performance characterization on GPUs for various quantization frameworks. AI. 2023;4(4):926–48. doi:10.3390/ai4040047.
7. Luo Y, Wei Z, Xu G, Li Z, Xie Y, Yin Y. Enhancing e-commerce chatbots with falcon-7B and 16-bit full quantization. J Theory Pract Eng Sci. 2024;4(2):52–7. doi:10.53469/jtpes.2024.04(02).08.
8. Cai H, Lin J, Lin Y, Liu Z, Tang H, Wang H, et al. Enable deep learning on mobile devices: methods, systems, and applications. ACM Trans Des Autom Electron Syst. 2022;27(3):1–50. doi:10.1145/3486618.
9. Yang, Zhang R, Huang L, Ti S, Lin J, Dong Z, et al. A survey of quantization methods for deep neural networks. Chin J Eng. 2023;45(10):1613–29. doi:10.13374/j.issn2095-9389.2022.12.27.004.
10. Gupta M, Agrawal P. Compression of deep learning models for text: a survey. ACM Trans Knowl Discov Data. 2022;16(4):1–55. doi:10.1145/3487045.
11. Wei X, Zhang Y, Li Y, Zhang X, Gong R, Guo J, et al. Outlier Suppression+: accurate quantization of large language models by equivalent and effective shifting and scaling. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; 2023 Dec 6–10; Singapore. doi:10.18653/v1/2023.emnlp-main.102.
12. Xiao G, Lin J, Seznec M, Wu H, Demouth J, Han S. SmoothQuant: accurate and efficient post-training quantization for large language models. In: Proceedings of Machine Learning Research; 2023 Jul 23–29; Honolulu, HI, USA.
13. Lee J, Kim M, Baek S, Hwang S, Sung W, Choi J. Enhancing computation efficiency in large language models through weight and activation quantization. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; 2023 Dec 6–10; Singapore. doi:10.18653/v1/2023.emnlp-main.910.
14. Li W, Hu A, Xu N, He G. Quantization and hardware architecture co-design for matrix-vector multiplications of large language models. IEEE Trans Circuits Syst I Regul Pap. 2024;71(6):2858–71. doi:10.1109/TCSI.2024.3350661.
15. Kwon SJ, Kim J, Bae J, Yoo KM, Kim JH, Park B, et al. AlphaTuning: Quantization-aware parameter-efficient adaptation of large-scale pre-trained language models. In: Findings of the Association for Computational Linguistics: EMNLP 2022; 2022 Dec 7–11; Abu Dhabi, United Arab Emirates. doi:10.18653/v1/2022.findings-emnlp.240.
16. Fasoli A, Chen CY, Serrano M, Venkataramani S, Saon G, Cui X, et al. Accelerating inference and language model fusion of recurrent neural network transducers via end-to-end 4-bit quantization. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2022; 2022 Sep 18–22; Incheon, Republic of Korea. doi:10.21437/Interspeech.2022-413.
17. Park M, You J, Nagel M, Chang S. Quadapter: adapter for GPT-2 quantization. In: Findings of the Association for Computational Linguistics: EMNLP 2022; 2022 Dec 7–11; Abu Dhabi, United Arab Emirates. doi:10.18653/v1/2022.findings-emnlp.185.
18. Piao T, Cho I, Kang U. SensiMix: sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for BERT compression. PLoS One. 2022;17(4):e0265621. doi:10.1371/journal.pone.0265621.
19. Li L, Liu T, Wang C, Qiu M, Chen C, Gao M, et al. Resizing codebook of vector quantization without retraining. Multimed Syst. 2023;29(3):1499–512. doi:10.1007/s00530-023-01065-2.
20. Yao Z, Yazdani Aminabadi R, Zhang M, Wu X, Li C, He Y. ZeroQuant: efficient and affordable post-training quantization for large-scale transformers. In: Advances in Neural Information Processing Systems; 2022 Nov 28–Dec 9; New Orleans, LA, USA.
21. Dong G, Chen W. Blockwise compression of transformer-based models without retraining. Neural Netw. 2024;171:423–8. doi:10.1016/j.neunet.2023.12.001.
22. Wei X, Zhang Y, Zhang X, Gong R, Zhang S, Zhang Q, et al. Outlier suppression: pushing the limit of low-bit transformer language models. In: Advances in Neural Information Processing Systems; 2022 Nov 28–Dec 9; New Orleans, LA, USA.

23. Dettmers T, Zettlemoyer L. The case for 4-bit precision: K-bit inference scaling laws. In: Proceedings of Machine Learning Research; 2023 Jul 23–29; Honolulu, HI, USA.
24. Gonçalves G, Strubell E. Understanding the effect of model compression on social bias in large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; 2023 Dec 6–10; Singapore. doi:10.18653/v1/2023.emnlp-main.161.
25. Gong Z, Liu J, Wang Q, Yang Y, Wang J, Wu W, et al. PreQuant: a task-agnostic quantization approach for pre-trained language models. In: Findings of the Association for Computational Linguistics: ACL 2023; 2023 Jul 9–14; Toronto, ON, Canada. doi:10.18653/v1/2023.findings-acl.511.
26. Hawks B, Duarte J, Fraser NJ, Pappalardo A, Tran N, Umuroglu Y. Ps and qs: quantization-aware pruning for efficient low latency neural network inference. Front Artif Intell. 2021;4:676564. doi:10.3389/frai.2021.676564.
27. Antony Gnanaprakasam DJR, Sankaran SM, Tharmaraj Charlet JJ. Mitigating catastrophic forgetting in imitation learning for embodied AI using progressive neural networks. Econ Comput Econ Cybern Stud Res. 2025;59(3/2025):96–112. doi:10.24818/18423264/59.3.25.06.
28. Peng P, You M, Xu W, Li J. Fully integer-based quantization for mobile convolutional neural network inference. Neurocomputing. 2021;432:194–205. doi:10.1016/j.neucom.2020.12.035.
29. Bai H, Hou L, Shang L, Jiang X, King I, Lyu MR. Towards efficient post-training quantization of pre-trained language models. In: Advances in Neural Information Processing Systems; 2022 Nov 28–Dec 9; New Orleans, LA, USA.
30. Tang H, Sun Y, Wu D, Liu K, Zhu J, Kang Z. EasyQuant: an efficient data-free quantization algorithm for LLMs. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; 2023 Dec 6–10; Singapore. doi:10.18653/v1/2023.emnlp-main.565.
31. Cho M, Vahid KA, Fu Q, Adya S, Del Mundo CC, Rastegari M, et al. eDKM: an efficient and accurate train-time weight clustering for large language models. IEEE Comput Arch Lett. 2024;23(1):37–40. doi:10.1109/lca.2024.3363492.
32. Kim S, Gholami A, Yao Z, Mahoney MW, Keutzer K. I-BERT: integer-only BERT quantization. In: Proceedings of Machine Learning Research; 2021 Jul 18–24; Virtual.
33. Campos D, Marques A, Kurtz M, Zhai C. oBERTa: improving sparse transfer learning *via* improved initialization, distillation, and pruning regimes. In: Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP); 2023 Jul 13; Toronto, ON, Canada. doi:10.18653/v1/2023.sustainlp-1.3.
34. Zhang C, Cheng J, Shumailov I, Constantinides G, Zhao Y. Revisiting block-based quantisation: what is important for sub-8-bit LLM inference? In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; 2023 Dec 6–10; Singapore. doi:10.18653/v1/2023.emnlp-main.617.
35. Yu C, Chen T, Gan Z, Fan J. Boost vision transformer with GPU-friendly sparsity and quantization. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2023 Jun 17–24; Vancouver, BC, Canada. doi:10.1109/CVPR52729.2023.02170.