

Sustainable LLM Inference for Edge AI: Evaluating Quantized LLMs for Energy Efficiency, Output Accuracy, and Inference Latency

ERIK JOHANNES HUSOM, ARDA GOKNIL, MERVE ASTEKIN, SINTEF, Norway

LWIN KHIN SHAR, Singapore Management University, Singapore

ANDRE KÅSEN, Oslo Metropolitan University, Norway

SAGAR SEN, BENEDIKT ANDREAS MITHASSEL, SINTEF, Norway

AHMET SOYLU, Kristiania University of Applied Sciences, Norway

Deploying Large Language Models (LLMs) on edge devices presents significant challenges due to computational constraints, memory limitations, inference speed, and energy consumption. Model quantization has emerged as a key technique to enable efficient LLM inference by reducing model size and computational overhead. In this study, we conduct a comprehensive analysis of 28 quantized LLMs from the Ollama library, which applies by default Post-Training Quantization (PTQ) and weight-only quantization techniques, deployed on an edge device (Raspberry Pi 4 with 4GB RAM). We evaluate energy efficiency, inference performance, and output accuracy across multiple quantization levels and task types. Models are benchmarked on five standardized datasets (CommonsenseQA, BIG-Bench Hard, TruthfulQA, GSM8K, and HumanEval), and we employ a high-resolution, hardware-based energy measurement tool to capture real-world power consumption. Our findings reveal the trade-offs between energy efficiency, inference speed, and accuracy in different quantization settings, highlighting configurations that optimize LLM deployment for resource-constrained environments. By integrating hardware-level energy profiling with LLM benchmarking, this study provides actionable insights for sustainable AI, bridging a critical gap in existing research on energy-aware LLM deployment.

ACM Reference Format:

Erik Johannes Husom, Arda Goknil, Merve Astekin, Lwin Khin Shar, Andre Kåsen, Sagar Sen, Benedikt Andreas Mithassel, and Ahmet Soylu. 2025. Sustainable LLM Inference for Edge AI: Evaluating Quantized LLMs for Energy Efficiency, Output Accuracy, and Inference Latency. 1, 1 (April 2025), 30 pages.

1 Introduction

Context and Motivation. The rapid advancement of artificial intelligence (AI), particularly in the form of Large Language Models (LLMs), is reshaping our interactions with digital platforms and expanding the capabilities of various applications. LLM architectures, such as GPT (Generative Pre-trained Transformer) [56], LLaMA (Large Language Model Meta AI) [68], and BERT (Bidirectional Encoder Representations from Transformers) [14], exhibit sophisticated abilities to comprehend, generate, and engage in natural language, proving indispensable across diverse use cases—from customer support [75] and sentiment analysis [83] to machine translation [38] and creative content creation [64]. As these models grow in size and complexity, their demand for computational resources and robust cloud infrastructure rises accordingly. However, this dependency on centralized servers introduces challenges related to latency, data privacy, and scalability, particularly as LLM applications become increasingly embedded in personal and mobile devices. To mitigate these issues, researchers and developers are pursuing solutions that bring LLMs closer to the end-user by deploying them on edge devices like IoT gateways (e.g., Raspberry Pi), smartphones, and

Authors' Contact Information: Erik Johannes Husom, Arda Goknil, Merve Astekin, SINTEF, Oslo, Norway, firstname.lastname@sintef.no; Lwin Khin Shar, Singapore Management University, Singapore, Singapore, lkshar@smu.edu.sg; Andre Kåsen, Oslo Metropolitan University, Oslo, Norway, ankas9475@oslomet.no; Sagar Sen, Benedikt Andreas Mithassel, SINTEF, Oslo, Norway, firstname.lastname@sintef.no; Ahmet Soylu, Kristiania University of Applied Sciences, Oslo, Norway, firstname.lastname@kristiania.no.

in-vehicle embedded systems [6, 19, 23, 36, 41, 54, 73, 77, 79, 80, 82, 84]. This approach enables real-time processing and reduces reliance on cloud resources. Yet, deploying LLMs on resource-constrained edge devices poses significant challenges due to their high memory, processing, and energy demands, necessitating innovative techniques to make such on-device models feasible and effective [19, 45, 54, 55]. Model quantization, a technique reducing model size and computational requirements, has emerged as a promising solution to enhance the performance and efficiency of LLMs on these constrained devices [13, 18, 26, 43, 52, 57, 76, 78].

Motivated by the dual pressures of energy efficiency and model accuracy, quantization has gained traction as a viable strategy to adapt LLMs for edge environments. Model quantization techniques, including fixed-point, integer-only, and hybrid quantization methods [37], reduce the model's memory footprint and computational load, making it feasible to deploy on resource-constrained edge devices. However, quantization often affects model accuracy, raising a trade-off between computational efficiency and predictive performance that developers must carefully balance. Additionally, quantization can introduce numerical instability and precision loss, particularly in tasks requiring complex reasoning or arithmetic computations [21]. The extent of accuracy degradation varies across task types, model architectures, and quantization levels, making it essential to evaluate task-specific performance trade-offs. Moreover, while quantization reduces computational load, its impact on energy consumption remains underexplored, especially in real-world edge deployments where power efficiency is critical.

Problem Statement. Despite the advancements, critical challenges persist in understanding and optimizing the interplay between quantized model deployment, energy consumption, and inference accuracy on edge devices. While previous works have investigated quantization methods [7, 25, 43, 60, 61], accuracy trade-offs [28, 33, 39], and other model compression techniques [43, 72, 80], there is limited research on the energy consumption of quantized LLMs in real-world edge deployment. Additionally, prior research often focuses on individual models or quantization techniques without systematically comparing multiple quantization techniques or levels across diverse LLM model families and task types [84]. This gap in research makes it difficult for practitioners and researchers to make informed decisions about which quantization technique, level, and model combination best suits their specific application needs, particularly when balancing energy savings with acceptable levels of model performance for edge deployment scenarios.

Another pertinent issue is the lack of standardized benchmarks and testing environments that enable fair comparisons across LLMs and quantization techniques on edge devices for energy consumption, output accuracy, and inference performance. Several studies undertook evaluations of the performance of quantized LLMs without considering the trade-off for energy consumption, inference performance (latency), and accuracy on edge deployment scenarios [27, 28, 32, 33, 39, 44, 46]. Without a clear, comparative understanding, developers and researchers are often left in a trial-and-error cycle, deploying models that may not be optimal for their edge computing environments. Additionally, the role of dataset and task diversity in affecting energy and accuracy outcomes remains underexplored. Different datasets may stress models in unique ways, potentially impacting the energy-accuracy trade-off and the generalizability of results across use cases.

Our Objectives and Experiment Design. This paper addresses these challenges by presenting a systematic, comparative analysis of quantized LLMs deployed on edge. The primary objective is to measure and evaluate the energy efficiency of different LLMs under different quantization settings and to assess how these configurations impact inference accuracy and latency across multiple datasets and tasks. This analysis will identify and quantify the trade-offs between energy consumption, inference performance, and output accuracy, providing insights into how specific quantization techniques perform on a range of models. By offering this comparative perspective,

the paper aims to equip developers with actionable knowledge on selecting quantization techniques and models that align with both energy and accuracy goals for their edge applications.

In our experiments, we utilize 28 quantized versions LLMs from the **Ollama library** [51], including variants based on Gemma 2 (2B), Lama 3.2 (1B), and Qwen 2.5 (0.5B, 1.5B). The models were selected due to their varying parameter scales and architectures, offering a diverse testbed for analyzing the impact of quantization on energy efficiency and accuracy across edge devices. The Ollama library was chosen for its extensive repository of pre-quantized models, ensuring consistency in quantization methods while enabling direct performance comparisons across models. This diversity allows us to capture insights into the trade-offs between model size, energy consumption, latency, and predictive accuracy, providing a comprehensive understanding of LLM behavior in resource-constrained environments.

For our experiments, we utilize datasets from the **NeurIPS 2024 Challenge: Edge-Device Large Language Model Competition** [45], which are specifically designed to test the diverse capabilities of LLMs deployed on resource-constrained edge devices. These datasets span critical dimensions such as commonsense knowledge (CommonsenseQA [9], TruthfulQA [69]), reasoning (BIG-Bench Hard [5]), mathematical problem-solving (GSM8K [22]), and programming (HumanEval [29]). Due to the constraints of time and computational resources, we employ a systematic sampling strategy to extract representative subsets from each dataset. This approach ensures a balance between preserving the datasets' diversity and maintaining a practical evaluation scope, enabling a robust yet efficient assessment of energy efficiency and output accuracy for quantized LLMs on edge.

We measure the energy consumption of quantized LLM inference using **Joulescope (JS110)** [35], a high-precision hardware-based power monitoring tool. The setup consists of a **Raspberry Pi 4 (4GB RAM)** running the inference workload, with Joulescope capturing real-time voltage and current measurements to compute total energy consumption. Data is recorded on a separate computer, ensuring accurate power profiling without introducing additional computational overhead on the edge device. Joulescope was chosen for its high sampling rate (2 MHz for JS110), fine-grained power measurement capabilities, and ability to provide precise, hardware-based energy profiling, making it ideal for evaluating LLM energy efficiency in real-world edge deployments.

Our Findings. In our experiments, we investigate, based on five datasets, the following Research Questions (RQ)s:

- **RQ1.** *How does model quantization affect the energy consumption of LLM inference on an edge device?* This question focuses on quantifying energy savings from different quantization levels while running LLMs on a constrained hardware platform.
- **RQ2.** *What are the trade-offs between accuracy and energy efficiency across different quantization levels in LLMs deployed on an edge device?* This question explores how varying degrees of quantization affect inference performance, highlighting the balance between computational savings and potential degradation in output quality.
- **RQ3.** *How do different quantization techniques impact inference speed on an edge device?* This research question explores the relationship between quantization levels and inference latency, examining how reduced precision affects token generation rates, model execution time, and overall computational efficiency in a resource-constrained environment.

Our findings reveal key tradeoffs between energy efficiency, accuracy, and inference speed in quantized LLMs deployed on edge devices. For RQ1, quantization consistently reduces energy consumption, with q3 and q4 variants cutting energy use by up to 79% compared to FP16, though extreme quantization can sometimes introduce inefficiencies. For RQ2, accuracy varies significantly across quantization levels and tasks, with some q3 models maintaining competitive accuracy, while others experience substantial degradation, particularly in mathematical reasoning tasks. For RQ3,

quantization reduces inference latency by up to 69%, but the benefits diminish at lower precision due to computational overhead. Smaller models achieve higher throughput, while larger models struggle with latency, particularly in code generation tasks, where longer responses drive higher processing times.

Our Contributions. This paper makes the following key contributions:

- **Comprehensive Evaluation of Quantized LLMs on Edge:** We conduct an extensive evaluation of 28 quantized LLMs from the Ollama framework, which by default applies PQT and weight-only quantization techniques, deployed on an edge device. These models span different model families, parameter sizes, and quantization levels.
- **Analysis of the Energy-Accuracy Trade-Off:** We systematically measure and compare the energy consumption and inference accuracy of quantized LLMs using multiple real-world datasets. This analysis highlights the trade-offs between computational efficiency and model performance, providing actionable insights for optimizing LLM deployment in resource-constrained environments.
- **Insights into Dataset and Task Influence on Quantized Models:** We investigate the impact of dataset and task characteristics on the energy and accuracy outcomes of quantized LLMs. Our findings offer guidance on selecting datasets that appropriately stress models for specific application scenarios, improving the generalizability of results.

The remainder of this paper is as follows. In Section 2, we summarize the background on LLMs and energy monitoring. Section 3 presents our experimental setup and analysis. In Section 4, we summarize our findings for each research question. Section 5 presents threats to the validity of our experiments. In Section 6, we review the related work. Finally, Section 7 concludes the paper.

2 Background

2.1 Model Quantization Techniques for LLMs

Model quantization reduces numerical precision to enhance storage efficiency and computational performance while preserving accuracy. It is broadly classified into post-training quantization (PTQ) and quantization-aware training (QAT). Most LLM quantization techniques use PTQ [84], as it enables efficient inference without requiring retraining.

2.1.1 Post-Training Quantization (PTQ). Post-training quantization (PTQ) applies quantization to a fully trained model without modifying the original training process. This technique converts high-precision weights (typically 32-bit floating-point, FP32) to lower-bit formats such as **INT8 (8-bit integers), INT4, or BF16 (brain floating-point 16-bit)**.

PTQ offers different techniques to reduce memory consumption by replacing high-precision values with low-bit representations for weights, activations, and KV caches:

- (1) **Weight-Only Quantization** only quantizes the weight tensor W of each linear layer. It accelerates memory-bounded General Matrix-Vector Multiply (GEMV) operations during the decoding stage of the LLM inference process [17, 43, 52]. Common weight quantization methods include **uniform quantization**, mapping floating-point values into a fixed range of integer values, and **non-uniform Quantization**, using techniques such as logarithmic or k-means clustering to optimize quantization scales.
- (2) **Weight-Activation Quantization** quantizes both the input activation X and weight tensor W of each linear layer. It leverages low-precision Tensor Cores in GPUs to optimize compute-bounded General Matrix Multiply (GEMM) operations in the prefill stage of the LLM inference process [74, 76, 78].

- (3) **KV Cache Quantization** quantizes the key tensor K and value tensor V in each self-attention block. It minimizes memory overhead, improving efficiency for long texts and large batch sizes [62].

PTQ techniques use different bit-widths to balance accuracy, memory efficiency, and speed. **FP16 (Half Precision)** reduces 32-bit weights to 16-bit, cutting memory usage. **INT8 Quantization** maps weights to 8-bit integers, improving speed with minimal accuracy loss. **INT4 Quantization** quantization further reduces memory and power consumption, maximizing efficiency.

2.1.2 Quantization-Aware Training (QAT). Quantization-aware training (QAT) integrates quantization into the training process, simulating reduced numerical precision during forward and backward passes. Unlike PTQ, QAT enables the model to adjust its weights, minimizing precision loss and improving quantized inference performance. The process involves fake quantization, parameter adaptation via backpropagation, and fine-tuning to retain accuracy while reducing computational demands. QAT is especially useful for deep LLMs, where PTQ alone may cause accuracy degradation, but it requires training data and additional compute resources, making it less practical for large-scale pre-trained models.

2.1.3 Implications for Edge Deployment and Quantization Techniques in Our Experiments. Deploying quantized LLMs on edge devices requires careful consideration of the following factors:

- **Energy Efficiency vs. Accuracy Trade-offs:** Lower-bit quantization improves energy efficiency but can degrade model accuracy, especially in high-precision tasks.
- **Inference Latency:** While quantization reduces memory footprint, extreme quantization levels may introduce non-trivial computational overhead due to dequantization steps.
- **Model Adaptability:** QAT and adaptive quantization methods can help maintain accuracy, but they require additional training, which is often infeasible for large LLMs.

In this study, we systematically evaluate the quantization techniques implemented in the Ollama framework [50], which serves as a high-level abstraction of llama.cpp [47], a highly optimized C++-based inference framework supporting various quantization formats. By default, llama.cpp applies PTQ and weight-only quantization techniques [20]. Specifically, it supports multiple precision levels, including q4_0, q4_1, q8_0, q3_K_S, q3_K_M, q3_K_L, q4_K_S, and q4_K_M [50]. The quantization format follows the pattern **qX_K_Y**, where X denotes bit precision, K indicates K-means clustering (if present), and Y specifies the variant affecting accuracy, memory usage, and speed.

- The parameter **qX** denotes the **bit precision for weight quantization**, where **q4** represents **4-bit** and **q8** denotes **8-bit quantization**.
- K indicates the use of **K-means clustering**, a **non-uniform quantization technique**. In contrast, the absence of K signifies **uniform quantization**, where all values share a fixed scaling factor.
- The parameter Y defines a specific quantization variant that balances **accuracy, memory usage, and inference speed**:
 - Y = 0 applies a **single uniform scaling parameter per block of weights**, resulting in a simple but less adaptive compression method.
 - Y = 1 utilizes **two scaling parameters per block of weights**, which ensures that small weight blocks have a minimum scaling factor, reducing the impact of outliers and improving accuracy.
 - Y = S optimizes for **minimal memory usage** by reducing the number of clusters, making it ideal for **low-resource edge devices**.

- $Y = M$ represents a **balanced quantization strategy**, employing a moderate number of clusters to achieve a compromise between **memory efficiency and accuracy**, suitable for **mid-range GPUs and CPUs**.
- $Y = L$ applies **high-cluster quantization**, minimizing accuracy loss at the cost of increased memory usage, making it best suited for **high-end hardware with ample memory**.

2.2 Energy Monitoring Tools and Technologies

Several energy monitoring tools and technologies have been developed to track and analyze power usage at different levels, including *hardware-level measurements*, *software-based estimation*, and *profiling frameworks tailored for AI workloads*.

2.2.1 Hardware-Based Power Measurement Tools. Hardware-based energy monitoring tools offer the **most precise power consumption data** by directly measuring electrical power usage at the **system level**. While essential for accurate energy profiling, they require specialized equipment. Common tools include **Joulescope** [35], a high-precision external power meter for embedded systems and edge devices; **Monsoon Power Monitor** [49], widely used for mobile and embedded device power analysis. Though highly accurate, these tools cannot isolate CPU vs. GPU consumption, necessitating software-based analysis for component-level breakdowns.

2.2.2 Software-Based Energy Estimation Tools. Software-based energy monitoring tools estimate power consumption using **pre-calibrated models and system telemetry**, offering non-intrusive, lightweight solutions for large-scale experiments. **Scaphandre** [59] profiles CPU and memory power usage in Linux-based environments, making it ideal for cloud and edge AI deployments. Microsoft’s **Joulemeter** [34] estimates power consumption on Windows using machine learning models and system performance counters. For GPU workloads, **NVIDIA-SMI** [11] provides real-time power monitoring widely used for profiling LLM energy consumption on GPU-based devices.

2.2.3 Profiling Frameworks for AI Energy Analysis. **AI-specific profiling frameworks** combine energy tracking with performance evaluation, enabling analysis of trade-offs between inference speed, accuracy, and efficiency. **PyJoules** [53] is a Python-based tool for deep learning workloads, integrating with PyTorch and TensorFlow for per-operation power profiling. **MELODI** [30] monitors LLM inference energy consumption, integrating Scaphandre and NVIDIA-SMI to provide detailed insights into power usage during inference.

2.2.4 Energy Monitoring Solution used in Our Experiments. Due to the limitations of software-based estimations and the need for high-precision, real-time monitoring, we selected **Joulescope** [35], an external hardware-based power meter, as our primary energy measurement tool. Unlike software-based tools and AI profiling frameworks, which rely on telemetry data and estimation models, Joulescope provides direct power measurements. Software-based methods are suitable for standardized environments like x86 desktops, data centers, or cloud AI workloads, but they lack reliable telemetry and hardware support for edge devices, making them unsuitable for our study. Both Joulescope and Monsoon Power Monitor offer high-precision, microsecond-level energy measurements. We chose Joulescope for its broader compatibility, ease of use, and higher dynamic range for measuring power consumption in edge devices.

3 Study Design

3.1 Research Questions

We investigate, based on five datasets, the following Research Questions (RQ)s:

- *RQ1. How does model quantization affect the energy consumption of LLM inference on an edge device?*
- *RQ2. What are the trade-offs between accuracy and energy efficiency across different quantization levels in LLMs deployed on an edge device?*
- *RQ3. How do different quantization techniques impact inference speed on an edge device?*

3.2 Energy Measurement Setup

The experiment is designed to precisely measure LLM inference power consumption on a resource-constrained edge device, the Raspberry Pi 4 (4GB RAM). Equipped with a quad-core Cortex-A72 (ARM v8) CPU at 1.5 GHz and running a 64-bit Linux OS, it represents a typical edge computing platform, making it ideal for assessing the feasibility of deploying LLMs under real-world constraints.

To obtain high-resolution power measurements, the amperemeter-ports of a Joulescope precision power analyzer are connected in series between the Raspberry Pi 4 and its power supply, enabling direct measurement the total current (mA) flowing into it. We also measure the voltage (V) from the power supply. This allows us to calculate power (W), and cumulative energy (J) drawn by the Raspberry Pi. The Joulescope operates at a microsecond-level sampling resolution, capturing fine-grained fluctuations in power usage throughout the inference process. Its sampling frequency of the power consumption is 2 MHz, and we recorded the accumulated energy consumption at 2 Hz. The Joulescope software, running on a laptop, records and visualizes the power consumption in real time, providing a detailed view of energy trends across different inference workloads.

We implemented automated scripts to log LLM-generated responses and corresponding energy consumption values. Responses are stored in structured files for accuracy evaluation, while Joulescope readings are recorded for analyzing quantization effects. Custom scripts process these logs, computing aggregate energy metrics (e.g., average power, total energy per query, peak power) and visualizing energy-performance trade-offs introduced by model quantization.

3.3 Datasets

For our experiments, we used datasets from the **NeurIPS 2024 Challenge: Edge-Device Large Language Model Competition** [45], designed to evaluate LLMs on resource-constrained edge devices. Table 1 provides the summary of the datasets. These datasets cover commonsense knowledge (CommonsenseQA [9], TruthfulQA [69]), reasoning (BIG-Bench Hard [5]), mathematical problem-solving (GSM8K [22]), and programming (HumanEval [29]). We sampled **four out of the five datasets—CommonsenseQA, BIG-Bench Hard, TruthfulQA, and GSM8K**—to create a computationally feasible evaluation set for LLM inference with **28 models on an edge device**. Each dataset is reduced to **200 tasks** using appropriate sampling strategies while maintaining representativeness. **HumanEval** is not sampled, as it contains only **164 tasks**.



Fig. 1. Setup for measuring the energy consumption of LLM inference on an edge device. A Joulescope power meter is between a Raspberry Pi 4 and its power supply, with real-time energy data visualized on a laptop.

Table 1. Summary of the datasets used in our experiments.

Dataset	Task Type	Full Size	Sampled Size	Sampling Method
CommonsenseQA	Multiple-choice QA	12,102	200	Uniform random sampling
BIG-Bench Hard	Reasoning	6,511	200	Hierarchical stratified sampling
TruthfulQA	Multiple-choice QA	817	200	Uniform sampling
GSM8K	Mathematical problem-solving	8,500	200	Uniform random sampling
HumanEval	Code generation	164	164	No sampling (entire dataset used)

3.4 Prompts and Augmented Information

We structured the prompts by combining original dataset questions with explicit instructions to clarify the expected output format and required output data. Following Astekin et al. [2], we instructed the LLMs to enclose answers within `<ANS>` and `</ANS>` tags for easy extraction. To reduce post-processing and ensure efficient inference, we explicitly directed the models to omit additional explanations in their responses.

For all datasets except HumanEval, we applied a standard instruction to the original prompts. Since HumanEval focuses on code generation, we adapted the instruction to align with programming tasks, creating a version tailored for this dataset. The instructions in the experiments are as follows:

- **instruction_general** = "*Print only the answer surrounded by <ANS> and </ANS>. Never print any extra explanations about how the response was generated.*"
- **instruction_humaneval** = "# Complete the function implementation based on the provided docstring, and print only the completed function surrounded by <ANS> and </ANS>. Never print any extra explanations about how the code was generated."

3.5 Accuracy Evaluation Metrics

To evaluate LLM accuracy across datasets, we selected task-specific metrics suited to multiple-choice, reasoning, math problem-solving, and code generation tasks, ensuring a comprehensive assessment of model performance. The accuracy metrics for each dataset are as follows:

- **CommonsenseQA:** Accuracy measures the percentage of correctly answered multiple-choice questions, assessing the model's commonsense reasoning ability.
- **BIG-Bench Hard:** Accuracy evaluates performance across various reasoning challenges by calculating the proportion of correct answers.
- **TruthfulQA:** We implemented an *ensemble approach* (*Exact Match, ROUGE-L, and Cosine Similarity*). In TruthfulQA, answer choices vary from single words to full sentences, and LLM-generated responses may partially or fully match a choice. To evaluate the accuracy, we first checked for exact matches and then used ROUGE-L and cosine similarity to identify the best matching choice.
 - *Exact Match (EM)* checks whether the response exactly matches the ground-truth answer.
 - *ROUGE-L* evaluates text similarity using the Longest Common Subsequence (LCS), measuring how well word order is preserved without requiring an exact match [12, 42].
 - *Cosine Similarity* measures the angle between two text vectors in a high-dimensional space. It assesses semantic closeness between the generated and reference answers [71].We explored other metrics but found them unsuitable due to limitations. Jaccard Similarity measures word overlap but ignores word order, making it ineffective for partial responses. BLEU Score, designed for full sentences, can misrepresent partial matches. LCS alone lacks the flexibility of ROUGE-L in handling minor variations.
- **GSM8K:** Accuracy determines correctness by verifying whether the final computed answer matches the target answer in the solution.

- **HumanEval:** *pass@k* measures the probability that at least one of the top-k generated code solutions passes all unit tests [8]. In our experiments, we set k = 1 since we request only a single solution from the LLMs.

To calculate accuracy, we use the standard formula: *the proportion of correct answers out of the total number of questions or tasks.*

3.6 Models

Table 2 presents an **overview of the quantized LLMs used in our experiments**, detailing their model names, parameter sizes, storage requirements, and quantization formats. The models are categorized into three main *model families*: **LLaMA 3.2, Qwen 2.5, and Gemma 2**, with parameter sizes ranging from 494 million to 2.6 billion. We have in total four *base models*: LLaMA 3.2:1b, Qwen 2.5:0.5b, Qwen 2.5:1.5b, and Gemma 2:2b. To evaluate the trade-offs between model size, inference accuracy, and energy consumption on an edge device, we selected models that have been quantized using different precision levels, giving us multiple *model variants* for each base model.

The table consists of four columns: **Model, Parameter Size, Size (MB), and Quantization Type.** The Model column lists the model and quantization format applied. The Parameter Size column indicates the total number of model parameters, ranging from **494M (million) to 2.6B (billion)**. The Size (MB) column represents the storage footprint of each model, which varies based on the quantization level. Finally, the Quantization Type column specifies the applied quantization format, directly influencing model accuracy, memory footprint, and inference efficiency.

The **LLaMA 3.2 family** consists of 1.2B parameter models employing PTQ and weight-only quantization techniques, including FP16 (half precision), 8-bit (q8_0), 4-bit (q4_0, q4_1, q4_K_M, q4_K_S), and 3-bit (q3_K_L, q3_K_M, q3_K_S). The **Qwen 2.5 family** includes models with 1.5B and 494M parameters, supporting 8-bit, 4-bit, and 3-bit quantization, with formats such as FP16, q8_0, q4_1, q4_0, q4_K_M, q4_K_S, q3_K_L, and q3_K_S. These models provide a balance between memory efficiency and accuracy, making them well-suited for various edge-computing environments. The **Gemma 2 family** comprises 2.6B parameter models utilizing 3-bit quantization (q3_K_S, q3_K_M, q3_K_L), which are the largest models used in our experiments. While these models require more memory, they offer enhanced performance for complex inference tasks.

Quantization significantly reduces model size, making LLM deployment feasible on resource-constrained hardware. The **FP16 models** are the largest, with **LLaMA 3.2 FP16 requiring 2364.74MB and Qwen 2.5 FP16 using 948.11MB**. In contrast, **lower-bit models (q3_K_S,**

Table 2. Overview of the quantized LLMs in the experiments.

	<i>Model</i>	<i>Paramet.</i>	<i>Size (MB)</i>	<i>Quanti.</i>
<i>Llama</i>	llama3.2:1b-instruct-fp16	1.2B	2364.74 MB	FP16
	llama3.2:1b-instruct-q8_0	1.2B	1259.90 MB	Q8_0
	llama3.2:1b-instruct-q4_1	1.2B	793.23 MB	Q4_1
	llama3.2:1b-instruct-q4_K_M	1.2B	770.29 MB	Q4_K_M
	llama3.2:1b-instruct-q4_0	1.2B	735.23 MB	Q4_0
	llama3.2:1b-instruct-q4_K_S	1.2B	739.73 MB	Q4_K_S
	llama3.2:1b-instruct-q3_K_L	1.2B	698.60 MB	Q3_K_L
	llama3.2:1b-instruct-q3_K_M	1.2B	658.85 MB	Q3_K_M
	llama3.2:1b-instruct-q3_K_S	1.2B	611.98 MB	Q3_K_S
	qwen2.5:1.5b-instruct-q4_1	1.5B	969.75 MB	Q4_1
<i>Qwen</i>	qwen2.5:1.5b-instruct-q4_K_M	1.5B	940.38 MB	Q4_K_M
	qwen2.5:1.5b-instruct-q4_0	1.5B	891.66 MB	Q4_0
	qwen2.5:1.5b-instruct-q4_K_S	1.5B	896.76 MB	Q4_K_S
	qwen2.5:1.5b-instruct-q3_K_L	1.5B	839.40 MB	Q3_K_L
	qwen2.5:1.5b-instruct-q3_K_M	1.5B	786.01 MB	Q3_K_M
	qwen2.5:1.5b-instruct-q3_K_S	1.5B	725.71 MB	Q3_K_S
	qwen2.5:0.5b-instruct-fp16	494.03M	948.11 MB	FP16
	qwen2.5:0.5b-instruct-q8_0	494.03M	506.48 MB	Q8_0
	qwen2.5:0.5b-instruct-q4_1	494.03M	357.18 MB	Q4_1
	qwen2.5:0.5b-instruct-q4_K_M	494.03M	379.39 MB	Q4_K_M
	qwen2.5:0.5b-instruct-q4_0	494.03M	335.85 MB	Q4_0
	qwen2.5:0.5b-instruct-q4_K_S	494.03M	367.63 MB	Q4_K_S
	qwen2.5:0.5b-instruct-q3_K_L	494.03M	352.26 MB	Q3_K_L
	qwen2.5:0.5b-instruct-q3_K_M	494.03M	339.01 MB	Q3_K_M
	qwen2.5:0.5b-instruct-q3_K_S	494.03M	322.61 MB	Q3_K_S
<i>Gemma</i>	gemma2:2b-instruct-q3_K_S	2.6B	1297.64 MB	Q3_K_S
	gemma2:2b-instruct-q3_K_M	2.6B	1393.96 MB	Q3_K_M
	gemma2:2b-instruct-q3_K_L	2.6B	1478.62 MB	Q3_K_L

q3_K_M, q3_K_L) require substantially less memory, making them ideal for edge applications. However, this reduction in size comes with trade-offs in accuracy, as higher-bit quantization formats (e.g., q8_0, q4_1, q4_0) retain more precision, whereas lower-bit formats (q3_K_S, q3_K_L) prioritize efficiency at the cost of accuracy. Additionally, different quantization formats optimize models for various hardware constraints. Weight-only quantization formats (**qX_K_Y**) help balance memory efficiency and computational performance, with higher K values (q3_K_L, q4_K_L) reducing accuracy loss but demanding more memory.

By evaluating these quantized models on an edge device, we analyzed how reduced precision affects inference energy consumption while maintaining usable accuracy. This selection of models allows us to systematically compare:

- Higher precision vs. lower precision models (e.g., FP16 vs. INT8 vs. INT4 vs. INT3).
- Different quantization methods (e.g., uniform quantization (Q4_0) vs. balanced, non-uniform K-means quantization (Q4_K_M)).
- Model family impact on energy efficiency and inference latency (e.g., LLaMA3.2 vs. Qwen2.5 vs. Gemma2).

3.7 Variables

In our experiment, we analyzed the **energy efficiency, inference performance, and output accuracy** of quantized LLMs deployed on an edge device. These aspects are influenced by various factors, which can be categorized as **independent** and **dependent** variables.

3.7.1 Independent Variables (Factors We Manipulate). These are the variables we control or modify to observe their effects on the dependent variables:

- (1) **LLM family** – The specific language model family tested (e.g., LLaMA3.2, Qwen2.5, Gemma2)
- (2) **Model size in number of parameters** - The number of parameters for a given model, which is independent of the quantization technique.
- (3) **Model size in bytes** - The file size of the model, which varies according to the quantization type and level.
- (4) **Model Quantization Type and Level** – The bit-width of the quantized models (e.g., 8-bit).
- (5) **Task Type** – The nature of the benchmark datasets (e.g., multiple-choice reasoning, mathematical problem-solving, code generation).
- (6) **Dataset Used** – The dataset from which inference tasks are drawn (CommonsenseQA, BIG-Bench Hard, TruthfulQA, GSM8K, HumanEval).

3.7.2 Dependent Variables (Measured Outcomes). These are the variables we observe and measure to evaluate the impact of the independent variables:

- (1) **Energy Consumption** – The total power usage (measured in joules) during inference, captured using the hardware-based energy monitoring tool (i.e., Joulescope).
- (2) **Inference Latency** – The time taken for the model to generate a response.
- (3) **Token Generation Throughput** – The number of tokens generated per second.
- (4) **Output Accuracy** – The correctness of LLM responses, benchmarked against ground truth labels for each dataset.

By analyzing the impact of independent variables, we aimed to gain insights into optimal configurations for deploying LLMs in energy-constrained environments.

3.8 Experiment Execution

The experiment setup includes a Raspberry Pi 4 for inference, a Joulescope power meter for energy measurement, and a computer for data recording (see Section 3.2). A systematic procedure

ensures precise power monitoring across all models and datasets. The five sampled datasets and accompanying scripts are available in our repository [16].

To ensure accurate energy measurements, we followed a structured execution procedure. The system was initialized by powering on the computer and Joulescope, supplying power to the Raspberry Pi 4 (RPi). We verified the Ollama server and ensured all models fit within the RPi's memory to prevent inference failures. The experiment script (`exp.sh`) was configured to specify models and datasets, with session sizes adjusted based on available disk space. To align power measurements with inference execution, we synchronized clocks between the RPi and the recording computer. The `inference.py` script included a 20-minute timeout to prevent indefinite hangs. Joulescope-UI recorded raw signal samples and statistical data, ensuring precise energy profiling.

During inference, models were evaluated sequentially per dataset, with continuous power logging. To ensure consistency, we completed all model evaluations on one dataset before proceeding to the next. After each session, power logs and outputs were backed up, and the RPi was restarted. For datasets with long responses (e.g., HumanEval), a batching strategy was used to manage storage constraints. This systematic approach ensured repeatability, precision, and reliable energy-performance analysis of quantized LLMs.

3.9 Data Analysis

Data Preparation. Our raw data includes energy consumption, token counts, processing time, and accuracy metrics per inference operation. To ensure statistical robustness, we aggregated measurements across multiple runs for each model variant. The data is categorized by model architecture, quantization level, and dataset, with analysis focused on the dependent variables outlined in Section 3.7.2.

The recorded energy consumption includes the entire device's usage, including background processes. To isolate LLM inference energy, we measured the Raspberry Pi's idle power consumption and used the average value to subtract the baseline energy consumption from our measurements. As shown in Table 3, the device's average idle power was 2.85W (energy consumption of 2.85J/s), which was deducted from the energy consumption values recorded during inference.

Data Post-processing. The first step in data post-processing involves extracting the generated answer enclosed within `<ANS>` and `</ANS>` tags. Since LLMs may not strictly follow the instructed format, responses often require cleaning and extraction from unstructured or semi-structured text. Missing opening or closing tags, extra parentheses, or misplaced characters are corrected to isolate the target answer. Invalid cases, such as empty responses or improperly formatted tags, are filtered out. Additionally, some dataset target answers require post-processing to extract the final value. For example, in GSM8K, solutions include detailed explanations, requiring scripts to identify and extract the final numerical or textual answer. We developed scripts to extract the actual target answers from the dataset.

Statistical Analysis. For each model variant and quantization level, we computed mean, median, and variance of the collected metrics to support visualization and analysis of the results. To identify optimal model configurations balancing energy efficiency and task performance, we applied Pareto efficiency analysis. A configuration is Pareto-optimal if no other variant achieves higher accuracy with the same or lower energy consumption or equal accuracy with lower energy usage. Formally, a model variant $P_1(energy_1, accuracy_1)$ dominates another configuration $P_2(energy_2, accuracy_2)$ if:

- (1) $energy_1 \leq energy_2$ (equal or less energy consumption)
- (2) $accuracy_1 \geq accuracy_2$ (equal or higher accuracy)

Table 3. Idle power consumption (in Watts) of the Raspberry Pi 4, measured over a period of 104 minutes.

	mean	std	min	25%	50%	75%	max
power (W)	2.85	0.17	2.81	2.81	2.81	2.82	5.63

- (3) Either ($energy_1 < energy_2$) or ($accuracy_1 > accuracy_2$) (strictly better in at least one dimension)

The Pareto-optimal models represent the best-performing configurations, enabling practitioners to choose the most suitable balance between task performance and energy efficiency.

Scaling Analysis. To assess the impact of model quantization on performance and efficiency, we analyzed the relationship between model size (bytes), energy consumption, and accuracy. Model size was used instead of parameter count as it directly reflects memory footprint differences across quantization levels. For each model-dataset pair, we applied linear regression (scikit-learn’s `LinearRegression`) to quantify scaling trends across quantization levels.

Visualization. We employed multiple visualization techniques to represent our analysis:

- (1) Box plots: To visualize the distribution of energy consumption metrics across different models and quantization levels. Each box plot depicts the median (horizontal line splitting the main box), interquartile range (IQR, main box), and whiskers extending to 1.5 times the IQR, effectively summarizing central tendency and variability.
- (2) Scatter plots: To examine relationships between energy consumption and accuracy, with points representing different model configurations. Data points are colored and styled according to which model and dataset they represent. For certain figures we plot the fitted regression lines indicating trend lines.
- (3) Pareto frontier curves: Connecting Pareto-optimal configurations to visualize the trade-off boundary between energy efficiency and accuracy.

4 Experiment Results

4.1 How does model quantization affect the energy consumption of LLM inference on an edge device? (RQ1)

To address RQ1, we computed the average energy consumption (Joules per token) for all model variants across the five benchmark datasets. Results are shown in Table 6 and Figure 2.

4.1.1 Baseline Energy Consumption Across Base Models. As shown in Figure 2, energy consumption per token varies significantly across base models. Table 4 summarizes these differences, with `qwen2.5_0.5b` being the most efficient at 2.61 J/token, approximately 3–3.5 times more efficient than other model families due to its smaller parameter size. Despite having more parameters, `qwen2.5_1.5b` (7.57 J/token) is slightly more efficient than `llama3.2_1b` (8.40 J/token), and also outperforms `Gemma2_2b` (9.35 J/token). Standard deviations indicate high variability, particularly in `llama3.2_1b` (± 5.36 J/token), which shows the greatest range between its full-precision and most quantized variants.

Examining energy consumption per response (Table 5) reveals patterns similar to per-token measurements.

`Qwen2.5_0.5b` is the most efficient at 58.83 J/response, while `Gemma2_2b` consumes the most at 255.79 J/response.

Quantization significantly reduces energy usage, with savings of 52% in `LLaMA3.2_1b` and 54% in `qwen2.5_0.5b` when comparing

Table 4. Energy per token for model families averaged across the datasets.

Base Model	Mean Energy per Token (J/tok)
<code>qwen25_0.5b</code>	2.61 ± 1.49
<code>qwen25_1.5b</code>	7.57 ± 4.54
<code>llama3.2_1b</code>	8.40 ± 5.36
<code>gemma2_2b</code>	9.35 ± 3.30

Table 5. Average energy consumption per response (Joules) by base model and quantization level. For comparison, the average cost of prompting `Gemma2_2b` equals powering a 5W light bulb for 51 seconds.

	<code>gemma2_2b</code>	<code>llama3.2_1b</code>	<code>qwen25_0.5b</code>	<code>qwen25_1.5b</code>	Average
<code>fp16</code>	N/A	159.42	91.59	N/A	125.51
<code>q8_0</code>	N/A	75.85	42.34	N/A	59.10
<code>q4 variants (avg)</code>	N/A	83.95	49.75	107.00	80.23
<code>q3 variants (avg)</code>	255.79	101.02	51.65	113.72	130.54
Average	255.79	105.06	58.83	110.36	

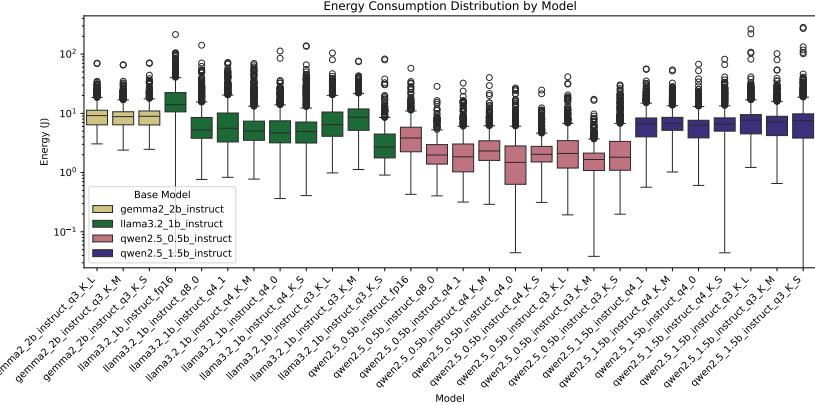


Fig. 2. Distribution of energy consumption per token for all models, all datasets included.

Table 6. Mean and standard deviation energy consumption of models on various datasets.

Model	bigbenchhard	commonsenseqa	gsm8k	humaneval	truthfulqa	Average
gemma2_2b_instruct_q3_K_L	14.40 ± 9.02	10.38 ± 1.65	10.34 ± 2.20	4.69 ± 1.15	7.98 ± 2.59	9.56 ± 3.32
gemma2_2b_instruct_q3_K_M	13.86 ± 8.66	9.56 ± 1.46	10.26 ± 1.92	4.70 ± 1.20	8.00 ± 2.66	9.28 ± 3.18
gemma2_2b_instruct_q3_K_S	14.38 ± 9.69	9.67 ± 1.43	10.48 ± 2.49	3.14 ± 0.71	8.40 ± 2.67	9.22 ± 3.40
llama3.2_1b_instruct_fp16	26.89 ± 25.11	15.01 ± 7.34	25.44 ± 16.74	5.93 ± 2.22	14.74 ± 7.09	17.60 ± 11.70
llama3.2_1b_instruct_q8_0	16.74 ± 15.31	5.81 ± 2.41	9.59 ± 6.22	2.40 ± 0.88	4.83 ± 2.02	7.87 ± 5.37
llama3.2_1b_instruct_q4_1	20.66 ± 14.02	6.69 ± 2.06	5.74 ± 3.42	1.67 ± 0.46	7.36 ± 4.13	8.42 ± 4.82
llama3.2_1b_instruct_q4_K_M	16.70 ± 13.19	5.21 ± 0.94	6.58 ± 3.73	2.25 ± 0.79	4.82 ± 2.21	7.11 ± 4.17
llama3.2_1b_instruct_q4_0	12.84 ± 11.27	5.00 ± 1.60	6.21 ± 2.53	1.95 ± 1.02	4.56 ± 2.50	6.11 ± 3.78
llama3.2_1b_instruct_q4_K_S	18.78 ± 16.65	5.21 ± 1.19	5.90 ± 2.50	1.91 ± 0.59	4.43 ± 1.93	7.25 ± 4.57
llama3.2_1b_instruct_q3_K_L	13.95 ± 12.71	8.84 ± 3.72	9.20 ± 4.53	2.23 ± 1.10	7.02 ± 4.70	8.25 ± 5.35
llama3.2_1b_instruct_q3_K_M	13.82 ± 10.75	10.93 ± 3.88	7.90 ± 3.53	2.33 ± 0.99	11.22 ± 5.64	9.24 ± 4.96
llama3.2_1b_instruct_q3_K_S	6.03 ± 9.46	4.00 ± 2.30	3.02 ± 1.70	2.33 ± 1.23	3.38 ± 2.76	3.75 ± 3.49
qwen2.5_0.5b_instruct_fp16	7.25 ± 6.87	6.74 ± 2.27	3.68 ± 1.16	1.01 ± 0.10	3.43 ± 1.47	4.42 ± 2.37
qwen2.5_0.5b_instruct_q8_0	3.40 ± 3.08	3.14 ± 1.40	1.79 ± 0.59	0.58 ± 0.06	1.79 ± 0.80	2.14 ± 1.19
qwen2.5_0.5b_instruct_q4_1	5.00 ± 4.56	3.19 ± 0.88	1.60 ± 0.87	0.58 ± 0.10	1.82 ± 1.01	2.44 ± 1.48
qwen2.5_0.5b_instruct_q4_K_M	4.43 ± 4.59	3.48 ± 1.49	2.17 ± 1.00	0.62 ± 0.09	2.08 ± 0.94	2.56 ± 1.62
qwen2.5_0.5b_instruct_q4_0	5.51 ± 4.78	3.06 ± 1.01	1.01 ± 0.61	0.52 ± 0.08	1.42 ± 0.55	2.30 ± 1.41
qwen2.5_0.5b_instruct_q4_K_S	3.80 ± 3.19	2.34 ± 1.01	2.09 ± 0.67	0.58 ± 0.08	2.10 ± 1.03	2.18 ± 1.20
qwen2.5_0.5b_instruct_q3_K_L	5.52 ± 5.36	3.68 ± 0.75	2.07 ± 0.69	0.54 ± 0.06	1.77 ± 1.34	2.72 ± 1.64
qwen2.5_0.5b_instruct_q3_K_M	3.03 ± 2.32	1.87 ± 0.47	1.90 ± 0.51	0.51 ± 0.05	1.61 ± 0.78	1.78 ± 0.83
qwen2.5_0.5b_instruct_q3_K_S	7.04 ± 6.01	3.51 ± 0.85	1.54 ± 0.66	0.91 ± 0.30	1.51 ± 0.57	2.90 ± 1.68
qwen2.5_1.5b_instruct_q4_1	12.07 ± 8.02	7.86 ± 1.34	8.13 ± 4.02	1.84 ± 0.42	4.99 ± 1.50	6.98 ± 3.06
qwen2.5_1.5b_instruct_q4_K_M	12.44 ± 7.76	7.44 ± 1.77	7.24 ± 1.97	2.28 ± 0.94	5.50 ± 2.02	6.98 ± 2.89
qwen2.5_1.5b_instruct_q4_0	11.51 ± 8.18	7.35 ± 1.20	6.77 ± 1.55	2.11 ± 0.95	4.88 ± 1.83	6.52 ± 2.74
qwen2.5_1.5b_instruct_q4_K_S	15.23 ± 10.19	7.15 ± 1.57	6.79 ± 1.86	1.96 ± 0.61	5.22 ± 2.11	7.27 ± 3.27
qwen2.5_1.5b_instruct_q3_K_L	16.58 ± 26.36	8.81 ± 1.37	9.87 ± 4.96	2.37 ± 0.65	6.04 ± 2.44	8.73 ± 7.16
qwen2.5_1.5b_instruct_q3_K_M	13.28 ± 11.44	8.51 ± 1.51	7.56 ± 3.94	2.09 ± 0.54	6.23 ± 2.77	7.53 ± 4.04
qwen2.5_1.5b_instruct_q3_K_S	18.62 ± 31.40	8.58 ± 1.82	11.18 ± 6.84	0.25 ± 0.60	6.18 ± 2.44	8.96 ± 8.62
Average	11.92 ± 10.71	6.54 ± 1.81	6.64 ± 2.98	1.94 ± 0.64	5.12 ± 2.30	6.43

FP16 to q8_0. Notably, q4 quantization is generally more efficient than q3, suggesting that aggressive quantization may introduce overhead that offsets energy gains.

4.1.2 Impact of Quantization on Energy Consumption. Quantization significantly reduces energy consumption compared to higher-precision models. Llama3.2_1b_instruct_fp16 consumes 17.60 J/token, while its most efficient quantized variant (q3_K_S) requires only 3.75 J/token, achieving a 79% reduction. Similarly, for qwen2.5_0.5b_instruct, energy consumption drops from 4.42 J/token (FP16) to 1.78 J/token (q3_K_M), a 60% reduction. The impact of quantization levels below FP16 on energy consumption is non-linear, and more aggressive quantization does not always yield greater

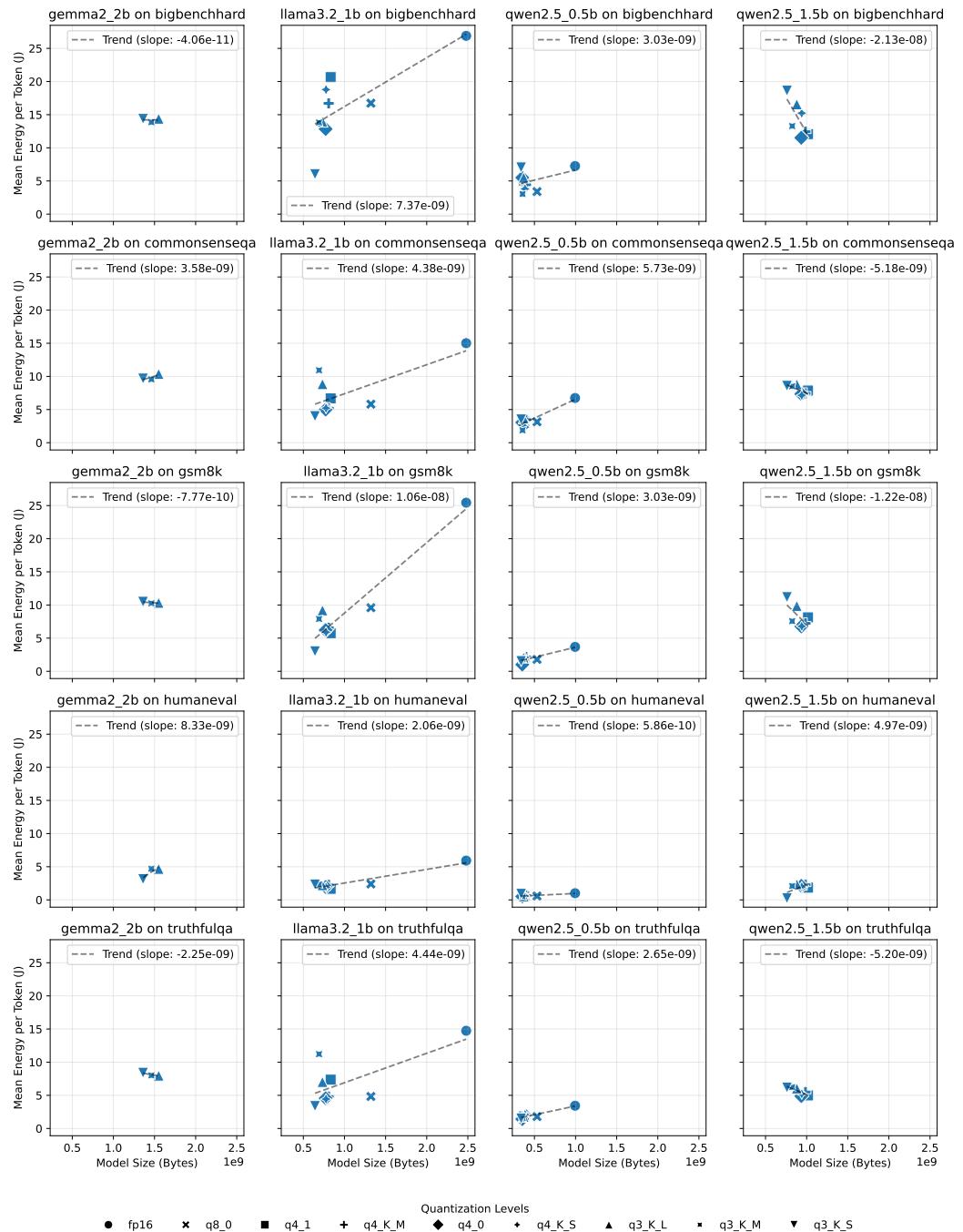


Fig. 3. Relationship between model size (in bytes) and energy consumption per token across different model families and benchmark datasets. Each subplot represents a specific model-dataset combination with fitted linear regression trend lines showing the scaling relationship between model size and energy consumption. Marker shapes indicate different quantization techniques.

efficiency. For example, in qwen2.5_0.5b, q3_K_M (1.78 J/token) is 62% more efficient than q3_K_S (2.90 J/token) despite having the same bit precision. Similarly, in qwen2.5_1.5b, all q3 variants consume more energy per token than q4 variants, emphasizing that quantization method selection is as crucial as reducing bit width.

Table 6 shows high variability in energy consumption, particularly in highly quantized models. Llama3.2_1b_instruct_q3_K_S exhibits extremely high variability (± 3.49 J/token on average), sometimes exceeding the mean. This variance suggests that energy consumption in highly quantized models may be less predictable and more influenced by input characteristics. In contrast, qwen2.5_0.5b_instruct_q3_K_M is more stable (± 0.83 J/token).

Figure 3 examines how energy consumption per token scales with model size (bytes) across five benchmark datasets. Model size correlates with quantization level, where higher precision (e.g., FP16, Q8) increases size, while aggressive quantization (e.g., Q3) reduces it. This enables a relative comparison of quantization methods across different precision levels. With four base models and five datasets, we analyzed twenty configurations, though some quantization levels were excluded due to memory limitations on the Raspberry Pi. For each configuration, we fitted linear regression trend lines, revealing distinct scaling patterns across models and tasks. A positive trend indicates that energy consumption rises with model size, meaning aggressive quantization reduces energy use. Conversely, a negative trend suggests that larger models may consume less energy per token than their more aggressively quantized counterparts.

For llama3.2_1b and qwen2.5_0.5b, positive scaling coefficients across all benchmarks (0.625 to 1.15) indicate that energy consumption increases with model size, driven by the efficiency gains of quantization beyond FP16. However, other models show unexpected patterns. Qwen2.5_1.5b exhibits negative scaling coefficients (-0.572 to -1.27) on several benchmarks, suggesting that some larger quantized variants may be more energy-efficient than smaller ones. Gemma2_2b displays mixed behavior, with coefficients ranging from -0.403 to 3.16 across tasks. These diverse trends highlight the complex interplay between model architecture, quantization strategy, and task characteristics.

4.1.3 Task-Specific Energy Consumption Patterns.

Task characteristics greatly impact energy consumption. Table 6 shows HumanEval has the lowest energy use (1.94 J/token), while BigBenchHard has the highest (11.92 J/token)—a 6× difference. Table 7 reveals that HumanEval responses are significantly longer (162.49 tokens on average) compared to 8.00–21.63 tokens for other tasks. Since HumanEval requires code generation, longer outputs distribute the fixed inference overhead across more tokens, reducing energy per token. The quantization impact varies by task. In GSM8K (mathematical reasoning), llama3.2_1b shows high variability, with energy consumption ranging from 3.02 J/token (q3_K_S) to 25.44 J/token (FP16)—an 8.4× difference. In contrast, for HumanEval (code generation), the range is narrower (2.33 to 5.93 J/token, 2.5× difference). This suggests quantization provides greater efficiency gains for mathematical reasoning than for code generation.

4.1.4 Correlation Between Response Length and Energy Consumption.

Due to varying response lengths across datasets (Table 7), we analyzed their correlation with energy consumption. Figure 4 shows the Pearson correlation coefficients for each model variant and benchmark, highlighting this relationship. The average correlation across all models and benchmarks is 0.54, indicating that response length significantly impacts energy consumption but is not the sole factor. This analysis reveals several key patterns:

The correlation strength varies across datasets, with HumanEval showing the highest (0.91 on average). This is likely due to its longer, variable-length responses (162.49 tokens on average) in code generation tasks. The strong correlation suggests response length can be a reliable proxy for estimating energy consumption in such tasks.

BigBenchHard and CommonsenseQA show weaker correlations (0.26 and 0.31), suggesting for shorter responses, factors beyond token count play a larger role in energy consumption.

Energy consumption varies across model families, with the variant llama3.2_1b_instruct_q3_K_S showing a strong correlation (0.96) between response length and energy usage across benchmarks. Its higher average response length (74.54 vs. overall 44.49, Table 7). For models with longer responses, energy consumption becomes more predictable based on output length because fixed computational overhead becomes proportionally smaller. In contrast, models generating shorter responses show more variable energy-output relationships, as initialization and prompt processing costs dominate.

These findings are crucial for energy estimation in real-world deployments. In tasks with longer responses (e.g., code generation), response length reliably predicts energy consumption. However, for shorter responses, energy estimates must consider additional factors beyond length.

RQ1 Conclusion. Our analysis reveals that quantization significantly reduces energy consumption, with more aggressive quantization (e.g., q3_K_M vs. FP16) yielding up to 79% savings in some models. However, the relationship between quantization level and energy efficiency is non-linear, as extreme quantization sometimes increases energy usage, possibly due to overhead or inefficiencies. Additionally, task characteristics influence energy consumption, with longer response tasks (e.g., HumanEval) showing lower energy per token due to fixed inference overhead distribution, while short-response tasks exhibit greater variability. These highlight the importance of selecting quantization strategies based on task type and deployment constraints.

4.2 What are the trade-offs between accuracy and energy efficiency across different quantization levels in LLMs deployed on an edge device? (RQ2)

To address RQ2, we analyzed the accuracy-energy tradeoff across quantization levels and model families. Table 8 reports accuracy scores for all models on the five benchmark datasets, while Figure 5 visualizes accuracy comparisons across variants.

4.2.1 Impact of Quantization on Accuracy. Figure 5 shows that quantization impacts accuracy differently across model families. In gemma2_2b base model, accuracy ranges from 0.30 ± 0.35 to 0.45 ± 0.42 , with q3_K_M (0.45 ± 0.42) outperforming q3_K_L despite more aggressive quantization. This suggests that some quantization techniques can preserve or even enhance model performance for specific architectures. The llama3.2_1b base model shows the widest accuracy variation (0.17 ± 0.30 to 0.39 ± 0.40), with greater quantization leading to clear degradation. While q4_K variants retain competitive accuracy (0.39 ± 0.40), q3_K variants drop to $0.17\text{--}0.20$, losing about 40% accuracy compared to higher-precision models.

The qwen2.5_0.5b base model remains stable across quantization levels, with accuracy ranging from 0.28 ± 0.40 to 0.35 ± 0.42 and no clear link between quantization aggressiveness and accuracy.

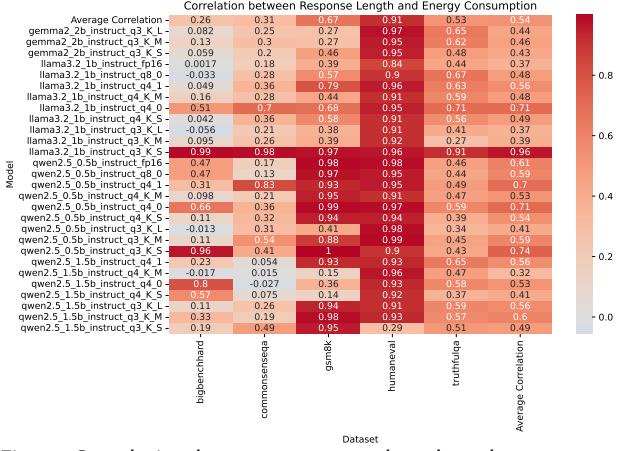


Fig. 4. Correlation between response length and energy consumption per response.

Table 7. Average response length for models on various datasets, with standard deviation.

Model	bigbenchhard	commonsenseqa	gsm8k	humaneval	truthfulqa	Average
gemma2_2b_instruct_q3_K_L	14.07 ± 8.25	11.12 ± 1.53	12.28 ± 3.32	190.56 ± 106.81	19.54 ± 10.20	49.51 ± 26.02
gemma2_2b_instruct_q3_K_M	14.37 ± 11.39	11.53 ± 1.60	11.89 ± 1.70	168.29 ± 89.30	19.18 ± 9.68	45.05 ± 22.73
gemma2_2b_instruct_q3_K_S	14.90 ± 10.00	12.19 ± 1.34	13.40 ± 5.87	196.04 ± 87.90	18.64 ± 5.59	51.03 ± 22.14
llama3.2_1b_instruct_fp16	9.27 ± 8.92	7.41 ± 2.13	6.49 ± 5.32	52.72 ± 38.65	12.19 ± 6.64	17.61 ± 12.33
llama3.2_1b_instruct_q8_0	9.16 ± 8.90	7.68 ± 1.94	6.68 ± 4.94	64.20 ± 40.02	12.59 ± 7.58	20.06 ± 12.67
llama3.2_1b_instruct_q4_1	10.98 ± 13.78	6.62 ± 2.05	15.59 ± 18.46	124.67 ± 83.34	9.86 ± 8.24	33.54 ± 25.17
llama3.2_1b_instruct_q4_K_M	10.32 ± 17.00	7.95 ± 1.08	9.14 ± 5.15	68.99 ± 46.95	12.93 ± 7.88	21.87 ± 15.61
llama3.2_1b_instruct_q4_0	13.26 ± 32.87	9.10 ± 7.88	9.62 ± 8.00	103.50 ± 85.60	14.43 ± 14.30	29.98 ± 29.73
llama3.2_1b_instruct_q4_K_S	10.57 ± 12.84	7.99 ± 2.70	9.33 ± 5.96	81.93 ± 50.05	13.31 ± 7.38	24.63 ± 15.79
llama3.2_1b_instruct_q3_K_L	9.13 ± 10.72	5.88 ± 2.15	6.95 ± 4.03	84.43 ± 66.48	10.97 ± 6.24	23.47 ± 17.93
llama3.2_1b_instruct_q3_K_M	7.18 ± 10.24	4.36 ± 2.12	7.30 ± 3.18	110.13 ± 82.76	5.48 ± 3.28	26.89 ± 20.32
llama3.2_1b_instruct_q3_K_S	80.35 ± 316.80	27.71 ± 33.27	47.45 ± 57.44	179.57 ± 193.26	37.62 ± 35.56	74.54 ± 127.27
qwen25_0.5b_instruct_fp16	13.29 ± 21.25	4.64 ± 0.96	24.72 ± 60.38	271.23 ± 109.90	14.47 ± 5.63	65.67 ± 39.62
qwen25_0.5b_instruct_q8_0	11.96 ± 18.42	4.65 ± 1.02	20.57 ± 53.84	209.98 ± 69.21	14.54 ± 5.42	52.34 ± 29.58
qwen25_0.5b_instruct_q4_1	11.15 ± 21.93	6.07 ± 8.71	66.37 ± 119.06	233.70 ± 107.01	14.07 ± 6.70	66.27 ± 52.68
qwen25_0.5b_instruct_q4_K_M	12.33 ± 12.43	5.58 ± 2.80	31.31 ± 64.23	205.92 ± 78.68	14.90 ± 6.97	54.01 ± 33.02
qwen25_0.5b_instruct_q4_0	18.09 ± 45.29	5.25 ± 2.25	121.16 ± 142.43	236.14 ± 114.56	16.91 ± 7.96	79.51 ± 62.50
qwen25_0.5b_instruct_q4_K_S	12.42 ± 10.26	8.56 ± 2.93	16.88 ± 43.78	201.92 ± 81.86	14.57 ± 6.23	50.87 ± 29.01
qwen25_0.5b_instruct_q3_K_L	10.97 ± 10.36	4.45 ± 0.85	10.92 ± 4.43	258.34 ± 104.98	15.45 ± 6.28	60.03 ± 25.38
qwen25_0.5b_instruct_q3_K_M	12.73 ± 12.66	9.55 ± 3.00	13.85 ± 21.87	259.72 ± 111.43	15.94 ± 8.07	62.36 ± 31.41
qwen25_0.5b_instruct_q3_K_S	27.30 ± 218.86	4.82 ± 2.31	45.29 ± 300.07	206.67 ± 105.84	16.43 ± 4.93	60.10 ± 126.40
qwen25_1.5b_instruct_q4_1	10.60 ± 18.21	7.08 ± 0.48	13.14 ± 36.77	158.28 ± 85.26	16.14 ± 5.13	41.05 ± 29.17
qwen25_1.5b_instruct_q4_K_M	8.36 ± 6.13	7.08 ± 0.86	9.27 ± 1.32	125.84 ± 96.84	15.39 ± 4.86	33.19 ± 22.00
qwen25_1.5b_instruct_q4_0	14.55 ± 72.90	7.01 ± 0.14	10.16 ± 5.02	130.84 ± 83.80	16.02 ± 8.20	35.71 ± 34.01
qwen25_1.5b_instruct_q4_K_S	11.04 ± 40.66	7.03 ± 0.17	9.43 ± 1.23	133.92 ± 91.46	15.38 ± 4.80	35.36 ± 27.66
qwen25_1.5b_instruct_q3_K_L	10.67 ± 15.45	7.26 ± 0.83	16.48 ± 38.77	144.63 ± 79.20	16.19 ± 8.63	39.05 ± 28.57
qwen25_1.5b_instruct_q3_K_M	12.60 ± 23.88	7.24 ± 0.93	26.62 ± 66.98	151.29 ± 96.29	15.19 ± 8.56	42.59 ± 39.33
qwen25_1.5b_instruct_q3_K_S	13.32 ± 20.97	8.22 ± 4.75	13.29 ± 35.52	196.29 ± 106.73	16.02 ± 7.71	49.43 ± 35.14
Average	14.82 ± 36.83	8.00 ± 3.31	21.63 ± 39.97	162.49 ± 89.08	15.51 ± 8.17	44.49

Table 8. Average accuracy of models.

Dataset Model	bigbenchhard	commonsenseqa	gsm8k	humaneval	truthfulqa	Average
gemma2_2b_instruct_q3_K_L	0.35 ± 0.48	0.67 ± 0.47	0.07 ± 0.26	0.25 ± 0.43	0.37 ± 0.48	0.34 ± 0.43
gemma2_2b_instruct_q3_K_M	0.36 ± 0.48	0.67 ± 0.47	0.07 ± 0.26	0.77 ± 0.42	0.36 ± 0.48	0.45 ± 0.42
gemma2_2b_instruct_q3_K_S	0.34 ± 0.47	0.70 ± 0.46	0.04 ± 0.21	0.02 ± 0.13	0.40 ± 0.49	0.30 ± 0.35
llama3.2_1b_instruct_fp16	0.16 ± 0.37	0.35 ± 0.48	0.03 ± 0.16	0.62 ± 0.49	0.31 ± 0.46	0.29 ± 0.39
llama3.2_1b_instruct_q8_0	0.17 ± 0.37	0.38 ± 0.49	0.04 ± 0.18	0.85 ± 0.36	0.34 ± 0.48	0.35 ± 0.38
llama3.2_1b_instruct_q4_1	0.15 ± 0.35	0.23 ± 0.43	0.04 ± 0.18	0.84 ± 0.37	0.27 ± 0.44	0.30 ± 0.35
llama3.2_1b_instruct_q4_K_M	0.23 ± 0.43	0.43 ± 0.50	0.06 ± 0.24	0.87 ± 0.34	0.34 ± 0.47	0.39 ± 0.39
llama3.2_1b_instruct_q4_0	0.20 ± 0.40	0.27 ± 0.44	0.05 ± 0.22	0.79 ± 0.41	0.39 ± 0.49	0.34 ± 0.39
llama3.2_1b_instruct_q4_K_S	0.27 ± 0.44	0.43 ± 0.50	0.04 ± 0.18	0.84 ± 0.37	0.39 ± 0.49	0.39 ± 0.40
llama3.2_1b_instruct_q3_K_L	0.14 ± 0.35	0.34 ± 0.47	0.03 ± 0.17	0.34 ± 0.48	0.14 ± 0.35	0.20 ± 0.36
llama3.2_1b_instruct_q3_K_M	0.10 ± 0.30	0.13 ± 0.34	0.03 ± 0.17	0.54 ± 0.50	0.03 ± 0.17	0.17 ± 0.30
llama3.2_1b_instruct_q3_K_S	0.08 ± 0.26	0.26 ± 0.44	0.02 ± 0.14	0.49 ± 0.50	0.07 ± 0.26	0.18 ± 0.32
qwen25_0.5b_instruct_fp16	0.27 ± 0.45	0.32 ± 0.47	0.04 ± 0.21	0.74 ± 0.44	0.23 ± 0.42	0.32 ± 0.40
qwen25_0.5b_instruct_q8_0	0.29 ± 0.45	0.32 ± 0.47	0.04 ± 0.18	0.54 ± 0.50	0.22 ± 0.42	0.28 ± 0.40
qwen25_0.5b_instruct_q4_1	0.28 ± 0.45	0.36 ± 0.48	0.07 ± 0.26	0.75 ± 0.44	0.29 ± 0.46	0.35 ± 0.42
qwen25_0.5b_instruct_q4_K_M	0.25 ± 0.43	0.32 ± 0.47	0.06 ± 0.24	0.56 ± 0.50	0.26 ± 0.44	0.29 ± 0.42
qwen25_0.5b_instruct_q4_0	0.22 ± 0.42	0.28 ± 0.45	0.05 ± 0.22	0.72 ± 0.45	0.24 ± 0.43	0.30 ± 0.39
qwen25_0.5b_instruct_q4_K_S	0.25 ± 0.43	0.36 ± 0.48	0.08 ± 0.27	0.52 ± 0.50	0.26 ± 0.44	0.29 ± 0.43
qwen25_0.5b_instruct_q3_K_L	0.28 ± 0.45	0.30 ± 0.46	0.03 ± 0.17	0.72 ± 0.45	0.31 ± 0.46	0.33 ± 0.40
qwen25_0.5b_instruct_q3_K_M	0.26 ± 0.44	0.37 ± 0.48	0.07 ± 0.25	0.70 ± 0.46	0.24 ± 0.43	0.33 ± 0.41
qwen25_0.5b_instruct_q3_K_S	0.20 ± 0.40	0.26 ± 0.44	0.01 ± 0.10	0.77 ± 0.42	0.31 ± 0.46	0.31 ± 0.36
qwen25_1.5b_instruct_q4_1	0.33 ± 0.47	0.67 ± 0.47	0.09 ± 0.28	0.08 ± 0.27	0.29 ± 0.46	0.29 ± 0.39
qwen25_1.5b_instruct_q4_K_M	0.32 ± 0.47	0.62 ± 0.49	0.12 ± 0.33	0.20 ± 0.40	0.24 ± 0.43	0.30 ± 0.42
qwen25_1.5b_instruct_q4_0	0.33 ± 0.47	0.66 ± 0.48	0.09 ± 0.29	0.27 ± 0.44	0.24 ± 0.43	0.32 ± 0.42
qwen25_1.5b_instruct_q4_K_S	0.33 ± 0.47	0.57 ± 0.50	0.13 ± 0.34	0.18 ± 0.39	0.23 ± 0.42	0.29 ± 0.42
qwen25_1.5b_instruct_q3_K_L	0.29 ± 0.45	0.45 ± 0.50	0.09 ± 0.28	0.03 ± 0.17	0.32 ± 0.47	0.23 ± 0.37
qwen25_1.5b_instruct_q3_K_M	0.26 ± 0.44	0.40 ± 0.49	0.07 ± 0.25	0.02 ± 0.13	0.24 ± 0.43	0.20 ± 0.35
qwen25_1.5b_instruct_q3_K_S	0.26 ± 0.44	0.57 ± 0.50	0.03 ± 0.17	0.17 ± 0.37	0.18 ± 0.39	0.24 ± 0.37
Average Accuracy	0.25 ± 0.42	0.42 ± 0.47	0.06 ± 0.22	0.51 ± 0.40	0.27 ± 0.43	0.30

Even the most aggressively quantized variant (q3_K_S) achieves 0.31 ± 0.36 , only slightly below the full-precision model (0.32 ± 0.40). In contrast, qwen2.5_1.5b shows moderate accuracy loss with stronger quantization, dropping from 0.32 ± 0.42 (q4_0) to 0.20 ± 0.35 (q3_K_M), suggesting greater sensitivity to precision reduction than its smaller counterpart.

4.2.2 Task-Specific Accuracy Patterns.

Table 8 highlights task-specific accuracy variations and quantization sensitivity. HumanEval (code generation) has the highest average accuracy (0.51) but exhibits high variability. For example, gemma2_2b ranges from 0.02 to 0.77, while qwen2.5_1.5b varies from 0.02 to 0.27, showing inconsistent responses to quantization.

GSM8K (mathematical reasoning) is challenging across all models and quantization levels, with an average accuracy of 0.06. Notably, aggressive quantization does not significantly degrade performance, suggesting that higher precision offers little benefit for models struggling with complex reasoning.

CommonsenseQA shows a strong correlation between model size and accuracy, with larger models (gemma2_2b, qwen2.5_1.5b) outperforming smaller ones, regardless of quantization level. Gemma2_2b achieves 0.67–0.70 accuracy, while qwen2.5_1.5b reaches 0.57–0.67, compared to 0.26–0.43 for smaller models. For commonsense reasoning, model scale remains crucial, even under quantization constraints.

High standard deviations (0.35–0.45) across tasks indicate significant instance-level variability in model performance.

This suggests that overall accuracy metrics may not fully capture quantization effects, as some instances are highly impacted while others remain stable.

4.2.3 Energy-Accuracy Tradeoffs. Figure 6 highlights the tradeoff between energy consumption and average accuracy across all datasets. The Pareto frontier (dashed line) marks models that achieve optimal accuracy-energy efficiency balances. It includes only five models, showing that few configurations achieve truly optimal energy-accuracy tradeoffs across tasks. These Pareto-optimal models vary in model family and quantization technique.

- (1) Gemma2_2b_instruct_q3_K_M achieves the highest overall accuracy (0.45) despite aggressive q3 quantization, showing that effective quantization can maintain performance at low precision. It consumes 109 Joules/response on average, placing it mid-range in energy use.

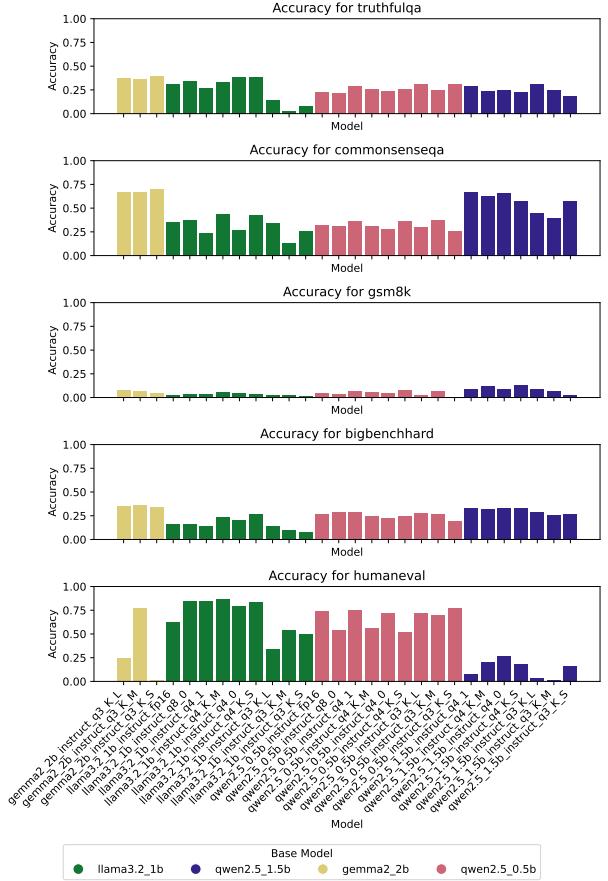


Fig. 5. Accuracy comparison across datasets for all model variants, grouped by model family, highlighting task-specific sensitivities to model architecture and quantization.

- (2) Llama3.2_1b_instruct_q4_K_S balances accuracy (0.39) and energy efficiency (40 Joules/response), making it ideal for performance-conscious, energy-constrained deployments.
- (3) The qwen2.5_0.5b base model dominates the low-energy Pareto frontier, with q4_1, q4_0, and q8_0 variants achieving 0.28–0.35 accuracy while consuming just 14–16 Joules. These configurations are ideal for energy-constrained environments.

No qwen2.5_1.5b variants reach the Pareto frontier despite their larger parameter count. While strong in CommonsenseQA, their energy-efficiency ratio is lower than other models across diverse tasks. The Pareto frontier shows diminishing returns—improving from qwen2.5_0.5b (0.35 accuracy) to llama3.2_1b (0.39 accuracy) nearly doubles energy use, while reaching gemma2_2b (0.45 accuracy) requires 40% more energy. This non-linear tradeoff is critical for deployment decisions, highlighting the energy cost of incremental accuracy gains.

Comparing the average Pareto frontier (Figure 6) with task-specific frontiers (Figures 6–11) reveals key differences. Models optimal on average may not be best for specific tasks, and vice versa. For example, qwen2.5_1.5b, absent from the average frontier, dominates in BigBenchHard and GSM8K. This underscores the need for task-specific model selection rather than relying solely on average performance.

In BigBenchHard (Figure 7), qwen2.5_1.5b and gemma2_2b achieve the highest accuracy (0.33–0.36), but qwen2.5_1.5b is more energy-efficient. At the lower end of the Pareto frontier, qwen2.5_0.5b_q8_0 offers 0.29 accuracy with minimal energy use (28 Joules/response), making it a strong low-power option.

In CommonsenseQA (Figure 8), larger models (gemma2_2b, qwen2.5_1.5b) dominate in accuracy (0.57–0.70), emphasizing the role of model scale in common-sense reasoning. However, smaller models, llama3.2_1b_q4_K, achieve lower accuracy (0.43) but offer greater energy efficiency, appearing on the Pareto frontier.

GSM8K (Figure 9) highlights the challenges of mathematical reasoning, with low accuracy across all models. Qwen2.5_0.5b_q4_K_S offers the best efficiency-accuracy balance (0.08 accuracy, 26 Joules/response), while qwen2.5_1.5b_q4_K_S achieves higher accuracy (0.13) but at 2× the energy cost (63 Joules/response) for accuracy-critical deployments.

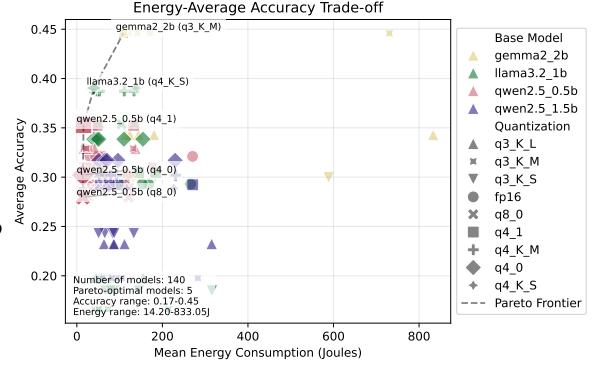


Fig. 6. Energy-accuracy tradeoff across all datasets. The Pareto frontier (dashed line) highlights optimal models, with points colored by the dataset and shaped by the quantization technique. Pareto-optimal models are annotated.

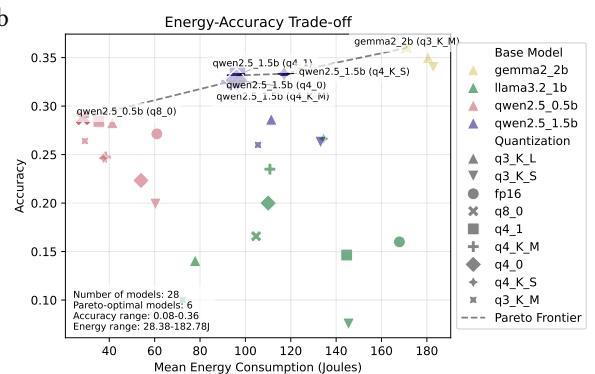


Fig. 7. Energy-accuracy tradeoff for BigBenchHard. The Pareto frontier highlights six optimal models, with qwen2.5_0.5b and qwen2.5_1.5b variants achieving the best balance for this complex reasoning task.

HumanEval (Figure 10) presents sharp energy-accuracy tradeoffs, with model llama3.2_1b_q4_K_M achieving high accuracy (0.87) in code generation at moderate energy use (134 Joules/response). Notably, qwen2.5_0.5b variants with aggressive quantization also perform well, reaching 0.70–0.77 accuracy with slightly lower energy consumption (114–133 Joules/response).

TruthfulQA (Figure 11) features a distinct Pareto frontier with five optimal models. Gemma2_2b_q3_K_S (0.40 accuracy, 148 Joules/response) and model llama3.2_1b_q4_K_S (0.39 accuracy, 51 Joules/response) achieve high accuracy but differ in energy use. Meanwhile, three qwen2.5_0.5b variants (0.24–0.31 accuracy, 21–23 Joules/response) offer 60% lower energy consumption than llama3.2_1b, trading some accuracy for efficiency. Notably, qwen2.5_1.5b variants are absent, suggesting that larger models don't always excel in factual reasoning under resource constraints.

Comparing task-specific Pareto frontiers reveals distinct energy-accuracy tradeoff patterns. HumanEval and CommonsenseQA show steep accuracy gains at high energy costs, while GSM8K exhibits diminishing returns, as more powerful models offer little improvement in this task. BigBenchHard and TruthfulQA follow a moderate tradeoff, where small energy increases yield notable accuracy gains. These task-dependent patterns are crucial for deployment decisions based on performance constraints.

4.2.4 Model Size and Accuracy Scaling

Figure 12 explores the scaling of accuracy with model size across quantization techniques, revealing complex, task-dependent patterns.

- For qwen2.5_1.5b, accuracy increases with model size across most tasks (slopes: 0.21 to 0.61), suggesting higher-precision variants better retain model capabilities.
- Gemma2_2b exhibits mixed scaling patterns. In CommonsenseQA (-0.24) and TruthfulQA (-0.20), aggressive quantization can maintain or enhance accuracy. However, HumanEval (2.08) shows a strong positive correlation, benefiting significantly from higher precision.
- Llama3.2_1b shows modest positive scaling (0.01 to 0.13) across tasks, except for GSM8K (-0.005), where quantization has little effect on its low performance.
- Qwen2.5_0.5b exhibits task-dependent scaling, with positive slopes in BigBenchHard (0.03), negative in TruthfulQA (-0.06), and near-zero in HumanEval (0.001), indicating the minimal impact of quantization across tasks.

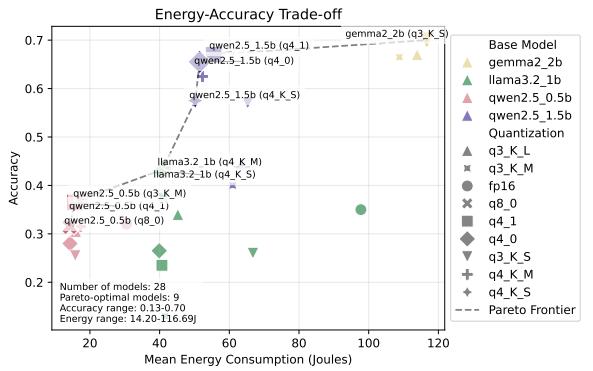


Fig. 8. Energy-accuracy tradeoff for CommonsenseQA. The Pareto frontier identifies nine optimal models, with gemma2_2b variants achieving the highest accuracy (0.70) at higher energy costs.

Meanwhile, three qwen2.5_0.5b variants (0.24–0.31 accuracy, 21–23 Joules/response) offer 60% lower energy consumption than llama3.2_1b, trading some accuracy for efficiency. Notably, qwen2.5_1.5b variants are absent, suggesting that larger models don't always excel in factual reasoning under resource constraints.

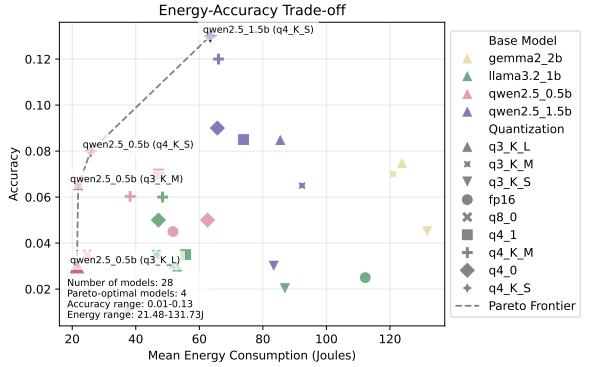


Fig. 9. Energy-accuracy tradeoff for GSM8K. The Pareto frontier includes four models, with qwen2.5_1.5b and qwen2.5_0.5b achieving the highest accuracy for mathematical reasoning despite low overall performance across models.

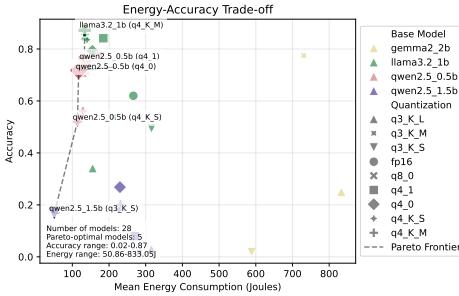


Fig. 10. Energy-accuracy tradeoff for HumanEval. The Pareto frontier highlights five optimal configurations, with llama3.2_1b_q4_K_M achieving the highest accuracy (0.87) for code generation while maintaining a moderate energy cost.

These scaling patterns highlight the complex interplay between model architecture, quantization technique, and task-specific requirements when optimizing for both accuracy and energy efficiency.

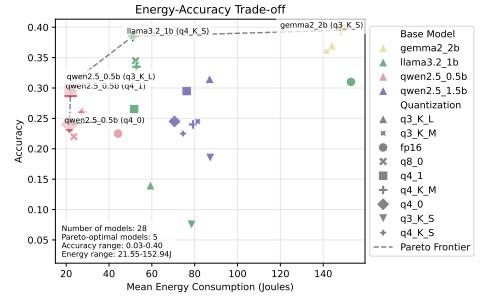


Fig. 11. Energy-accuracy tradeoff for TruthfulQA. The Pareto frontier highlights optimal configurations for factual reasoning, with qwen2.5_1.5b (q3_K_S) achieving 0.18 accuracy at the lowest energy cost.

RQ2 Conclusion. Our analysis reveals that quantization impacts accuracy differently across models and tasks. While higher-precision variants generally retain better accuracy, some aggressive quantization techniques perform comparably, particularly in CommonsenseQA and TruthfulQA. Gemma2_2b benefits from higher precision in code generation, whereas qwen2.5_0.5b shows minimal sensitivity to quantization across tasks. The Pareto frontier analysis highlights that optimal energy-accuracy tradeoffs are highly task-dependent, emphasizing the need for task-specific model selection rather than relying solely on average performance metrics.

4.3 How do different quantization techniques impact inference speed on an edge device? (RQ3)

To address RQ3, we analyzed latency (time from prompt request to response completion) and token generation throughput (tokens generated per second).

4.3.1 Token Generation Throughput. Figure 13 displays token generation throughput across all models and datasets, showing significant variation by model family and quantization technique. Higher values indicate faster generation. Model variants are grouped by base model and the color of the bars represent different dataset. Qwen2.5_0.5b consistently achieves the highest throughput (3.2–10.1 tokens/sec), with quantized variants outperforming fp16 (3.2–4.1 tokens/sec) by 2–3×, particularly in q4 and q3 formats.

Llama3.2_1b exhibits variable throughput, heavily influenced by the quantization technique. Q4 variants (1.9–4.8 tokens/sec) generally outperform FP16 (1.4–1.7 tokens/sec) and q8_0 (2.3–3.3 tokens/sec), suggesting that aggressive quantization can enhance performance. Larger models (gemma2_2b, qwen2.5_1.5b) exhibit lower throughput, with gemma2_2b limited

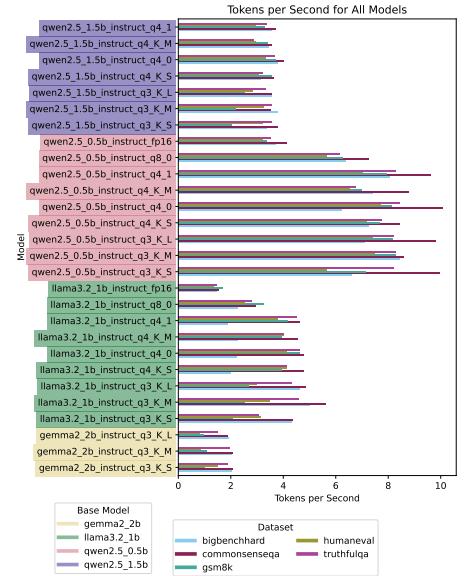


Fig. 13. Token generation throughput (tokens per second) for all models across the datasets.

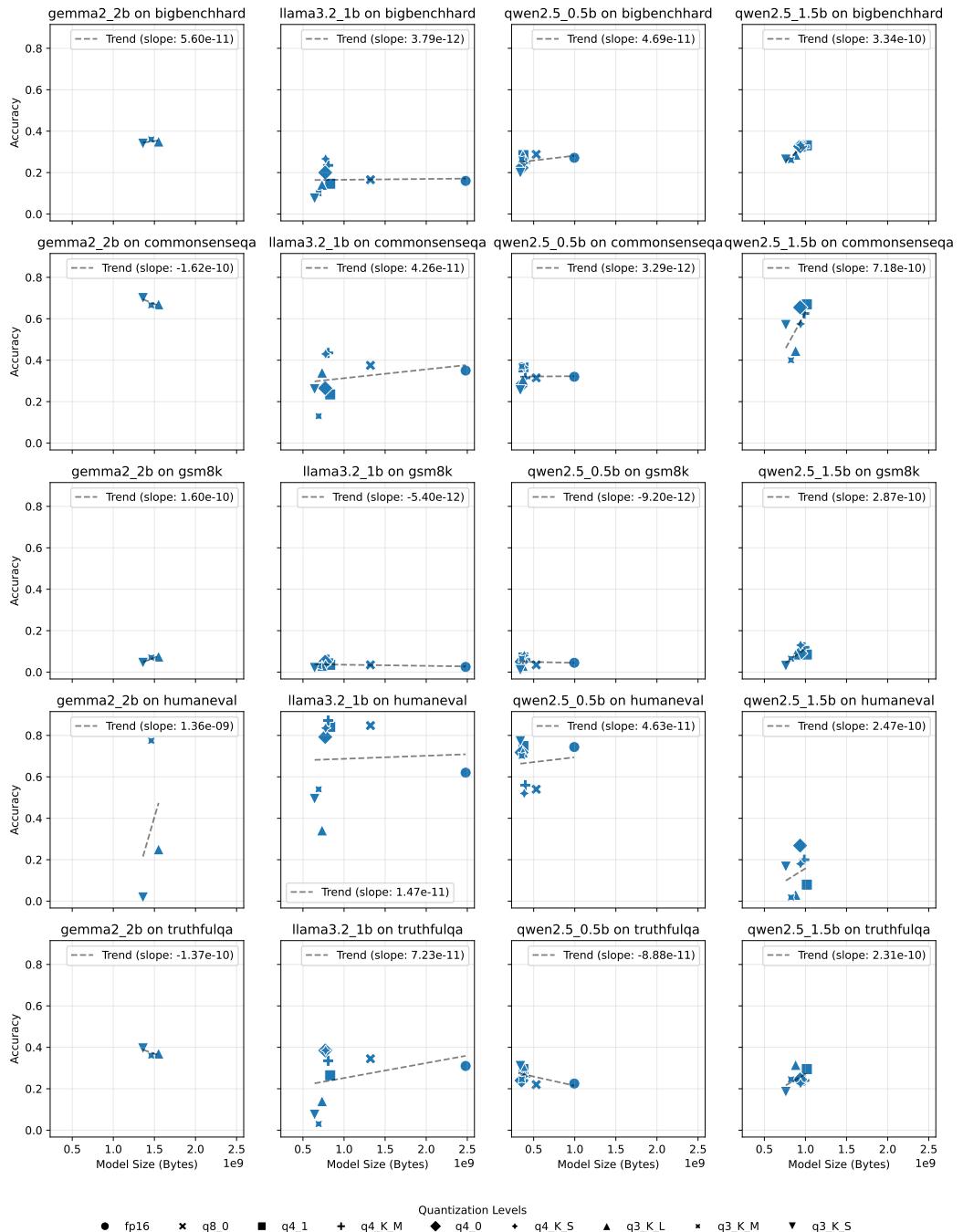


Fig. 12. Relationship between model size (in bytes) and accuracy across different base models and datasets. Trend lines show how accuracy scales with increasing or decreasing model size resulting from different quantization techniques. Positive slopes indicate that higher precision (larger model sizes) generally correlates with better accuracy.

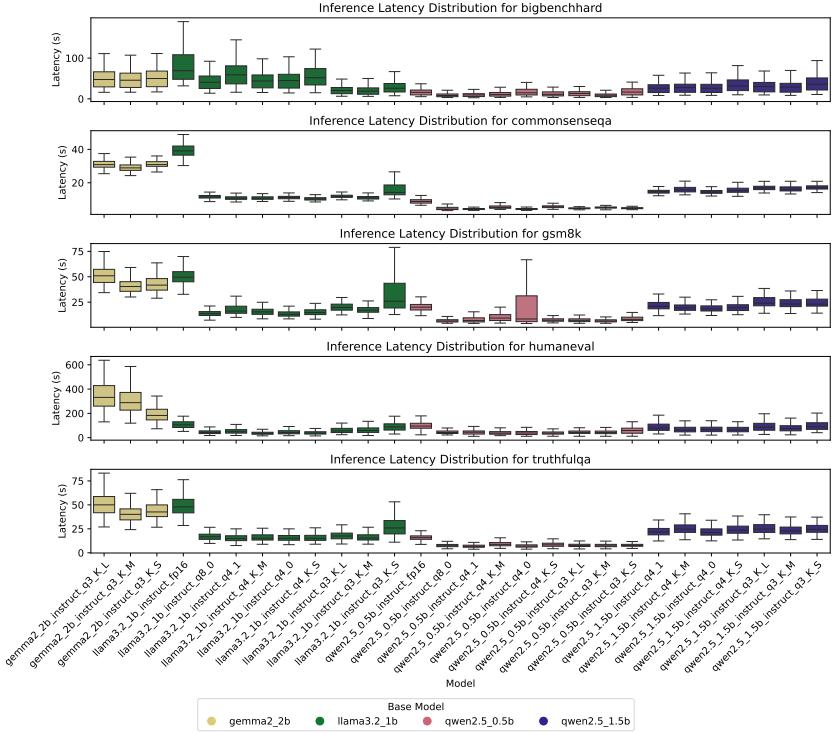


Fig. 14. Inference latency distribution (in seconds) for all models across the five benchmark datasets. Lower values indicate faster response times. Note the significant differences in scale across benchmarks, particularly for HumanEval which requires longer responses.

to 0.8–2.1 tokens/sec and qwen2.5_1.5b performing slightly better at 2.0–4.0 tokens/sec. Unlike smaller models, quantization has less impact on throughput in these larger architectures.

Throughput varies by task, with CommonsenseQA achieving the highest rates due to its short, simple responses, while GSM8K (math reasoning) and HumanEval (code) show lower throughput.

4.3.2 Inference Latency. Figure 14 shows inference latency across models and benchmarks, varying by architecture, quantization, and task. Table 9 summarizes average latency, with qwen2.5_0.5b being

the fastest (17.88s avg.), particularly q8_0 (9.84s). Llama3.2_1b has moderate latency (36.33s avg.), where quantized variants (26–31s) outperform fp16 (59.66s). Larger models show higher latency, with gemma2_2b (86.43s avg.) and qwen2.5_1.5b (33.85s avg.).

Quantization significantly reduces latency, with FP16 to quantized formats improving performance by 56% (llama3.2_1b) and 69% (qwen2.5_0.5b) when comparing against the model variants with the lowest latency. However, differences among q8, q4, and q3 are less consistent. In llama3.2_1b, q8 (26.32s) and q4 (27.97s) perform similarly, while q3 (31.39s) is slightly slower. In qwen2.5_0.5b, q8_0 (9.84s) is fastest, outperforming q4 (14.09s) and q3 (16.19s). This indicates that despite reducing

Table 9. Average inference latency (seconds) by base model and quantization level. The N/A values are ignored when computing averages.

	gemma2_2b	llama3.2_1b	qwen2.5_0.5b	qwen2.5_1.5b	Average
fp16	N/A	59.66	31.40	N/A	45.53
q8_0	N/A	26.32	9.84	N/A	18.08
q4 variants (avg)	N/A	27.97	14.09	29.42	23.83
q3 variants (avg)	86.43	31.39	16.19	38.28	43.07
Average	86.43	36.33	17.88	33.85	

model size, more aggressive quantization may introduce computational overhead, limiting latency gains on resource-constrained edge devices.

Number of parameters alone does not determine latency. Despite having 1.5B parameters, qwen2.5_1.5b (33.85s avg.) outperforms llama3.2_1b (36.33s avg.), which has only 1B parameters. This suggests that architecture and optimization play a greater role in inference speed than parameter count.

Table 10 shows HumanEval has the highest latency (94.84s avg.), far exceeding other benchmarks (13.96–33.19s). This aligns with its longer responses (Table 7) and a strong correlation (>0.9) between response length and latency across most models (as derived from Figure 4).

Table 10. Average inference latency (seconds) by the datasets.

Dataset	Average Latency (s)
bigbenchhard	33.19
commonsenseqa	13.96
gsm8k	22.52
humaneval	94.84
truthfulqa	21.07
Average	37.12

RQ3 Conclusion. Our analysis highlights the impact of quantization, model architecture, and task complexity on inference latency and token throughput. Quantized models significantly reduce latency, with fp16 to quantized formats improving performance by up to 69%. However, more aggressive quantization (e.g., q3) does not always yield additional speed gains due to computational overhead. Smaller models (qwen2.5_0.5b) consistently achieve the highest throughput, while larger models (gemma2_2b) exhibit the slowest performance. Task complexity also plays a major role, with HumanEval showing the highest latency (94.84s avg.) due to longer response lengths, strongly correlating with latency increases across models.

5 Threats to Validity

5.1 Internal Validity

Internal validity concerns factors that could affect **the accuracy of our measurements** and results. One primary threat is **energy measurement variability** due to hardware or software fluctuations. While we use Joulescope for precise power monitoring, external factors such as background processes on the Raspberry Pi (RPi), thermal throttling, and power fluctuations could introduce noise into the energy readings. Additionally, model inference times may vary due to system load and memory management, potentially influencing the energy efficiency results. To mitigate these issues, we ensured **consistent experimental conditions**, active and passive cooling of the Raspberry Pi, synchronized system clocks, and repeated measurements for verification.

Another potential internal validity threat is **inference timeout handling**. The inference timeout parameter prevents excessive runtime, but an improperly set threshold could **prematurely terminate** models that require more time for complex tasks. While we used a **default timeout of 20 minutes**, tuning this parameter for different models may affect the results. We mitigated this by **manually verifying model behavior** and adjusting timeouts when necessary.

5.2 External Validity

External validity relates to how well our findings **generalize to other settings, hardware platforms, and real-world applications**. Our experiments were conducted on a single edge device (Raspberry Pi 4 with 4GB RAM), which may not be representative of other edge AI hardware such as Jetson Nano, Coral TPU, or mobile AI accelerators. Differences in hardware architectures, power management strategies, and optimization techniques could lead to varying energy consumption trends. While our results provide insights into LLM quantization trade-offs on constrained devices, further experiments on diverse edge platforms are necessary for broader generalization.

Additionally, our evaluation focuses on a specific set of benchmark datasets (CommonsenseQA, BBH, TruthfulQA, GSM8K, and HumanEval). While these datasets cover diverse reasoning, mathematical, and programming tasks, they may not fully represent the entire spectrum of real-world LLM use cases. Performance and energy consumption could vary in interactive applications, dialogue-based systems, and multilingual environments. Future work should explore a wider range of tasks and real-world deployment scenarios.

5.3 Construct Validity

Construct validity refers to the degree to which our chosen **metrics and evaluation methods** accurately measure what we intend to study. Our primary focus is energy efficiency, measured via total energy consumption during inference. While this provides a direct measure of power usage, other efficiency-related factors, such as latency, throughput, and cost per inference, are not explicitly analyzed. In real-world deployment, energy efficiency should be evaluated alongside latency constraints and task performance requirements.

Furthermore, we evaluate quantized models in terms of accuracy and energy consumption, but we do not explicitly measure **quantization-induced accuracy degradation** beyond standard benchmark results. Some quantization techniques may introduce specific types of errors, such as precision loss in arithmetic tasks or sensitivity to specific linguistic structures. Expanding the evaluation to quantization-aware robustness assessments could improve construct validity.

5.4 Future Considerations

We ensure that our findings provide valuable insights into energy-efficient LLM inference on edge devices, while acknowledging the need for further validation across different environments. Future work should (i) extend the evaluation to multiple edge devices (e.g., Jetson Nano, Edge TPUs) to assess hardware-dependent energy efficiency trends, (ii) analyze a broader range of real-world AI tasks, including conversational AI and multilingual inference, and (iii) incorporate additional efficiency metrics, such as inference latency, throughput, memory usage, and task-specific accuracy degradation due to quantization.

6 Related Work

6.1 Energy Efficiency in Large Language Models

Energy Consumption in LLM Inference. Recent studies have analyzed the impact of model architecture, token processing speed, and parallelization on LLM energy efficiency. Argerich and Patiño-Martínez [1] identified model size, layer count, and quantization levels as key factors influencing power consumption, showing that quantized LLMs significantly reduce energy usage. Doo et al. [15] examined accuracy-energy trade-offs in medical AI, finding that smaller fine-tuned models (e.g., Vicuna 1.5, 7B/13B) were both more energy-efficient and accurate than larger models like LLaMA 2 (70B), which consumed over seven times more power than its 7B counterpart. These findings underscore the importance of choosing the right model architecture to optimize both predictive performance and energy efficiency. Husom et al. [30] examined LLM energy consumption during inference, finding that response length and duration strongly correlate with power usage, while prompt complexity has minimal impact. Their results suggest that controlling response generation is more effective for reducing energy consumption than simplifying prompts.

Reducing the Carbon Footprint of LLMs. Various strategies have been proposed to reduce the carbon footprint of LLM training and deployment. Shi et al. [63] introduced Avatar, an optimization framework reducing model size, inference latency, and energy consumption while maintaining accuracy, achieving up to $184\times$ lower energy usage and $157\times$ lower carbon emissions through

multi-objective tuning. Bai et al. [3] categorized energy-efficient LLM techniques, highlighting architecture design, training efficiency, and system-level optimizations. Their study underscores the role of energy-aware fine-tuning and adaptive precision scaling in minimizing carbon footprint.

Energy-Efficient LLM Deployment on Edge. Deploying LLMs on edge devices is challenging due to limited battery life and computational resources, requiring energy-efficient solutions. Yuan et al. [81] proposed intelligent offloading strategies, dynamically balancing computations between mobile devices and edge servers, minimizing energy consumption. Tian et al. [67] introduced GreenLLM, a framework optimizing memory, power, and computational efficiency through pruning, achieving a 34.1% energy and 33.5% latency reduction within acceptable performance loss.

While prior studies have analyzed **LLM energy consumption, carbon footprint, and edge deployment in a limited scope or even without any edge focus** (e.g., estimating power usage through software-based profiling, evaluating only select quantization methods, or focusing on theoretical energy models rather than real-world edge deployments), our work extends this research by **conducting real-world energy measurements using hardware-based profiling, benchmarking 28 quantized LLMs from the Ollama framework (applying by default PTQ and weight-only quantization) on an edge device, and analyzing energy-performance trade-offs across quantization levels and task types**. By integrating hardware-level energy profiling with LLM benchmarking, we provide a comprehensive evaluation of energy-efficient deployment strategies.

6.2 Quantized Language Models on Edge Devices

Several studies [10, 24, 48, 57, 60, 61, 66] have investigated quantized LLMs in edge environments to address limited computational resources, memory constraints, and energy efficiency requirements.

Weight and Activation Quantization for LLMs on Edge. Shen et al. [60] introduced Agile-Quant, an activation-guided quantization framework that simultaneously quantizes weights and activations, achieving up to $2.55\times$ speedup on edge devices over FP16 models. Unlike weight-only quantization, it enhances inference efficiency by optimizing activations. Similarly, Tan et al. [66] proposed MobileQuant, a post-training quantization method for on-device language models. The approach jointly optimizes weight transformation and activation range parameters, reducing latency and energy consumption by 20%-50% compared to other on-device quantization strategies. While weight and activation quantization improves inference efficiency, latency, and energy consumption, it comes with trade-offs. Weight-only quantization retains higher computational efficiency and accuracy, as activations remain in full or higher precision. In contrast, weight and activation quantization provides greater memory savings, making it more suitable where both model and activations must fit within strict memory limits.

Optimizing Memory and Latency for Quantized LLMs. Rahman et al. [57] evaluated quantized transformer models, MobileBERT [65], on edge devices, showing that the converted and quantized MobileBERT models have $160\times$ smaller footprints for a 4.1% drop in accuracy. Shen et al. [61] proposed EdgeQAT, a quantization-aware training framework for the optimization of lightweight LLMs to achieve inference acceleration on Edge devices. Using entropy and distribution guided techniques, it achieves an on-device speedup of $2.37\times$ compared with its FP16 counterparts.

Existing studies focus on reducing memory footprint, improving inference speed, and preserving accuracy in quantized LLMs but **often overlook energy consumption trade-offs across different quantization techniques**. Different than these studies, our work conducts **real-world energy measurements using hardware-based profiling, benchmarking quantized LLMs from the Ollama framework on an edge device, and analyzing energy-performance-accuracy trade-offs across quantization levels and task types**.

6.3 Existing Benchmarks for Evaluating LLM Performance and Efficiency on Edge

Several benchmarking frameworks have been developed to assess the performance, efficiency, and resource footprint of LLMs deployed on edge devices. Traditional LLM benchmarks often focus on accuracy and latency, but emerging frameworks are now incorporating memory consumption and energy efficiency as critical evaluation metrics. **PalmBench** [40] is a comprehensive benchmark that evaluates quantized LLMs on mobile platforms, analyzing memory usage, GPU execution time, and power consumption. **The Edge-Device Large Language Model Competition (NeurIPS 2024)** [45] provides a platform for benchmarking LLM inference efficiency on edge devices, with a focus on model size, throughput, and latency. **MLPerf Tiny** [4], **MLPerf Power** [70], **MLPerf Inference** [58], and **Ai benchmark** [31] offer standardized AI benchmarking tools that include power-aware evaluations, making them valuable for assessing energy-efficient AI models.

Task-specific datasets like CommonsenseQA and HumanEvalserve as standardized benchmarks for evaluating LLM reasoning, mathematical problem-solving, and programming under computational constraints. Our work builds upon these datasets by introducing energy consumption as a key performance metric. While previous evaluations have focused primarily on memory and latency, our study provides a comprehensive energy-efficiency and LLM output accuracy analysis of quantized LLMs deployed on edge, leveraging high-resolution hardware-based energy monitoring.

7 Conclusion

Our study provides a comprehensive evaluation of quantized LLMs on edge devices, analyzing the trade-offs between energy efficiency, accuracy, and inference speed. We find that quantization significantly reduces energy consumption and latency, but the impact on accuracy varies across tasks and model families. While q3 and q4 quantization achieve substantial energy savings, extreme compression can degrade performance in complex reasoning tasks. Our findings emphasize the importance of selecting the right quantization strategy based on task requirements and deployment constraints. Future work should explore adaptive quantization techniques to further optimize efficiency and performance for real-world edge AI applications.

Acknowledgments

The work has been conducted as part of the ENFIELD project (101120657) funded by the European Commission within the HEU Programme, and the National Research Foundation of Korea (NRF) grant RS-2023-00268071.



References

- [1] Mauricio Fadel Argerich and Marta Patiño-Martínez. 2024. Measuring and Improving the Energy Efficiency of Large Language Models Inference. *IEEE Access* (2024).
- [2] Merve Astekin, Max Hort, and Leon Moonen. 2024. A Comparative Study on Large Language Models for Log Parsing. In *ESEM'24*. 234–244.
- [3] Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. 2024. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625* (2024).
- [4] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Jeremy Holleman, Nat Jeffries, Csaba Kiraly, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, et al. 2021. Mlperf tiny benchmark. *arXiv preprint arXiv:2106.07597* (2021).
- [5] BIG-Bench-Hard. Visited in 2024. <https://github.com/suzgunmirac/BIG-Bench-Hard>.
- [6] Fenglong Cai, Dong Yuan, Zhe Yang, and Lizhen Cui. 2024. Edge-LLM: A Collaborative Framework for Large Language Model Serving in Edge Computing. In *ICWS'24*. 799–809.
- [7] Yuji Chai, Mujin Kwen, David Brooks, and Gu-Yeon Wei. 2025. FlexQuant: Elastic Quantization Framework for Locally Hosted LLM on Edge Devices. *arXiv preprint arXiv:2501.07139* (2025).
- [8] Mark Chen, Jerry Tworek, and etc. 2021. Evaluating Large Language Models Trained on Code. (2021). *arXiv:2107.03374 [cs.LG]*

- [9] CommonsenseQA. Visited in 2024. <https://www.tau-nlp.sites.tau.ac.il/commonsenseqa>.
- [10] Tolga Çöplü, Marc Loedi, Arto Bendiken, Mykhailo Makohin, Joshua J Bouw, and Stephen Cobb. 2023. A performance evaluation of a quantized large language model on various smartphones. *arXiv preprint arXiv:2312.12472* (2023).
- [11] NVIDIA Corporation. 2023. NVIDIA System Management Interface. <https://docs.nvidia.com/deploy/nvidia-smi/index.html> 2024-05-03.
- [12] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2023. Benchmark probing: Investigating data leakage in large language models. In *NeurIPS 2023 Workshop on Backdoors in Deep Learning-The Good, the Bad, and the Ugly*.
- [13] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 () 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems* 35 (2022), 30318–30332.
- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [15] Florence X Doo, Dharmam Savani, Adway Kanhere, Ruth C Carlos, Anupam Joshi, Paul H Yi, and Vishwa S Parekh. 2024. Optimal large language model characteristics to balance accuracy and energy use for sustainable medical applications. *Radiology* 312, 2 (2024), e240320.
- [16] Experiment Repo. Visited in 2025. <https://github.com/ejhusom/neurips-edge-lm-challenge-sampled/>.
- [17] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).
- [18] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.
- [19] Othmane Friha, Mohamed Amine Ferrag, Burak Kantarci, Burak Cakmak, Arda Ozgun, and Nassira Ghoualmi-Zine. 2024. Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. *IEEE Open Journal of the Communications Society* (2024).
- [20] ggml-org. Visited in 2025. llama.cpp. <https://github.com/ggml-org/llama.cpp>.
- [21] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*. Chapman and Hall/CRC, 291–326.
- [22] GSM8K. Visited in 2024. <https://github.com/openai/grade-school-math>.
- [23] Jude Harris, Rappy Saha, Wenhai Hu, and José Cano. 2024. Designing Efficient LLM Accelerators for Edge Devices. *arXiv preprint arXiv:2408.00462* (2024).
- [24] Jahid Hasan. 2024. Optimizing Large Language Models through Quantization: A Comparative Analysis of PTQ and QAT Techniques. *arXiv preprint arXiv:2411.06084* (2024).
- [25] Xing Hu, Yuan Chen, Dawei Yang, Sifan Zhou, Zhihang Yuan, Jiangyong Yu, and Chen Xu. 2024. I-LLM: Efficient Integer-Only Inference for Fully-Quantized Low-Bit Large Language Models. *arXiv preprint arXiv:2405.17849* (2024).
- [26] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shimeng Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. 2024. Billm: Pushing the limit of post-training quantization for llms. *arXiv preprint arXiv:2402.04291* (2024).
- [27] Wei Huang, Xudong Ma, Haotong Qin, Xingyu Zheng, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. How good are low-bit quantized llama3 models? an empirical study. *arXiv e-prints* (2024), arXiv-2404.
- [28] Wei Huang, Xingyu Zheng, Xudong Ma, Haotong Qin, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. An empirical study of llama3 quantization: From llms to mllms. *Visual Intelligence* 2, 1 (2024), 36.
- [29] HumanEval. Visited in 2024. <https://github.com/openai/human-eval>.
- [30] Erik Johannes Husom, Arda Goknil, Lwin Khin Shar, and Sagar Sen. 2024. The Price of Prompting: Profiling Energy Use in Large Language Models Inference. *arXiv preprint arXiv:2407.16893* (2024).
- [31] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. 2019. Ai benchmark: All about deep learning on smartphones in 2019. In *ICCVW'19*. 3617–3635.
- [32] Ajay Jaiswal, Zhe Gan, Xianzhi Du, Bowen Zhang, Zhangyang Wang, and Yinfei Yang. 2023. Compressing llms: The truth is rarely pure and never simple. *arXiv preprint arXiv:2310.01382* (2023).
- [33] Renren Jin, Jiangcun Du, Wuwei Huang, Wei Liu, Jian Luan, Bin Wang, and Deyi Xiong. 2024. A comprehensive evaluation of quantization strategies for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*. 12186–12215.
- [34] Joulemeter. Visited in 2025. <https://www.microsoft.com/en-us/research/project/joulemeter-computational-energy-measurement-and-optimization/>.
- [35] Joulescope. Visited in 2025. <https://www.joulescope.com/>.
- [36] Aria Khoshirsat, Giovanni Perin, and Michele Rossi. 2024. Decentralized LLM Inference over Edge Networks with Energy Harvesting. *arXiv preprint arXiv:2408.15907* (2024).

- [37] Jiedong Lang, Zhehao Guo, and Shuyu Huang. 2024. A Comprehensive Study on Quantization Techniques for Large Language Models. *arXiv preprint arXiv:2411.02530* (2024).
- [38] Jiahuan Li, Hao Zhou, Shujian Huang, Shanbo Cheng, and Jiajun Chen. 2024. Eliciting the translation ability of large language models via multilingual finetuning with translation instructions. *Transactions of the Association for Computational Linguistics* 12 (2024), 576–592.
- [39] Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Evaluating quantized large language models. *arXiv preprint arXiv:2402.18158* (2024).
- [40] Yilong Li, Jingyu Liu, Hao Zhang, M Badri Narayanan, Utkarsh Sharma, Shuai Zhang, Pan Hu, Yijing Zeng, Jayaram Raghuram, and Suman Banerjee. 2024. PalmBench: A Comprehensive Benchmark of Compressed Large Language Models on Mobile Platforms. *arXiv preprint arXiv:2410.05315* (2024).
- [41] Zonghang Li, Wenjiao Feng, Mohsen Guizani, and Hongfang Yu. 2024. TPI-LLM: Serving 70B-scale LLMs Efficiently on Low-resource Edge Devices. *arXiv preprint arXiv:2410.00531* (2024).
- [42] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.
- [43] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems* 6 (2024), 87–100.
- [44] Peiyu Liu, Zikang Liu, Ze-Feng Gao, Dawei Gao, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2023. Do emergent abilities exist in quantized large language models: An empirical study. *arXiv preprint arXiv:2307.08072* (2023).
- [45] Shiwei Liu, Kai Han, Adriana Fernandez-Lopez, Ajay Kumar Jaiswal, Zahra Atashgahi, Boqian Wu, Edoardo Ponti, Callie Hao, Rebekka Burkholz, Olga Saukh, et al. 2024. Edge-LLMs: Edge-Device Large Language Model Competition. In *NeurIPS 2024 Competition Track*.
- [46] Yijun Liu, Yuan Meng, Fang Wu, Shenhao Peng, Hang Yao, Chaoyu Guan, Chen Tang, Xinzhu Ma, Zhi Wang, and Wenwu Zhu. 2024. Evaluating the generalization ability of quantized llms: Benchmark, analysis, and toolbox. *arXiv preprint arXiv:2406.12928* (2024).
- [47] Llama.cpp. Visited in 2024. <https://github.com/ggerganov/llama.cpp>.
- [48] Yu Mao, Weilan Wang, Hongchao Du, Nan Guan, and Chun Jason Xue. 2024. On the compressibility of quantized large language models. *arXiv preprint arXiv:2403.01384* (2024).
- [49] Monsoon. Visited in 2025. <https://www.msoon.com/high-voltage-power-monitor/>.
- [50] Ollama framework. Visited in 2025. <https://github.com/ollama/ollama>.
- [51] Ollama Library. Visited in 2025. <https://ollama.com/library>.
- [52] Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. 2022. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *arXiv preprint arXiv:2206.09557* (2022).
- [53] PowerAPI. Visited in 2025. PyJoules: Python-based energy measurement library. <https://github.com/powerapi/pyJoules>.
- [54] Ruiyang Qin, Dancheng Liu, Zheyu Yan, Zhaoxuan Tan, Zixuan Pan, Zheng Jia, Meng Jiang, Ahmed Abbasi, Jinjun Xiong, and Yiyu Shi. 2024. Empirical Guidelines for Deploying LLMs onto Resource-constrained Edge Devices. *arXiv preprint arXiv:2406.03777* (2024).
- [55] Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. 2024. Mobile edge intelligence for large language models: A contemporary survey. *arXiv preprint arXiv:2407.18921* (2024).
- [56] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
- [57] Mohammad Wali Ur Rahman, Murad Mehrab Abrar, Hunter Gibbons Copening, Salim Hariri, Sicong Shao, Pratik Satam, and Soheil Salehi. 2023. Quantized transformer language model implementations on edge devices. In *ICMLA'23*. 709–716.
- [58] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2020. Mlperf inference benchmark. In *ISCA'20*. 446–459.
- [59] Scaphandre. Visited in 2025. <https://github.com/hubblo-org/scaphandre>.
- [60] Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. 2024. Agile-quant: Activation-guided quantization for faster inference of LLMs on the edge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18944–18951.
- [61] Xuan Shen, Zhenglun Kong, Changdi Yang, Zhaoyang Han, Lei Lu, Peiyan Dong, Cheng Lyu, Chih-hsiang Li, Xuehang Guo, Zhihao Shu, et al. 2024. Edgeqat: Entropy and distribution guided quantization-aware training for the acceleration of lightweight llms on the edge. *arXiv preprint arXiv:2402.10787* (2024).

- [62] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *ICML'23*. 31094–31116.
- [63] Jieke Shi, Zhou Yang, Hong Jin Kang, Bowen Xu, Junda He, and David Lo. 2024. Greening large language models of code. In *ICSE'24*. 142–153.
- [64] Shyam Sudhakaran, Miguel González-Duque, Matthias Freiberger, Claire Glanois, Elias Najarro, and Sebastian Risi. 2024. Mariogpt: Open-ended text2level generation through large language models. *Advances in Neural Information Processing Systems* 36 (2024).
- [65] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984* (2020).
- [66] Fuwen Tan, Royson Lee, Łukasz Dudziak, Shell Xu Hu, Sourav Bhattacharya, Timothy Hospedales, Georgios Tzimiropoulos, and Brais Martinez. 2024. Mobilequant: Mobile-friendly quantization for on-device language models. *arXiv preprint arXiv:2408.13933* (2024).
- [67] Chunlin Tian, Xinpeng Qin, and Li Li. 2024. Greenllm: Towards efficient large language model via energy-aware pruning. In *IWQoS'24*. 1–2.
- [68] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
- [69] TruthfulQA. Visited in 2024. <https://github.com/sylinrl/TruthfulQA>.
- [70] Arya Tschan, Arun Tejasvee Raghunath Rajan, Sachin Idgunji, Anirban Ghosh, Jeremy Holleman, Csaba Kiraly, Pawan Ambalkar, Ritika Borkar, Ramesh Chukka, Trevor Cockrell, et al. 2024. MLPerf Power: Benchmarking the Energy Efficiency of Machine Learning Systems from Microwatts to Megawatts for Sustainable AI. *arXiv preprint arXiv:2410.12032* (2024).
- [71] Tempest A. van Schaik and Brittany Pugh. 2024. A Field Guide to Automatic Evaluation of LLM-Generated Summaries. In *SIGIR '24*. ACM, 2832–2836.
- [72] Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748* (2024).
- [73] Jianyu Wei, Shijie Cao, Ting Cao, Lingxiao Ma, Lei Wang, Yanyong Zhang, and Mao Yang. 2024. T-mac: Cpu renaissance via table lookup for low-bit llm deployment on edge. *arXiv preprint arXiv:2407.00088* (2024).
- [74] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems* 35 (2022), 17402–17414.
- [75] Jochen Wulf and Juerg Meierhofer. 2024. Exploring the potential of large language models for automation in technical customer service. *arXiv preprint arXiv:2405.09161* (2024).
- [76] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *ICML'23*. 38087–38099.
- [77] Minrui Xu, Hongyang Du, Dusit Niyato, Jiawen Kang, Zehui Xiong, Shiwen Mao, Zhu Han, Abbas Jamalipour, Dong In Kim, Xuemin Shen, et al. 2024. Unleashing the power of edge-cloud generative ai in mobile networks: A survey of aigc services. *IEEE Communications Surveys & Tutorials* (2024).
- [78] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems* 35 (2022), 27168–27183.
- [79] Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. 2024. Llm as a system service on mobile devices. *arXiv preprint arXiv:2403.11805* (2024).
- [80] Zhongzhi Yu, Zheng Wang, Yuhan Li, Ruijie Gao, Xiaoya Zhou, Sreenidhi Reddy Bommu, Yang Zhao, and Yingyan Lin. 2024. Edge-llm: Enabling efficient large language model adaptation on edge devices via unified compression and adaptive layer voting. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*. 1–6.
- [81] Xingyu Yuan, He Li, Kaoru Ota, and Mianxiong Dong. 2024. Generative inference of large language models in edge computing: An energy efficient approach. In *IWCNC'24*. 244–249.
- [82] Mingjin Zhang, Jiannong Cao, Xiaoming Shen, and Zeyang Cui. 2024. EdgeShard: Efficient LLM Inference via Collaborative Edge Computing. *arXiv preprint arXiv:2405.14371* (2024).
- [83] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198* (2023).
- [84] Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Lunling Wang, Zhihang Yuan, Xiuhong Li, et al. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294* (2024).