논문 25-50-09-04 The Journal of Korean Institute of Communications and Information Sciences '25-09 Vol.50 No.09
https://doi.org/10.7840/kics.2025.50.9.1364

Analyzing Quantized Small Language Models for Efficient Edge Deployment

Sooyoung Jang◆, Seungho Yang*, Changbeom Choi°

ABSTRACT

Quantized small language models (SLMs) offer a promising approach for deploying advanced natural language process- ing (NLP) services on resource-constrained edge devices. However, an in-depth examination of how different quantization configurations influence accuracy and efficiency remains underexplored. This paper systematically evaluates 72 quantized variants of Llama 3.2 (1B and 3B parameters) and Qwen 2.5 (1.5B and 3B parameters) across 13 quantization configura- tions, ranging from q2_K to q6_K. We use the MMLU-Pro benchmark to measure the accuracy (including and excluding random guesses), inference time, resource utilization, and power consumption on an NVIDIA Jetson Orin Nano. Our findings reveal that low-bit quantized models often rely heavily on random guessing, with modest accuracy improvements observed when these are excluded. Furthermore, Qwen 2.5 models generally yield superior accuracy and lower latency than Llama 3.2, albeit with higher sensitivity to quantization, whereas Llama 3.2 exhibits more consistent performance across quantization configurations. CPU utilization remains low (approximately 1-4%), with GPU utilization peaking up to 90% and power consumption ranging from 9.2 W to 11.5 W. Variability across different domains (computer science, engineering, and math) underscores the importance of selecting the appropriate model family, parameter size, and quantization configuration for specific applications. We conclude by outlining future directions for improving on-device NLP, including mixed-precision quantization, hardware-specific optimizations, and broader assessments covering multilingual or multimodal tasks.

Key Words : Small Language Models, Edge AI Deployment, Quantization, Edge Computing, MMLU-Pro

Ⅰ. Introduction

The emergence of Transformer-based architectures[1] has transformed the field of natural language processing (NLP), giving rise to large language models (LLMs) that exhibit exceptional performance in tasks such as language comprehension and generation. Models like OpenAI o1, GPT-4o, Gemini 1.5 Pro, and Claude 3.5 Sonnet have showcased the ability to produce human-like text, sparking widespread academic interest and commercial applications. Simultaneously, the availability of open-source LLMs such as Meta's Llama has democratized access to advanced language models, fostering a growing ecosystem for their distribution, deployment, and utilization in various services and platforms, including Ollama, LangChain, vLLM, and Hugging Face.

The utilization of LLMs is hindered by substantial computational and memory requirements, which poses significant challenges for deployment on resource-constrained edge devices. To mitigate these challenges, the concept of small language models

※ This research was supported by the 2024 KNUDP(Korea National University Development Project) funded by the Ministry of Education(MOE, Korea) and National Research Foundation of Korea(NRF)
◆ First Author : Hanbat National University, Department of Computer Engineering, syjang@hanbat.ac.kr, 종신회원
° Corresponding Author: Hanbat National University, Department of Computer Engineering, cbchoi@hanbat.ac.kr, 정회원
* Hanbat National University, Department of Urban Engineering, sh.yang@hanbat.ac.kr
논문번호 : 202501-012-A-RN, Received January 6, 2025; Revised March 7, 2025; Accepted April 11, 2025

(SLMs) has emerged as a promising alternative for edge deployment. Based on the literature, SLMs are often defined as having fewer than one billion parameters[2] or less than 10 billion parameters[3]. These scaled-down models are more suitable for deployment on resource-constrained edge devices, as they exhibit reduced computational and memory footprints.

While existing research has examined various characteristics of LLMs, such as inference latency, memory usage, and accuracy[4], these studies primarily focus on high-performance server hardware, neglecting the distinct constraints of edge devices. Assessments of small language models specifically designed for edge deployment remain limited. This gap is crucial, as the successful integration of SLMs could unlock a diverse range of NLP applications on edge platforms, including ondevice assistants, context-aware IoT systems, and realtime translation services while maintaining acceptable performance levels.

Quantization methods[5,6] have demonstrated the potential to substantially reduce the memory requirements and computational complexity of language models, thereby facilitating the deployment of large-scale models on resource-constrained edge platforms like the NVIDIA Jetson devices. Previous studies have investigated the application of quantization techniques in general neural networks[7] and diverse NLP applications[8,9]. However, the intricate relationship between model scale, quantization approach, and performance on edge hardware remains an area that requires further exploration.

Building upon prior research on quantization, model evaluation, and SLMs, this study extends those insights to edge computing. Specifically, we provide a comprehensive assessment of quantized SLMs on resource-constrained hardware. We evaluate the performance and efficiency-measured in terms of inference time, resource utilization, and power consumption-of 72 quantized models. These models, which range from 1 billion to 3 billion parameters and incorporate various quantization configurations, are designed for deployment on devices such as the NVIDIA Jetson Orin Nano. Our contributions are threefold: (1) we assess the viability of deploying quantized SLMs on edge devices; (2) we investigate the trade-offs between model accuracy and efficiency on resource-constrained hardware; and (3) we provide practical insights for selecting models and quantization configurations to support efficient NLP service deployments on edge devices.

Ⅱ. Related Work

Small Language Models: Wang et al.[10] have provided a detailed survey of small language models, clarifying their definitions and exploring their use cases in resource-constrained environments. Their work underscores the potential of SLMs to deliver competitive performance with significantly reduced parameter counts, laying the groundwork for further exploration of these models in edge hardware applications. Our study builds upon these insights by specifically evaluating quantized SLMs on edge devices, extending the analysis to scenarios with tight resource budgets.

Quantization Techniques: Jacob et al.[7] demonstrated the feasibility of integer-arithmetic-only inference through quantization, enabling significant model size reduction while maintaining acceptable accuracy levels. This pioneering research has influenced subsequent advancements in quantization techniques. Building upon this foundation, Gholami et al.[5] provided a comprehensive survey of various quantization methods, exploring their applications, challenges, and effectiveness across a range of scenarios. Lee et al.[9] conducted an extensive evaluation of quantized, instruction-tuned LLMs, analyzing models with up to 405 billion parameters. Their research highlights the trade-offs between quantization methods and model performance at scale. However, their focus on server-grade hardware limits the applicability of their findings to edge deployments, where computational resources and memory are constrained.

Model Storage Formats: The development of efficient model storage formats, such as GGUF (GPT-Generated Unified Format)[11], has been instrumental in enabling the deployment of quantized models on devices with limited resources. GGUF supports diverse quantization configurations while offering enhanced metadata storage capabilities. This extensibility ensures that models can be seamlessly loaded and executed across multiple platforms, facilitating systematic experimentation and reproducibility. Our work leverages the GGUF format to manage a variety of quantization configurations in a unified manner.

Ⅲ. Methodology

3.1 Hardware and Software Setup

Hardware: All experiments were conducted on an NVIDIA Jetson Orin Nano, which features a 1024-core NVIDIA Ampere GPU with 32 Tensor Cores, a 6-core Arm Cortex-A78A CPU, and 8GB of LPDDR5 memory.

Operating System: Ubuntu 22.04 LTS (64-bit).

LLM Serving Framework: We used the Ollama framework to serve the quantized LLMs due to its support for various quantization methods and compatibility with edge devices. The Ollama provides pre-quantized model variants, ensuring efficient execution on the Jetson device.

3.2 Model and Configuration

The models selected for evaluation comprise Meta's Llama 3.2 (1B and 3B) and Alibaba' s Qwen 2.5(1.5B and 3B). These represent two modern families of open-source SLMs. We applied 13 distinct quantization configurations using the GGUF storage format to each model to reduce both memory and computational overhead on edge devices. The configurations, denoted as 'q2_K', 'q3_K_L', 'q3_K_M', 'q3_K_S', 'q4_0', 'q4_1', 'q4_K_M', 'q4_K_S', 'q5_0', 'q5_1', 'q5_K_M', 'q5_K_S', and 'q6_K', are explained as follows:

- **'q2' to 'q6' (Quantization Bit-Width):** The numeral denotes the bit-width used to represent each weight. For instance, 'q2' indicates a 2-bit precision, while 'q6' signifies a 6-bit precision. Generally, a higher bit-width allows for a more nuanced representation of the model's weights, which can better preserve accuracy but at the cost of increased resource usage.

- **'0' and '1' (Quantization Types):** The suffixes '0' and '1' indicate the quantization type as defined in frameworks like llama.cpp. In 'type-0' quantization, each weight is reconstructed as $w = dq$, where d is the block scale, and q is the quantized value. In 'type-1' quantization, the reconstruction uses $w = dq + m$, where m represents the block minimum. These types differ in whether an offset (minimum) is used for each quantization block.

- **'K' (Quantization Method):** The presence of a K indicates the use of K-Quantization, a technique that divides the model's weights into large blocks, which are further split into smaller subblocks. Each sub-block is assigned its scale and minimum value, which are then quantized to a limited number of bits based on the target bitwidth. This approach helps the model maintain its performance despite the reduced precision by capturing distribution differences within large weight blocks.

- **'L', 'M', and 'S' (Quantization Variants):** These letters denote variant schemes to balance accuracy and efficiency by adjusting the bit-widths used for different parts of the model. In the 'L' (Large) variant, critical tensors are quantized with a higher bit-width to preserve precision, while all other tensors use the target bit-width. The 'M' (Medium) variant employs an intermediate bit-width for the critical tensors-lower than that of the 'L' variant but higher than the base target-thus compromising performance and compression. The 'S' (Small) variant uniformly applies the base target bit-width across all tensors, maximizing quantization efficiency but potentially sacrificing some accuracy.

For clarity, Table 1 provides a summary of the quantization configuration nomenclature and their key characteristics.

We chose the above models and quantization settings (as provided by the Ollama framework) following an initial screening based on prior works[2,3] and

Table 1. Summary of Quantization Configurations

Configuration	Description
q2_K	2-bit K-quantization (ultra-low precision for maximum memory efficiency).
q3_K_L	3-bit K-quantization, **L**arge variant: critical layers at higher quantization to preserve accuracy.
q3_K_M	3-bit K-quantization, **M**edium variant: critical layers at intermediate precision (trade-off approach).
q3_K_S	3-bit K-quantization, **S**mall variant: uniform 3-bit quantization across all layers (maximum efficiency).
q4_0	4-bit quantization, type-0 (block scale only, $w = d \times q$).
q4_1	4-bit quantization, type-1 (block scale + offset, $w = d \times q + m$).
4_K_M	4-bit K-quantization, **M**edium variant: 4-bit base with some higher-quantization sub-blocks for key weights.
q4_K_S	4-bit K-quantization, **S**mall variant: uniform 4-bit for all weights.
q5_0	5-bit quantization, type-0.
q5_1	5-bit quantization, type-1.
q5_K_M	5-bit K-quantization, **M**edium variant: higher than 5-bit for critical parts.
q5_K_S	5-bit K-quantization, **S**mall variant: uniform 5-bit quantization.
q6_K	6-bit K-quantization (highest precision considered in this study under K-quantization).

pilot experiments on the Jetson Orin Nano (8GB memory). The selection of Llama 3.2 and Qwen 2.5 was motivated by their complementary architectural and training characteristics, ensuring our findings' generalizability and robustness. Llama 3.2, introduced by Meta in September 2024, includes lightweight models with 1B and 3B parameters and multimodal variants with 11B and 90B parameters. This study specifically utilizes the lightweight versions optimized for efficient deployment on edge devices. The architecture features enhancements such as Grouped-Query Attention (GQA), significantly improving inference scalability. Furthermore, instruction-tuned versions of Llama 3.2 have been aligned with human preferences using supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF)[12]. Qwen 2.5, developed by Alibaba and also released in September 2024, complements Llama 3.2 by providing an alternative architecture pre-trained on an extensive corpus comprising 18 trillion tokens. Qwen 2.5 excels particularly in knowledge-intensive and reasoning tasks. Its variants, fine-tuned for instruction following and structured outputs, utilize RLHF extensively to enhance alignment and model responsiveness[13]. Including these two distinct model families-differentiated by Western and Chinese origins, distinct pre-training datasets, and varying tokenization strategies-strengthens the general applicability of our study. The comprehensive combination of models and quantization strategies (2 model families × 2 model sizes × 13 quantization configurations = 72 variants) facilitated an in-depth evaluation of how quantization affects key performance metrics, including storage efficiency, inference speed, and output accuracy.

3.3 Benchmark Dataset

We used a subset of the MMLU-Pro benchmark[14] to evaluate model accuracy, ensuring consistent and challenging test conditions across different models and quantization configurations. Specifically, we selected three knowledge domains from MMLU-Pro: computer science (410 questions), engineering (969 questions), and mathematics (1351 questions). MMLU-Pro is a more challenging extension of the original MMLU(Massive Multitask Language Understanding) benchmark[15], featuring complex reasoning questions and an increased number of answer choices (up to 10) to reduce the chance of correct guesses by random chance. Each question in these subsets is multiple-choice with one correct answer and 9 distractors, making naive guessing only 10% likely to succeed.

3.4 Evaluation Protocol

For each model variant, we measured several metrics: accuracy, inference time, resource utilization, and power consumption. Accuracy on MMLU-Pro was computed as the fraction of questions answered correctly. We report accuracy under two conditions: including random guesses and excluding random guesses. This distinction is made because the evaluation script is designed such that if a model' s generated response does not contain a recognizable answer option, a random answer is selected on its behalf (simulating a guess). Given the benchmark' s format of up to 10 choices, a random guess has a 10% chance of being correct. Including these guesses in accuracy can inflate the score of very poorly performing models. Therefore, we also calculate accuracy, excluding those cases where we remove all questions where the model failed to produce an answer and a guess was inserted. The "excluding" accuracy reflects the model' s performance on the subset of questions it meaningfully attempted. By comparing both, we can gauge how much a model relies on guesswork. We also recorded the inference time per question (averaged over all queries in a domain), using Ollama' s timing logs on the Jetson. CPU and GPU utilization were monitored via system tools (with GPU utilization focusing on the Jetson' s integrated GPU), and power consumption was measured using the Jetson' s onboard power meter, capturing average power draw (in milliwatts, mW) during inference on each domain. These measurements allow us to evaluate efficiency in terms of both computational load and energy usage. All experiments were run in a controlled setting on the Jetson (performance mode enabled, consistent ambient conditions) to ensure fair comparison across models and configurations.

Ⅳ. Experimental Results

This section presents an in-depth analysis of the experimental results for the quantized Llama 3.2 and Qwen 2.5 models. We first discuss accuracy outcomes (with and without random guess adjustments) across the different configurations, then examine inference latency and subsequently analyze the relationship between model size and performance. Finally, we consider resource utilization patterns and provide qualitative insights into the practical usability of these models.

4.1 Accuracy Analysis

Tables 2 and 3 summarize the accuracy of Llama 3.2 and Qwen 2.5 models on the MMLU-Pro benchmark, both including and excluding random guesses. The highest accuracy in each category is shown in bold. Each value represents accuracy, while the numbers in parentheses indicate the number of questions answered correctly over the total number of questions attempted. For instance, in Table 1, Llama 3.2 1B q2_K achieves 0.115 (47/410) when including random guesses, indicating 47 out of 410 questions were correct (i.e., 11.5% accuracy). Excluding random guesses, the model' s accuracy is 0.125 (43/345), meaning 43 out of 345 questions were correct (12.5% accuracy). From these figures, we can infer that 65 questions were randomly guessed, of which 4 were answered correctly. The MMLU-Pro benchmark evaluates language models on multiple-choice questions, with an average of 9.188, 9.583, and 9.771 answer choices for computer science, engineering, and math, respectively. This corresponds to probabilities of 0.109, 0.104, and 0.102, respectively, for randomly selecting the correct answer in each domain. Conventional MMLU-Pro evaluations include random guesses when a model' s response is unextractable, but this paper examines results both with and without random guesses to offer deeper insights into model performance. Excluding random guesses provides a clearer picture of a model' s predictive capability, as random responses can skew accuracy metrics. Notably, even small quantized models achieve higher accuracy when random guesses are excluded, underscoring their non-trivial predictive abilities.

Tables 2 and 3 reveal significant performance differences among categories. Generally, the math category exhibits the best performance, followed by computer science and engineering. Moreover, the optimal quantization technique varies by category. For example, Qwen 2.5 3B q4_1 performs best in math and computer science, yet in engineering it is surpassed by q5_K_S, which shows 5.9% higher accuracy. Similarly, for Llama 3.2 3B, the best-performing quantizer differs across categories. These variations indicate that even within MMLU-Pro results, it is essential to evaluate model performance by the required

Table 2. Accuracy of Llama 3.2 models on MMLU-Pro benchmark including and excluding random guesses

Quantization	Llama 3.2 1 B Accuracy		Llama 3.2 3 B Accuracy	
	including random guesses	excluding random guesses	including random guesses	excluding random guesses
		Computer science		
q2_K	0.115 (47/410)	0.125 (43/345)	0.185 (76/410)	0.188 (76/405)
q3_K_L	0.210 (86/410)	0.215 (85/395)	0.344 (141/410)	0.357 (140/392)
q3_K_M	0.163 (67/410)	0.161 (64/398)	0.322 (132/410)	0.335 (130/388)
q3_K_S	0.161 (66/410)	0.160 (64/401)	0.285 (117/410)	0.294 (117/398)
q4_0	0.183 (75/410)	0.186 (74/398)	**0.390 (160/410)**	**0.398 (159/400)**
q4_1	0.202 (83/410)	0.207 (82/396)	0.349 (143/410)	0.359 (142/395)
q4_K_M	0.224 (92/410)	0.232 (90/388)	0.378 (155/410)	0.390 (154/395)
q4_K_S	0.217 (89/410)	0.221 (87/393)	0.354 (145/410)	0.360 (144/400)
q5_0	0.212 (87/410)	0.220 (87/396)	0.324 (133/410)	0.333 (133/400)
q5_1	**0.232 (95/410)**	**0.237 (95/400)**	0.368 (151/410)	0.377 (150/398)
q5_K_M	0.207 (85/410)	0.215 (85/395)	0.354 (145/410)	0.363 (143/394)
q5_K_S	0.205 (84/410)	0.209 (82/392)	0.359 (147/410)	0.369 (146/396)
q6_K	0.220 (90/410)	0.226 (88/389)	0.359 (147/410)	0.377 (147/390)
		Engineering		
q2_K	0.104 (101/969)	0.109 (66/608)	0.156 (151/969)	0.156 (149/953)
q3_K_L	0.155 (150/969)	0.161 (147/912)	0.228 (221/969)	0.229 (214/935)
q3_K_M	0.139 (135/969)	0.141 (132/933)	0.224 (217/969)	0.230 (212/923)
q3_K_S	0.134 (130/969)	0.134 (125/934)	0.182 (176/969)	0.186 (174/938)
q4_0	0.146 (141/969)	0.150 (137/914)	0.253 (245/969)	0.257 (243/945)
q4_1	0.146 (141/969)	0.152 (137/904)	0.241 (234/969)	0.246 (231/940)
q4_K_M	0.168 (163/969)	0.173 (159/921)	0.233 (226/969)	0.234 (223/952)
q4_K_S	0.169 (164/969)	0.171 (157/918)	0.237 (230/969)	0.242 (227/939)
q5_0	**0.182 (176/969)**	**0.188 (173/921)**	0.266 (258/969)	0.272 (254/933)
q5_1	0.147 (142/969)	0.153 (140/916)	0.255 (247/969)	0.259 (242/934)
q5_K_M	0.155 (150/969)	0.158 (143/904)	0.238 (231/969)	0.245 (229/934)
q5_K_S	0.151 (146/969)	0.151 (139/921)	0.255 (247/969)	0.258 (244/946)
q6_K	0.152 (147/969)	0.158 (142/900)	**0.283 (274/969)**	**0.288 (270/938)**
		Math		
q2_K	0.100 (135/1351)	0.100 (113/1133)	0.201 (271/1351)	0.207 (258/1246)
q3_K_L	0.195 (264/1351)	0.211 (240/1140)	0.324 (438/1351)	0.357 (414/1160)
q3_K_M	0.198 (267/1351)	0.205 (246/1198)	0.338 (457/1351)	0.376 (430/1144)
q3_K_S	0.145 (196/1351)	0.153 (184/1202)	0.286 (387/1351)	0.312 (368/1181)
q4_0	0.204 (275/1351)	0.212 (244/1151)	0.335 (452/1351)	0.360 (432/1199)
q4_1	0.212 (286/1351)	0.236 (256/1086)	0.355 (480/1351)	0.384 (465/1210)
q4_K_M	0.208 (281/1351)	0.220 (261/1189)	0.360 (487/1351)	0.394 (467/1184)
q4_K_S	0.211 (285/1351)	0.223 (261/1170)	0.338 (456/1351)	0.374 (438/1170)
q5_0	**0.238 (321/1351)**	0.246 (300/1218)	**0.366 (494/1351)**	0.396 (468/1182)
q5_1	0.226 (305/1351)	**0.247 (286/1157)**	0.331 (447/1351)	0.362 (425/1174)
q5_K_M	0.209 (283/1351)	0.226 (268/1187)	0.363 (490/1351)	0.398 (473/1187)
q5_K_S	0.226 (305/1351)	0.239 (282/1178)	0.361 (488/1351)	**0.402 (468/1165)**
q6_K	0.230 (311/1351)	0.247 (289/1170)	0.356 (481/1351)	0.386 (456/1181)

Table 3. Accuracy of Qwen 2.5 models on MMLU-Pro benchmark including and excluding random guesses

Quantization	Qwen 2.5 1.5B Accuracy		Qwen 2.5 3B Accuracy	
	including random guesses	excluding random guesses	including random guesses	excluding random guesses
Computer science				
q2_K	0.127 (52/410)	0.135 (49/362)	0.105 (43/410)	0.000 (0/3)
q3_K_L	0.244 (100/410)	0.246 (100/406)	0.188 (77/410)	0.193 (77/398)
q3_K_M	0.271 (111/410)	0.273 (109/399)	0.190 (78/410)	0.195 (77/395)
q3_K_S	0.205 (84/410)	0.205 (83/405)	0.215 (88/410)	0.215 (87/405)
q4_0	0.300 (123/410)	0.302 (123/407)	0.395 (162/410)	0.398 (162/407)
q4_1	0.312 (128/410)	0.313 (128/409)	**0.398 (163/410)**	**0.400 (163/408)**
q4_K_M	0.302 (124/410)	0.303 (124/409)	0.337 (138/410)	0.337 (138/409)
q4_K_S	0.293 (120/410)	0.292 (119/408)	0.339 (139/410)	0.340 (139/409)
q5_0	0.310 (127/410)	0.314 (127/405)	0.368 (151/410)	0.368 (151/410)
q5_1	0.322 (132/410)	0.323 (131/406)	0.380 (156/410)	0.384 (156/406)
q5_K_M	0.283 (116/410)	0.286 (116/406)	0.388 (159/410)	0.391 (159/407)
q5_K_S	**0.332 (136/410)**	**0.333 (135/406)**	0.346 (142/410)	0.346 (142/410)
q6_K	0.324 (133/410)	0.327 (133/407)	0.373 (153/410)	0.373 (153/410)
Engineering				
q2_K	0.111 (108/969)	0.112 (101/905)	0.094 (91/969)	0.000 (0/0)
q3_K_L	0.184 (178/969)	0.185 (178/961)	0.213 (206/969)	0.214 (206/964)
q3_K_M	0.186 (180/969)	0.188 (180/959)	0.201 (195/969)	0.202 (194/961)
q3_K_S	0.164 (159/969)	0.163 (155/952)	0.185 (179/969)	0.185 (176/951)
q4_0	0.200 (194/969)	0.201 (194/963)	0.254 (246/969)	0.258 (245/950)
q4_1	**0.227 (220/969)**	**0.227 (218/961)**	0.273 (265/969)	0.275 (265/964)
q4_K_M	0.187 (181/969)	0.188 (181/961)	0.271 (263/969)	0.275 (262/951)
q4_K_S	0.209 (203/969)	0.210 (203/965)	0.254 (246/969)	0.259 (246/950)
q5_0	0.205 (199/969)	0.206 (199/965)	0.293 (284/969)	0.294 (284/965)
q5_1	0.200 (194/969)	0.200 (194/968)	0.311 (301/969)	0.314 (300/956)
q5_K_M	0.224 (217/969)	0.222 (213/959)	0.311 (301/969)	0.312 (299/957)
q5_K_S	0.214 (207/969)	0.213 (204/959)	**0.332 (322/969)**	**0.334 (320/959)**
q6_K	0.203 (197/969)	0.204 (197/967)	0.308 (298/969)	0.310 (297/957)
Math				
q2_K	0.097 (131/1351)	0.097 (114/1174)	0.116 (157/1351)	0.000 (0/0)
q3_K_L	0.279 (377/1351)	0.283 (366/1295)	0.170 (230/1351)	0.174 (226/1301)
q3_K_M	0.281 (379/1351)	0.291 (369/1269)	0.155 (209/1351)	0.157 (186/1188)
q3_K_S	0.199 (269/1351)	0.205 (260/1268)	0.216 (292/1351)	0.217 (287/1323)
q4_0	0.315 (425/1351)	0.321 (416/1297)	0.453 (612/1351)	0.464 (601/1295)
q4_1	0.358 (484/1351)	0.366 (467/1277)	**0.491 (664/1351)**	**0.514 (656/1276)**
q4_K_M	0.329 (445/1351)	0.341 (438/1285)	0.340 (459/1351)	0.352 (452/1283)
q4_K_S	0.319 (431/1351)	0.325 (423/1300)	0.437 (591/1351)	0.453 (584/1288)
q5_0	**0.376 (508/1351)**	**0.381 (498/1308)**	0.362 (489/1351)	0.368 (486/1321)
q5_1	0.358 (483/1351)	0.368 (479/1302)	0.346 (467/1351)	0.364 (458/1257)
q5_K_M	0.370 (500/1351)	0.380 (490/1290)	0.385 (520/1351)	0.390 (517/1324)
q5_K_S	0.361 (488/1351)	0.376 (482/1283)	0.383 (518/1351)	0.389 (516/1328)
q6_K	0.368 (497/1351)	0.377 (491/1304)	0.397 (536/1351)	0.408 (531/1303)

category.

When comparing Tables 2 and 3, Qwen 2.5 outperforms Llama 3.2 according to the highest accuracy criteria. In the math category, Qwen 2.5 3B q4_1 achieves 51.4%, notably higher than Llama 3.2 3B q5_K_S at 40.2%. However, Qwen 2.5 exhibits considerable performance variability across quantization configurations, particularly at low bit configurations (‘q2_K’, ‘q3_K_S’, ‘q3_K_M’, ‘q3_K_L’). Notably, Qwen 2.5 3B q2_K failed to extract any answers in engineering and math, and only three were extractable in computer science-all of which were incorrect, as shown in Table 2.

4.2 Inference Time Analysis

Tables 4 to 7 present the inference times for Llama 3.2 1B, Llama 3.2 3B, Qwen 2.5 1.5B, and Qwen 2.5 3B models across various quantization configurations. Each reported value is the mean, with parentheses showing the standard deviation. The inference time varies substantially by MMLU-Pro category, typically increasing from computer science to engineering and then math. Many outliers emerge at lower bit configurations: in Qwen models, q2_K on both 1.5B and 3B is particularly affected, while in Llama models, q3_K_S, q3_K_M, and q3_K_L on the 3B model exhibit notable outliers.

Figures 1 and 2 provide a more granular view of inference times per question for selected quantization settings (q2_K, q3_K_S, q4_K_M). Fig. 1 displays Llama 3.2 1B and 3B results for computer science, while Fig. 2 shows Qwen 2.5 1.5B and 3B in the same domain. Each dot represents the inference time for a single question, and the red dashed line indicates the average. For Qwen 2.5 1.5B and 3B, q2_K incurs very high inference times (23.366 and 137.915, respectively), attributable to numerous values exceeding 100. As quantization configurations increase, inference times become more consistent.

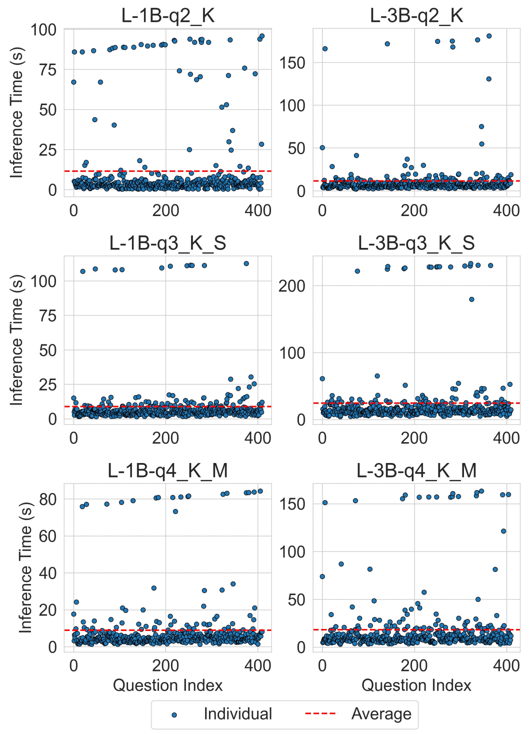

Fig. 1. Inference time distribution for computer science using Llama 3.2 (‘L’) models. ‘1B‘ and ‘3B’ refer to number of parameters, while ‘q2_K’, ‘q3_K_S’, and ‘q4_K_M’ specify the quantization methods.

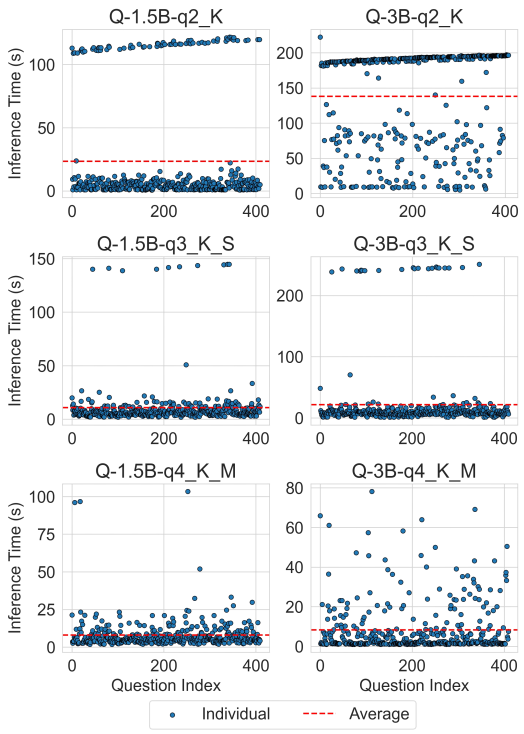

Fig. 2. Inference time distribution for computer science using Qwen 2.5 (‘Q’) models. ‘1.5B‘ and ‘3B’ refer to number of parameters, while ‘q2_K’, ‘q3_K_S’, and ‘q4_K_M’ specify the quantization methods.

Table 4. Inference time (s) of Llama 3.2 1B models across quantization configurations

Quantization	Computer science	Engineering	Math	Total
q2_K	11.462(23.551)	9.263(21.571)	10.538(22.432)	10.224(22.317)
q3_K_L	11.071(17.746)	17.564(21.967)	22.353(32.947)	18.959(27.776)
q3_K_M	9.895(16.827)	16.565(23.753)	22.497(33.928)	18.499(28.858)
q3_K_S	8.888(17.215)	12.539(21.681)	21.891(35.115)	16.619(29.159)
q4_0	9.354(14.871)	23.973(26.178)	20.747(28.456)	20.181(26.458)
q4_1	6.443(11.731)	15.107(20.641)	16.733(26.279)	14.611(22.934)
q4_K_M	8.963(15.910)	17.565(22.026)	18.672(26.841)	16.821(24.040)
q4_K_S	7.584(13.510)	15.039(19.683)	19.195(27.591)	15.976(23.615)
q5_0	7.972(11.936)	17.304(20.068)	17.335(25.609)	15.918(22.362)
q5_1	8.306(11.161)	17.671(19.107)	19.853(26.900)	17.344(22.843)
q5_K_M	8.397(12.939)	15.766(17.636)	17.657(25.445)	15.595(21.583)
q5_K_S	7.714(11.305)	15.763(18.159)	17.760(26.042)	15.542(21.989)
q6_K	8.813(14.254)	17.931(20.068)	19.285(27.803)	17.232(23.852)

Table 5. Inference time (s) of Llama 3.2 3B models across quantization configurations

Quantization	Computer science	Engineering	Math	Total
q2_K	11.374(23.145)	13.306(21.043)	30.060(55.294)	21.307(42.733)
q3_K_L	23.590(41.897)	42.906(55.409)	44.929(67.010)	41.007(60.249)
q3_K_M	20.084(35.182)	39.949(50.903)	45.832(68.397)	39.877(59.136)
q3_K_S	24.303(42.731)	41.209(57.042)	54.763(81.317)	45.378(69.411)
q4_0	14.563(24.193)	26.293(32.452)	30.409(43.701)	26.568(37.892)
q4_1	14.759(25.690)	27.733(34.375)	29.476(42.281)	26.647(37.800)
q4_K_M	18.207(30.425)	36.060(42.961)	35.125(50.195)	32.916(45.600)
q4_K_S	16.205(26.174)	35.918(41.811)	34.012(48.779)	32.014(44.111)
q5_0	19.761(32.906)	35.688(42.242)	34.912(49.827)	32.912(45.335)
q5_1	17.219(28.820)	33.025(41.639)	34.715(49.298)	31.487(44.490)
q5_K_M	21.813(36.426)	38.027(46.302)	35.727(49.826)	34.454(47.097)
q5_K_S	19.626(32.493)	34.875(43.101)	34.594(49.660)	32.446(45.469)
q6_K	22.257(37.381)	41.805(49.517)	40.804(57.350)	38.374(52.478)

Table 6. Inference time (s) of Qwen 2.5 1.5B models across quantization configurations

Quantization	Computer science	Engineering	Math	Total
q2_K	23.366(41.708)	20.377(38.791)	30.265(47.524)	25.720(43.980)
q3_K_L	10.696(16.927)	12.802(20.240)	21.131(29.700)	16.608(25.406)
q3_K_M	12.388(23.365)	13.187(22.379)	24.353(36.741)	18.593(30.989)
q3_K_S	10.879(21.321)	16.445(32.183)	21.860(40.491)	18.289(35.539)
q4_0	7.201(9.839)	8.789(10.291)	15.383(22.818)	11.814(17.960)
q4_1	6.231(6.174)	9.585(10.740)	18.462(22.360)	13.474(17.879)
q4_K_M	8.030(9.604)	10.405(12.910)	18.450(25.701)	14.029(20.485)
q4_K_S	7.079(8.855)	8.818(10.160)	17.114(22.846)	12.662(18.068)
q5_0	9.266(12.052)	11.518(11.455)	18.857(22.226)	14.812(18.150)
q5_1	7.888(10.743)	8.911(7.879)	16.121(20.971)	12.326(16.469)
q5_K_M	7.945(10.031)	9.510(8.849)	19.724(24.862)	14.330(19.431)
q5_K_S	8.220(10.942)	9.006(7.553)	18.944(24.531)	13.806(19.025)
q6_K	9.282(12.119)	10.741(7.607)	20.629(26.712)	15.415(20.556)

Table 7. Inference time (s) of Qwen 2.5 3B models across quantization configurations

Quantization	Computer science	Engineering	Math	Total
q2_K	137.915(71.527)	50.104(63.342)	12.707(12.517)	45.206(64.297)
q3_K_L	19.481(39.276)	29.959(38.045)	33.287(54.477)	30.036(47.289)
q3_K_M	18.699(38.964)	26.585(40.064)	29.027(51.892)	26.609(46.288)
q3_K_S	21.644(51.973)	21.542(38.120)	25.178(54.540)	23.357(48.957)
q4_0	16.034(15.441)	43.807(28.396)	38.215(28.779)	36.868(28.549)
q4_1	20.170(17.618)	44.744(26.949)	39.754(32.522)	38.584(29.881)
q4_K_M	8.202(12.565)	45.633(34.546)	11.927(20.739)	23.332(30.585)
q4_K_S	14.526(16.620)	45.047(31.467)	30.979(28.547)	33.502(30.013)
q5_0	14.962(17.719)	37.918(33.953)	17.002(24.819)	24.120(29.435)
q5_1	13.016(16.715)	40.227(29.497)	15.406(22.890)	23.857(27.535)
q5_K_M	17.617(17.876)	36.754(27.205)	19.859(26.641)	25.519(27.055)
q5_K_S	15.071(18.829)	30.236(28.956)	19.586(24.741)	22.688(26.220)
q6_K	20.242(22.045)	51.754(34.798)	21.533(28.985)	32.066(33.646)

4.3 Relationship Between Accuracy, Inference Time, and Model Size

Figure 3 illustrates the relationship between accuracy and inference time, while Figure 4 shows how accuracy varies with model size. Figure 5 depicts the relationship between inference time and model size. In this context, "accuracy" refers to the average performance across computer science, engineering, and math domains, including random guesses. In Figures 3 and 4, Qwen 2.5 3B q4_1 achieves the highest overall accuracy at 40%, surpassing Qwen 2.5 3B q5 and q6. For Llama 3.2 3B model, the q6_K configuration yields the highest accuracy at 33.04%. For Llama 3.2 3B, q3_K_S, q3_K_M, and q3_K_L configurations result in higher inference times while maintaining fair accuracy. In contrast, the same configurations applied to Qwen 2.5 3B lead to moderate inference times but lower accuracy. Additionally, Qwen 2.5 3B models

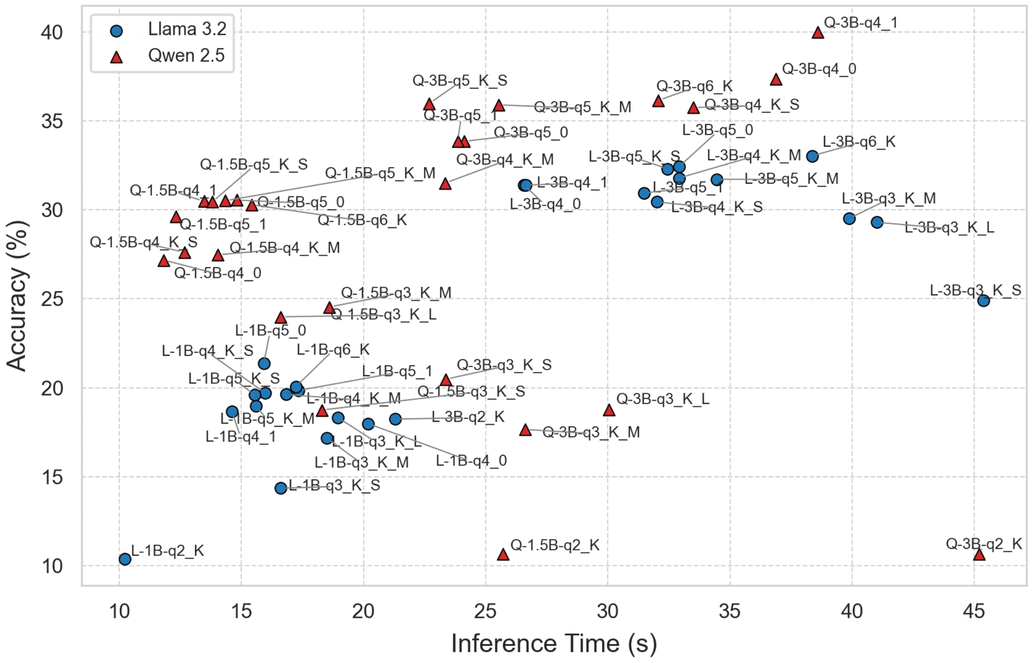

Fig. 3. Relationship between accuracy and inference time across all models and quantization configurations.

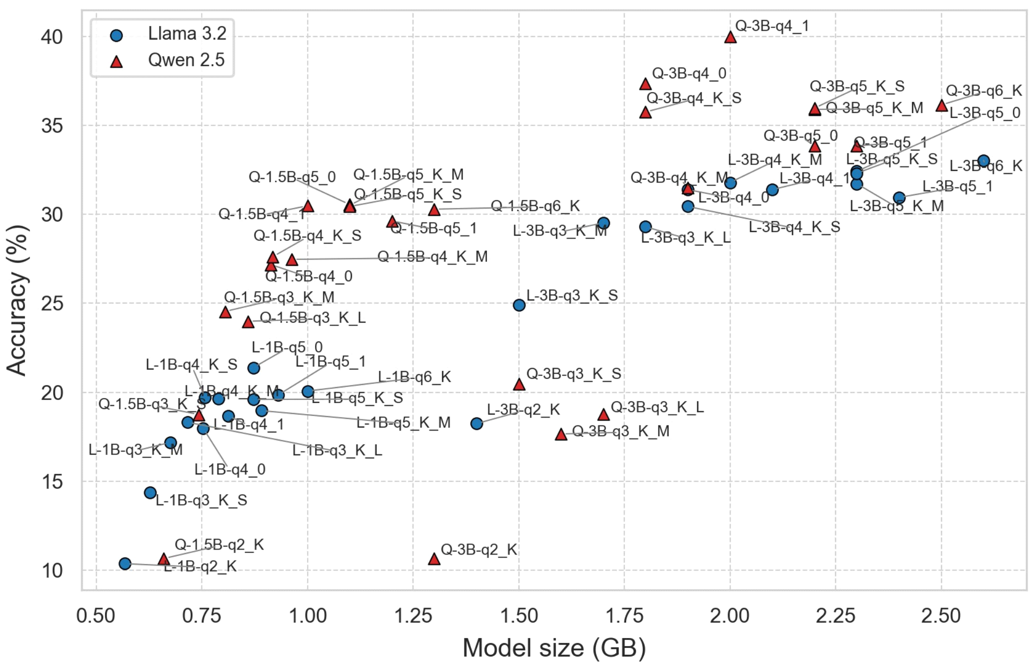

Fig. 4. Relationship between accuracy and model size across all models and quantization configurations.

using the q4_K_S, q4_0, and q4_1 configurations exhibit higher accuracy with longer inference times, whereas q2_K configuration produces very low accuracy with longer inference times. Comparing Qwen 2.5 1.5B and Llama 3.2 1B models, Qwen 2.5 1.5B generally shows lower inference times and higher accuracy across most quantization configurations-even though it has more parameters than Llama 3.2 1B. Similarly, Qwen 2.5 3B often outperforms Llama 3.2 3B in terms of both accuracy and inference speed.

We also revealed distinct clustering patterns as presented in Figures 4 and 5: Llama models are relatively well-clustered, whereas Qwen models exhibit great dispersion. This variability suggests that selecting the optimal quantization configuration for Qwen models is necessary and requires extensive experimentation. For example, even though Qwen 2.5 1.5B q6_K and Qwen 2.5 3B q2_K configurations have similar sizes, Qwen 2.5 1.5B q6_K significantly outperforms Qwen 2.5 3B q2_K in both accuracy and inference time.

4.4 Resource Utilization and Power Consumption Analysis

Deploying models on edge devices requires raw inference time and understanding of how they utilize hardware resources (CPU/GPU) and how much power they draw. During our tests, we monitored CPU load, GPU load, and power usage on the Jetson Orin Nano. Tables 8 to 11 present the resource utilization and power consumption for the Llama 3.2 and Qwen 2.5 models under each quantization configuration.

- Llama 3.2 1B Models: CPU utilization ranges from 1.726% to 3.587%, while GPU utilization peaks at 90.094%. Power consumption varies between 9825.426 mW and 11311.145 mW. Highbit configurations exhibit elevated power consumption despite reduced CPU load, likely due to increased GPU activity.

- Llama 3.2 3B Models: CPU utilization remains low (1.201%-2.398%), but GPU utilization fluctuates significantly, from 62.607% to 85.782%. Power consumption spans 9547.107 mW to 11458.084 mW. The q4_K_S configuration demonstrates the highest power draw, while q3_K_S balances lower GPU utilization with reduced energy consumption.

Table 8. Resource utilization and power consumption of Llama 3.2 1B models across quantization configurations

Quantization	CPU Utilization (%)	GPU Utilization (%)	Power Consumption (mW)
q2_K	2.852	83.706	9962.771
q3_K_L	2.564	85.131	10190.147
q3_K_M	2.658	82.492	10052.843
q3_K_S	2.428	83.708	9825.426
q4_0	3.587	79.850	10231.300
q4_1	3.543	78.055	10052.620
q4_K_M	1.874	90.094	11217.783
q4_K_S	3.460	79.364	10648.455
q5_0	3.553	79.697	10778.045
q5_1	3.557	80.015	10715.291
q5_K_M	3.309	79.040	10893.187
q5_K_S	3.372	80.730	10959.560
q6_K	1.726	89.405	11311.145

Table 9. Resource utilization and power consumption of Llama 3.2 3B models across quantization configurations

Quantization	CPU Utilization (%)	GPU Utilization (%)	Power Consumption (mW)
q2_K	1.637	79.311	10235.359
q3_K_L	1.282	76.252	9888.316
q3_K_M	1.374	85.782	10556.523
q3_K_S	1.559	76.015	9547.107
q4_0	2.146	78.249	11005.015
q4_1	2.398	67.443	10024.660
q4_K_M	2.106	69.784	10513.422
q4_K_S	1.840	81.614	11458.084
q5_0	1.201	73.463	10876.795
q5_1	2.032	68.757	10229.257
q5_K_M	1.909	67.834	10634.820
q5_K_S	1.952	69.566	10676.829
q6_K	1.788	62.607	10016.592

Table 10. Resource utilization and power consumption of Qwen 2.5 1.5B models across quantization configurations

Quantization	CPU Utilization (%)	GPU Utilization (%)	Power Consumption (mW)
q2_K	3.106	82.347	9784.272
q3_K_L	2.886	84.696	9842.946
q3_K_M	3.147	80.150	9625.500
q3_K_S	2.739	81.000	9447.782
q4_0	3.959	74.864	9835.472
q4_1	4.288	74.966	9576.185
q4_K_M	3.721	79.055	10432.955
q4_K_S	4.144	74.196	10163.166
q5_0	3.388	78.705	10457.940
q5_1	3.746	69.554	9570.465
q5_K_M	3.751	75.448	10432.320
q5_K_S	3.851	78.718	10331.470
q6_K	2.295	80.678	10515.761

Table 11. Resource utilization and power consumption of Qwen 2.5 3B models across quantization configurations

Quantization	CPU Utilization (%)	GPU Utilization (%)	Power Consumption (mW)
q2_K	3.383	84.773	10080.404
q3_K_L	1.372	77.756	9622.810
q3_K_M	1.844	78.777	9505.850
q3_K_S	2.478	78.393	9194.816
q4_0	2.561	69.440	9555.400
q4_1	2.569	74.382	9929.643
q4_K_M	2.739	68.165	9785.233
q4_K_S	2.733	80.807	10667.693
q5_0	2.163	71.740	10045.034
q5_1	2.773	71.190	10151.950
q5_K_M	2.440	64.571	9973.136
q5_K_S	2.554	63.123	9723.113
q6_K	2.029	72.238	10316.134

- Qwen 2.5 1.5B Models: CPU utilization ranges from 2.295% to 4.288%, with GPU utilization between 69.554% and 84.696%. Power consumption peaks at 10515.761 mW and drops to 9447.782 mW, indicating that lower-bit quantizations enhance energy efficiency.

- Qwen 2.5 3B Models: CPU utilization is modest (1.372%-3.383%), but GPU utilization varies widely, from 63.123% to 84.773%. Power consumption ranges from 9194.816 mW to 10667.693 mW. The q4_K_S configuration incurs the highest power cost, while q3_K_S achieves the lowest.

These findings underscore that one must align the quantization configuration with deployment priorities when optimizing for edge deployment. If the priority is energy efficiency, one might avoid configurations that cause long tail latencies or GPU thrashing (even if they save memory), because they can consume more power. If the priority is maximum accuracy, one might choose a higher bit-width and accept a moderate increase in power. Our analysis provides concrete data for these trade-offs. Such data can guide decisions depending on whether an application (say, a battery-powered IoT device vs. a plugged-in smart kiosk) values efficiency over accuracy or vice versa.

4.5 Qualitative Evaluation and Use Cases

While the above quantitative results shed light on performance metrics, it is equally important to consider qualitative aspects and the usability of these quantized models in real application scenarios. Ultimately, an edge-deployed language model must deliver useful responses to end-users within acceptable time frames. Here, we discuss some qualitative observations and the practical implications of our findings.

Firstly, the user-perceived quality of answers can degrade in non-obvious ways when models are heavily quantized. For configurations like q2_K or q3_K_S that showed very low accuracy, the content of their outputs often reflected confusion or failure. For example, in our trials, Qwen 2.5 3B with q2_K frequently produced irrelevant or malformed answers, or no answer at all, for many questions. This aligns with the quantitative result that it answered zero questions correctly. This model would be unusable in a practical setting such as an on-device engineering Q&A assistant a user would receive either incorrect answers or an “I don’ t know” far too often. In contrast, a model with slightly higher precision (say Qwen 2.5 3B q4_1) might produce mostly correct and fluent answers in the same scenario, resulting in a dramatically better user experience. This highlights that beyond accuracy percentages, there is a threshold of quality below which a model’ s outputs may be nonsensical or unhelpful to a human. Our results suggest that quantization beyond 4-bit for complex QA tasks crosses that

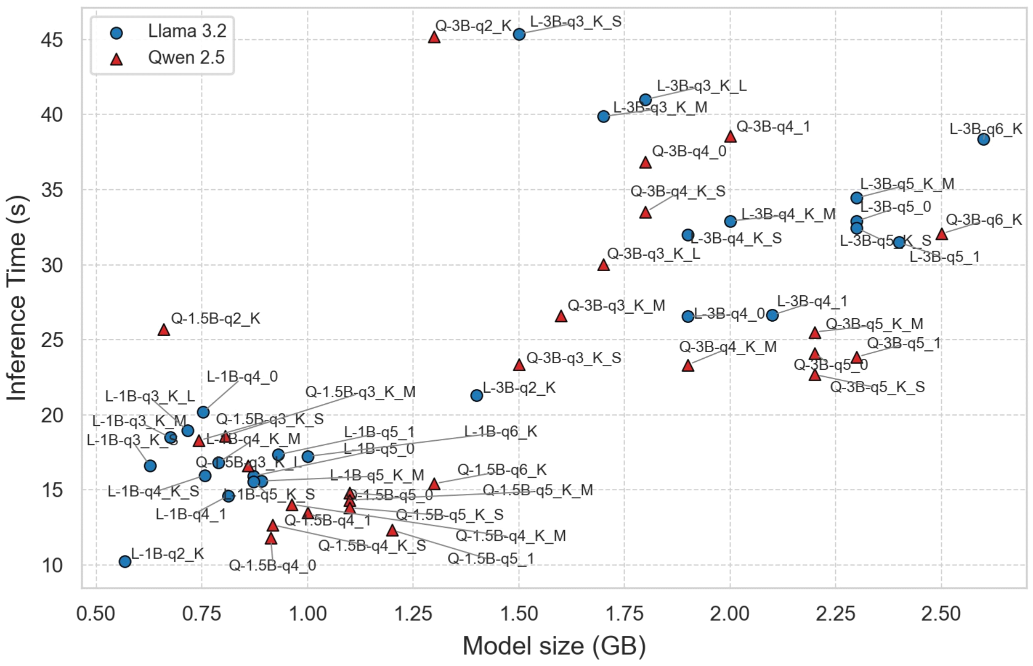

Fig. 5. Relationship between inference time and model size across all models and quantization configurations.

threshold for these models.

Secondly, consider application scenarios like ondevice virtual assistants, real-time translators, or voice-controlled IoT devices (as mentioned in the Introduction). Such applications require accuracy, real-time interaction constraints, and user satisfaction considerations. For instance, a real-time translator deployed on a mobile device must generate translations quickly and accurately. Suppose we were to deploy Llama 3.2 1B q3_K_S (3-bit, small variant) to such an application. In that case, a user might experience very fast responses (due to the small model and aggressive quantization), but the translations could be garbled or incorrect, rendering the service ineffective. On the other hand, using Llama 3.2 3B q5_0 (5-bit, type-0) might yield slower responses but much more reliable translations. User feedback in such a case would likely favor the slightly slower but more accurate model. This trade-off between speed and output quality must be tuned to the use case: for a casual chatbot that doesn't require high factual accuracy, a highly quantized small model might suffice, and the occasional nonsensical answer might be acceptable or even humorous. However, for a critical task (say, a medical information assistant on an edge device), one would choose the configuration that maximizes answer correctness, even if it means using more power or time.

We did not perform a formal user study, but qualitatively, we found that excluding random guesses from the evaluation (as discussed earlier) is a proxy for removing bogus answers. Users in a real scenario effectively “exclude” random guesses themselves by disregarding nonsensical responses. Our observation that excluding guesses modestly boosts measured accuracy suggests that some model outputs -especially from low-bit models-are essentially random. Thus, from a usability standpoint, those configurations have a high rate of unacceptable outputs. For practical deployment, one might implement safeguards such as confidencebased answer rejection. Indeed, one could design the system to detect when the model is likely guessing (e.g., low likelihood scores) and either prompt the user again or offload the query to a larger model in the cloud if possible. This difference could affect user trust; users might prefer a system that occasionally says “I'm not sure” (an understandable response) over one that returns gibberish. In summary, our practical evaluation suggests that there is a floor on quantization for real-world use beyond which the user experience degrades sharply. In our experiments, 4-bit quantization (especially with type-0/ 1 or K with some higher-precision variant) tended to be the lowest precision that still maintained reasonably coherent outputs across all domains. Configurations at 2-bit or 3-bit often led to outputs that would not meet user expectations in many applications. Therefore, developers targeting edge deployment should consider the qualitative behavior of models. Additionally, one can ensure better user satisfaction by aligning model choice with the application. These qualitative insights complement the quantitative metrics, reinforcing that the "best" configuration depends on the context of use, not just on raw performance numbers.

Ⅴ. Conclusions

This study presents the first systematic evaluation of quantized SLMs for deployment on resourceconstrained edge devices, focusing on the NVIDIA Jetson Orin Nano. By analyzing 72 quantized models of Llama 3.2 (1B and 3B parameters) and Qwen 2.5 (1.5B and 3B parameters) across 13 quantization configurations (q2_K to q6_K), we demonstrate the potential and limitations of these models for edge deployment. Our findings indicate that Qwen 2.5 models generally achieve higher accuracy and lower latency than Llama 3.2 models of similar size. Still, Qwen models are also more sensitive to aggressive quantization (suffering steep accuracy drops at low bit-widths). Excluding random guesses from evaluation improves the measured accuracy, while low-bit configurations (e.g., Qwen 2.5 3B q2_K) rely heavily on guessing due to the failure of answer extraction. We also observed that accuracy and inference time vary significantly across domains (computer science, engineering, and math), highlighting the importance of domain-specific model selection and quantization tuning. Resource utilization analysis shows that CPU utilization remains low (1-4%), while GPU was the main workhorse, reaching up to 90% utilization. Power consumption ranged from 9.2 W to 11.5 W under load. These results underscore the need to balance computational efficiency and energy consumption when deploying quantized models on edge devices.

Although our experiments were conducted on a specific hardware platform (Jetson Orin Nano), the insights are also relevant to other edge platforms. The qualitative trends we observed are likely to generalize across different hardware architectures. For edge devices that lack a GPU and rely on CPUs, quantization would remain crucial for fitting models in memory, though such devices may experience overall slower inference; the relative performance differences between, say, 4-bit and 6-bit models might be similar, but absolute latencies would increase. One could potentially deploy slightly larger models or use higher precision without exceeding resource limits on more capable edge accelerators or devices with greater memory (for example, a Jetson AGX Orin). Still, the same trade-off between efficiency and accuracy would apply. We expect that the optimal quantization strategy (e.g., choosing 4-bit vs. 8-bit) might shift with hardware – some processors are optimized for certain bit-widths- yet the core finding stands: moderate quantization offers big wins with tolerable impact on accuracy, whereas extreme quantization risks making the model ineffective. Our discussion on hardware generalizability is necessarily qualitative; verifying these hypotheses on diverse edge hardware (GPUs, CPUs, NPUs, DSPs across different vendors) is an important avenue for future work.

In addition to extending across hardware, future work should examine mixed-precision quantization strategies, where different layers of the model are quantized to different bit levels based on their sensitivity. Hardware-specific optimizations could further improve performance. Moreover, expanding evaluations to include multimodal and multilingual tasks will be valuable, especially since models like Llama 3.2 and Qwen 2.5 are inherently multilingual and some have vision or audio capabilities. We also suggest that future research incorporate user-centric evaluations (e.g., human feed-back on output quality, usability studies in actual edge applications) to complement quantitative metrics - addressing the gap between benchmark performance and real-world user satisfaction. By tackling these areas, the community can build on the findings of this work to enable more robust and energy-efficient NLP services on the next generation of resource-constrained edge devices.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is all you need," in *Advances in NeurIPS*, vol. 30, 2017.

[2] Z. Liu, C. Zhao, F. Iandola, et al., "MobileLLM: Optimizing sub-billion parameter language models for on-device use cases," in *Proc. ICML*, pp. 32431-32454, 2024.

[3] Y. Fu, H. Peng, L. Ou, A. Sabharwal, and T. Khot, "Specializing smaller language models towards multi-step reasoning," in *Proc. ICML*, pp. 10421-10430, 2023.

[4] C. Shin, Y. Go, Y. Yoo, G. Yang, and C. Yoo, "An analysis on inference time, accuracy, communication, and GPU memory usage for inference batch of large language models," *J. KICS*, vol. 49, no. 10, pp. 1377-1385, 2024.

[5] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv: 2103.13630*, 2021.

[6] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv: 2106.08295*, 2021.

[7] B. Jacob, S. Kligys, B. Chen, et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. CVPR*, pp. 2704-2713, 2018.

[8] J. Lang, Z. Guo, and S. Huang, "A comprehensive study on quantization techniques for large language models," *arXiv preprint arXiv:2411.02530*, 2024.

[9] J. Lee, S. Park, J. Kwon, J. Oh, and Y. Kwon, "A comprehensive evaluation of quantized instruction-tuned large language models: An experimental analysis up to 405B," *arXiv preprint arXiv:2409.11055*, 2024.

[10] F. Wang, Z. Zhang, X. Zhang, et al., "A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with LLMs, and trustworthiness," *arXiv preprint arXiv:2411.03350*, 2024.

[11] G. Gerganov, et al., *GGUF*, https://github.com/ggerganov/ggml/blob/master/docs/gguf.md, 2023.

[12] A. Grattafiori, A. Dubey, A. Jauhri, et al., "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783v3*, 2024.

[13] Qwen, : A. Yang, et al., "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115v2*, 2025.

[14] Y. Wang, X. Ma, G. Zhang, et al., "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark," in *Advances in NeurIPS*, vol. 37, pp. 95266-95290, 2024.

[15] D. Hendrycks, C. Burns, S. Basart, et al., "Measuring massive multitask language understanding," in *Proc. ICLR*, 2021.

Sooyoung Jang

Feb. 2006 : B.S. Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST).

Aug. 2008 : M.S. Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST). Feb. 2014 : Ph.D. Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST).

Mar. 2014~Sep. 2017 : Senior Researcher, amsung Electronics.

Oct. 2017~Feb. 2023 : Senior Researcher, Electronics and Telecommunications Research Institute (ETRI).

Mar. 2023~Current : Assistant Professor, Department of Computer Engineering, Hanbat National University.

<Research Interest> artificial intelligence, reinforcement learning, data analysis

[ORCID:0000-0002-6931-9592]

Seungho Yang

Feb. 2009 : B.S. Civil, Urban & Geosystem Engineering, Seoul National University.

Aug. 2014 : Ph.D. Civil & Environmental Engineering, Seoul National University.

Mar. 2015~Apr. 2017 : Team Manager, Han-a Urban Research Institute.

May 2017~Mar. 2018 : Researcher, Goyang Reseach Institute.

Jun. 2018~Jan. 2021 : Postdoctoral Fellow, York University, Canada.

Mar. 2021~Mar. 2025 : Assistant Professor, Department of Urban Engineering, Hanbat National University.

Apr. 2025~Current : Associate Professor, Department of Urban Engineering, Hanbat National University.

<Research Interest> urban planning, urban design, smart city

[ORCID:0000-0001-8371-599X]

Changbeom Choi

Feb. 2005 : B.S. Computer Engineering, Kyung Hee University.

Feb. 2007 : M.S. Computer Science, Korea Advanced Institute of Science and Technology (KAIST).

Aug. 2014 : Ph.D. Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST).

Sep. 2014~Feb. 2021 : Associate Professor, Department of Global Entrepreneurship and Information Communication Technology, Handong Global University.

Mar. 2021~Current : Associate Professor, Department of Computer Engineering, Hanbat National University.

<Research Interest> discrete event system, artificial intelligence, AI agent

[ORCID:0000-0002-4826-7949]