PALMBENCH: A COMPREHENSIVE BENCHMARK OF COMPRESSED LARGE LANGUAGE MODELS ON MOBILE PLATFORMS

Yilong Li[1], Jingyu Liu[1], Hao Zhang[1], M Badri Narayanan[1], Utkarsh Sharma[1], Shuai Zhang[2], Pan Hu[3], Yijing Zeng[1], Bangya Liu[1], Jayaram Raghuram[1], Suman Banerjee[1]

[1]University of Wisconsin – Madison, [2]Amazon Web Services AI, USA, [3]Uber, USA

ABSTRACT

Deploying large language models (LLMs) locally on mobile devices is advantageous in scenarios where transmitting data to remote cloud servers is either undesirable due to privacy concerns or impractical due to network connectivity. Recent advancements (MLC, 2023a; Gerganov, 2023) have facilitated the local deployment of LLMs. However, local deployment also presents challenges, particularly in balancing the (generative) quality, latency, and throughput within the hardware constraints of mobile devices. In this paper, we introduce our lightweight, *all-in-one automated benchmarking* framework that allows users to evaluate LLMs on mobile devices. We provide a comprehensive benchmark of various popular LLMs with different quantization configurations (both weights and activations), across multiple mobile platforms with varying hardware capabilities. Unlike traditional benchmarks that assess full-scale models on high-end GPU clusters, we focus on evaluating resource efficiency (memory and power consumption) and harmful output for compressed models on mobile devices. Our key observations include: **i)** differences in energy efficiency and throughput across mobile platforms; **ii)** the impact of quantization on memory usage, GPU execution time, and power consumption; and **iii)** accuracy and performance degradation of quantized models compared to their non-quantized counterparts; and **iv)** the frequency of hallucinations and toxic content generated by compressed LLMs on mobile devices.

1 INTRODUCTION

Large Language Models (LLMs) such as ChatGPT (OpenAI, 2023), Claude (Anthropic, 2023), and Llama (Touvron et al., 2023a;b;c) are powerful generative models that are revolutionizing interactive communication and various natural language processing tasks, including question-answering, document summarization, abstract reasoning, and code auto-completion (e.g., Github Copilot (Github)). LLMs require significant computational and memory resources to run due to their huge number of parameters (e.g., MT-NLG 530B (Smith et al., 2022)), making them more suitable for running on cloud infrastructures with high-end powerful GPU clusters. While significant attention has been dedicated to cloud-based LLMs, there is a growing need to run LLMs on resource-constrained mobile devices in order to obtain some key benefits. (1) *Privacy and Security*: Processing user data locally on mobile devices helps protect user privacy and enhances data security. There is also less risk of data breaches or unauthorized access to sensitive information. (2) *No Cloud Reliance*: By running LLMs locally, mobile applications can reduce their dependence on cloud services for language processing tasks. This can lead to cost savings and increased reliability, as the application's functionality is not reliant on the availability and performance of remote servers. (3) *Offline Access*: By running LLMs on mobile devices, users can access powerful language processing capabilities even when they are not connected (or have unreliable connection) to the Internet.

The rapidly flourishing LLM ecosystem, including various large models, architectures and frameworks, presents both opportunities and challenges for developers and researchers interested in deploying pre-trained LLMs on mobile devices. Existing efforts in on-device LLM inference have primarily focused on model compression and efficient inference techniques,with a strong focus on deploying models on edge SoCs (system-on-a-chip) equipped with GPUs and running Linux systems (Lin et al., 2024; Gerganov, 2023; MLC, 2023a;b), particularly MLC (MLC, 2023a), which is built on **TVM** (Chen et al., 2018), and **llama.cpp** (Gerganov, 2023). These approaches aim to achieve a desired quality, latency, and throughput while operating within the constraints of the target platform, *e.g.*, available memory and power limitations. However, the challenges and opportunities of efficiently deploying these large models on mobile platforms, such as smartphones or nano computers (*e.g.*, NVIDIA Jetson Orin Nano), remain largely unexplored.

Current LLM benchmarks primarily target the accuracy of large models on cloud clusters rather than mobile devices (Zheng et al., 2023). Some papers on mobile devices typically examines a limited range of models and platforms, often overlooking performance degradation and hallucination due to quantization (Çöplü et al., 2023; Laskaridis et al., 2024).

To bridge this gap, we propose a comprehensive benchmarking framework to evaluate the overall user experience of LLMs on mobile devices. This framework automatically tests various LLMs with different compiler options (e.g., weight and activation quantization) across mobile platforms with diverse hardware capabilities (see Table 1). We systematically evaluate each LLM using a range of metrics across efficiency, accuracy (generative quality relative to non-quantized models), and harmful output dimensions. Our primary focus is on inference efficiency on mobile platforms, evaluating the computation usage (CPU and GPU), latency, throughput, energy consumption, and memory footprint of different models on different platforms (as shown in Figure 1). Along the accuracy dimension, we evaluate quantized models on mobile devices using mainstream datasets like Natural Questions (Kwiatkowski et al., 2019) and SQuAD (Rajpurkar et al., 2016), employing exact match and perfect match metrics to quantify performance degradation and ensure proper functionality for basic use cases. Also, we evaluate the harmful outputs including hallucination and toxicity of LLMs with existing benchmark datasets Li et al. (2023b); Lin et al. (2022); Luong et al. (2024).

LLMs	LLaMa-2 (Touvron et al., 2023b), LLaMa-3/3.1 (Dubey et al., 2024), RedPajama (Computer, 2023), LLaMa-3.2 (Llama, 2024), Vicuna (Chiang et al., 2023), TinyLlama (Zhang et al., 2024), Qwen2 (Bai et al., 2023), Mistral-7B (Jiang et al., 2023), Gemma (Team, 2024)
Mobile Platforms	Google Pixel 4 / Pixel 5a / Pixel 7, iPhone 15 Pro, iPhone 12 Pro, S22 Ultra, Orange Pi 5 (Pi), Nvidia Jetson Nano (Nano)
Quantization	2-bit (MLC, llama.cpp), 4-bit (MLC, llama.cpp), 5-bit (llama.cpp), 6-bit (llama.cpp), 8-bit (MLC, llama.cpp) (Frantar et al., 2022; Lin et al., 2024; Li et al., 2020; Dettmers & Zettlemoyer, 2023)

Table 1: Summary of LLMs, mobile platforms, and quantization configurations explored by our benchmark.

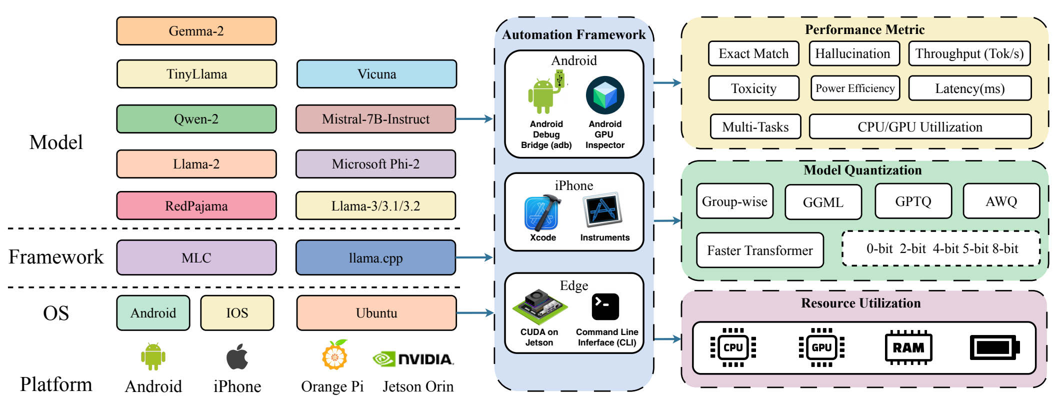


Figure 1: Overview and workflow of `PalmBench` – our evaluation and benchmarking framework for Large Language Models (LLMs) on mobile devices.

Unlike existing benchmarking efforts for LLM deployment on mobile devices, our study explores and evaluates feasible pre-trained models with the most popular quantizations. We highlight various combinations of LLM configurations suitable for mobile deployment (Zhang et al., 2024; Computer, 2023), along with several available quantization options (Lin et al., 2024; Frantar et al., 2022; MLC, 2023a; Dettmers & Zettlemoyer, 2023).

We found that 4-bit quantization methods, such as group-wise (Yang et al., 2024), GPTQ (Frantar et al., 2022), and AWQ (Lin et al., 2024), can generally preserve the performance of LLMs while reducing their size to one-quarter of the original non-quantized model. This configuration diversity can potentially lead developers (and users) to make sub-optimal choices in terms of performance and efficiency. Our benchmarking analysis, focused on resource usage during inference, provides insights into efficient deployment strategies tailored for mobile platforms. These strategies include joint considerations of 1) model architecture, 2) quantization strategy, and 3) model size. In summary, the major contributions of our paper are:

- To enable a comprehensive evaluation of various LLMs, we first develop a *lightweight, automated benchmarking framework* that collects performance metrics from mobile devices via USB, eliminating the need for additional equipment.
- We evaluate various quantized LLMs on mobile platforms with different hardware capabilities, measuring their resource utilization, power consumption, throughput, and inference latency.
- We validate the *knowledge and answering* accuracy of quantized models compared to their non-quantized counterparts, and investigate the potential issues of compressed models such as toxicity, bias, and the generation of erroneous or random outputs (hallucinations).
- Finally, our benchmarking leads to several key observations, highlighting the quantization differences across models, platforms, and frameworks. We also observed that the iOS platform outperforms others in power efficiency, latency, and throughput for LLM inference [1].

2 Related Work

Our benchmarking study extensively evaluated prior efforts focused on optimizing LLMs for mobile devices. These efforts include frameworks (MLC, 2023a;b; Gerganov, 2023), the development of smaller models (Computer, 2023; Abdin et al., 2024; Li et al., 2023c), and model quantization techniques (Frantar et al., 2022).

Large Language Models. LLMs like ChatGPT (OpenAI, 2023), the Llama series (Touvron et al., 2023a;c), Mistral (Jiang et al., 2023), Vicuna (Chiang et al., 2023) *etc.* are gaining substantial influence in generative AI applications. While LLMs are undoubtedly driving AI advancements, their sophisticated capabilities demand significant resources. Both power consumption and memory requirements for training these models and generating predictions scale linearly with their size (number of parameters), significantly increasing operational costs during inference.

The development of smaller models such as Google Gemma-2-2B (Team, 2024), Llama-3.2-1B/3B (Llama, 2024), RedPajama-INCITE-3B (Computer, 2023), Phi-2/Phi-3 (Abdin et al., 2024), and TinyLlama (Zhang et al., 2024) is providing enhanced options for resource-constrained mobile devices or edge devices that require efficient

Quantization. Post-training quantization (PTQ) applies quantization to LLMs after they have been fully trained, which is an effective compression method used to create smaller models for inference. Many efforts have been made to optimize LLMs for more efficient storage and faster computation. Group-wise quantization (Yang et al., 2024) involves partitioning the weights of a neural network into groups and quantizing each group independently. This allows the quantization process to be more finely tuned to the distribution of weights within each group, specifically reducing I/O costs and offloading on mobile platforms. GPTQ (Frantar et al., 2022) goes one step further and proposes a post-training weight quantization method that compresses LLM weights to 3 or 4 bits instead of 8 bits. Activation-aware Weight Quantization (AWQ) (Lin et al., 2024) observe that there exists a small subset of model weights called *salient weights*, characterized by larger activation magnitudes, plays a crucial role in reducing the quantization loss of LLMs, if they are preserved with high precision.

Inference Engine. Although there are a lot of efficient inference frameworks, MLC-LLM (Machine Learning Compilation) (MLC, 2023a;b) enables users to develop, quantize, and deploy LLMs across various platforms, including mobile devices and web browsers. It leverages compiler accelerations

[1] We plan to release the code of our framework to facilitate reproducibility and extensions of our research.

and runtime optimizations for native deployment across platforms based on TVM (Chen et al., 2018). LlaMa.cpp (Gerganov, 2023), developed in C++, offers a lighter and more portable alternative to traditional Python-based frameworks. It supports multiple BLAS backends for fast processing and employs a mixed set of quantizations, primarily focusing on K-Quants.

Metric	Definition
`CPU Utilization` (%)	Percentage of the total processor cycles consumed by LLM
`GPU Utilization` (%)	Percentage of the total GPU computing resource during LLM inference
`Memory Footprint` (GB)	Measurement of main memory used by the LLM application
`Memory Utilization` (%)	Percentage of main memory used by the LLM application
`Throughput` (Tok / s)	Number of output tokens per second generated by the LLM
`Output Matching`	Accuracy degradation of the compressed model relative to the original model
`Toxicity`	Toxicity score calculated on 25k sentences by Perspective API
`Hallucination` (%)	Percentage of erroneous or random outputs not related to the questions

Table 2: Metrics for evaluating the performance of LLMs on mobile devices. Memory usage includes both the model loaded to the memory and the framework program running on devices.

Benchmark. Most existing benchmark frameworks of LLMs focus on maximizing performance across different model architectures and evaluating a model's general world knowledge, question-answering, and reasoning ability (Zheng et al., 2023; Hendrycks et al., 2021; Rajpurkar et al., 2016; Kwiatkowski et al., 2019). Some existing studies on evaluating LLMs for mobile deployment are limited in scope, focusing either on a single platform (e.g., Çöplü et al. (2023) assesses various models on iPhone) or evaluating online models instead of on-device inference (Lee et al., 2024). They fail to comprehensively examine resource efficiency and power consumption, which are crucial for mobile deployment. MELTing (Laskaridis et al., 2024) is the closest to our work, evaluating five LLMs on various devices and reporting throughput, power consumption, and basic accuracy on Q&A datasets. However, MELTing does not comprehensively analyze resource utilization and energy efficiency for popular quantization methods, nor does it explore the impact on GPU workload. Furthermore, existing benchmark works ignore the toxicity, bias, and generation of erroneous or randomized outputs (which always occur in quantized models). However, they are crucial factors affecting user experience and are affected by different frameworks.

3 Methodology

To evaluate LLMs on mobile devices, we created the `PalmBench` framework, which focuses on the following three aspects:

1) Benchmark Automation. We developed an automated framework that uses Android Graphic Inspector (AGI) (AGI), Xcode profiler, and Nvidia Visual profilers to trace execution data and analyze runtime behavior across platforms when benchmarking LLM performance and resource utilization on edge and mobile devices. `PalmBench` requires USB debugging to connect to Android phones and iPhones.

2) Resource Utilization. Our primary focus is on the resource demands of different models across various platforms—such as CPU, GPU, memory, and NPU — that significantly impact user experience. Our study goes beyond resource demands, aiming to quantify the impact of different quantization techniques on both performance and resource efficiency across various state-of-the-art models.

3) Model Accuracy. While a model's architecture mostly dominates its outputs, we observe variations when applying different quantization methods on diverse devices. To validate the quantized LLMs and quantify the accuracy degradation due to compression, we evaluate the ***knowledge and answering*** accuracy of models using the *Exact Match* and *F1 score* compared with original models. We also test these models on conventional tasks with open-sourced datasets (Rajpurkar et al., 2016;

Kwiatkowski et al., 2019; Hendrycks et al., 2021). Moreover, we also investigate the potential issues of toxicity and the generation of erroneous or randomized outputs that always occur in compressed models and have not been thoroughly studied in previous work (Laskaridis et al., 2024).

3.1 Metrics and Datasets

Table 2 summarizes all the metrics used in our benchmark. To evaluate the accuracy and correctness of quantized LLMs, we use popular *Question-Answering* datasets such as SQuAD (Rajpurkar et al., 2016) and Natural Questions (NQ) (Kwiatkowski et al., 2019), comparing their performance with that of the original, non-quantized models. Additionally, we employ comprehensive benchmarks for different tasks, including MTBench (Zheng et al., 2023) to compare their language understanding and reasoning capabilities across different quantizations. Also, we measured *toxicity* by calculating **toxic score** by using Perspective API (Perspective, 2020) and TET (Luong et al., 2024), and evaluate *hallucination* in each quantized LLM using HaluEval (Li et al., 2023a) and TruthfulQA (Lin et al., 2022) benchmarks. Appendix D and Appendix F provide some examples of these datasets for benchmarking. These widely-recognized datasets ensure that our experiments and metrics are both convincing and reproducible.

3.2 Choice of LLMs

We have identified and converted several popular models for benchmarking on edge and mobile devices using model weights from their official Huggingface or GitHub repositories. These models are converted into experimental formats such as GGUF and K-quant for Llama.cpp Gerganov (2023) or compiled using TVM Chen et al. (2018) for MLC frameworks MLC (2023a;b). Given that mobile devices typically do not exceed 8GB of memory, it is impractical to test too large models, as they would surpass these devices' memory capacity. In our benchmark, we evaluated the various LLMs, including Llama-2-7b-chat Touvron et al. (2023b), Llama-3-8B-Instruct Touvron et al. (2023c) Microsoft Phi2 Abdin et al. (2024), Mistral-7B-Instruct Jiang et al. (2023), RedPajama-INCITE-Chat-7B Computer (2023), Vicuna Chiang et al. (2023)), TinyLlama-1.1B-Chat-v1.0 Zhang et al. (2024), and Qwen2 Bai et al. (2023). The prebuilt weights for these models are readily available in the MLC repository, which also offers options for compilation in various configurations.

3.3 Infrastructure

Our benchmarking framework's infrastructure includes essential components that automate device interactions, model execution, data collection, and performance evaluation. It features both software and hardware elements such as devices, system drivers, performance profilers, automated data collection tools, and mobile phone user interfaces (UIs), representing a significant engineering effort, as shown in Appendix A Table 6.

3.3.1 Mobile Devices

We evaluate the LLMs on a range of devices with varying hardware capabilities, as listed in Table 6 in Appendix, including Google Pixel 4 (**P4**), Pixel 5a (**P5**), Pixel 7 (**P7**), iPhone 12 Pro (**IP12**), iPhone 15 Pro (**IP15**), S22 Ultra (**S22U**), Orange Pi 5 (**OP5**) Pi, and Nvidia Jetson Orin Nano (**Nano**) Nano, covering mainstream operating systems.

3.3.2 Inference Engine

We use two frameworks, MLC-LLM (MLC, 2023a;b) and llama.cpp (Gerganov, 2023), as inference engines to execute LLMs on devices. Although many frameworks claim compatibility with mobile devices, they often lack support for popular platforms or models. MLC-LLM (MLC, 2023a;b) and llama.cpp (Gerganov, 2023) are two of the most popular frameworks that support a wide range of platforms and models. Unfortunately, llama.cpp (Gerganov, 2023) is still incompatible with iPhone.

3.3.3 Quantization

Model quantization is primarily handled by the built-in quantization programs of frameworks. MLC supports various quantization levels, including non-quantized float-16 (q0f16) and float-32 (q0f32), 3-bit quantization (q3f16_1), 4-bit quantization (q4f16_1), and 4-bit AWQ (q4f16_awq). The format qAfB(_id) denotes 'A' as the number of bits for weight storage and 'B' as the number of bits for activation storage. llama.cpp supports quantization using its GGUF format, which employs a type of group-wise quantization known as K-quant and supports more quantization methods (1.5-bit, 2-bit, 3-bit, 4-bit, 5-bit, 6-bit).

3.3.4 PROMPT INPUT

For platforms like Orange Pi 5 Pi and Nvidia Jetson Nano Nano using Ubuntu, we can only use the Command Line Interface (CLI). Benchmark scripts are running, and prompt texts are transferred via USB serial ports. On mobile platforms, such as iPhones and Android devices, we have used custom-developed Apps based on MLC's examples MLC (2023a) that automatically fetch prompts from text files and initiate the touch events to interact with the Apps. Prompt texts are from datasets listed in 3.1.

3.3.5 CONTROL AUTOMATION

Control operations are predominantly conducted through the Android Debug Bridge (ADB) for Android devices. Due to the limitations of the built-in Android Studio profilers in monitoring GPU usage, we have employed the Android GPU Inspector (AGI) tool AGI, developed by the Android team, to track resource utilization metrics such as CPU, GPU, memory, energy, and latency. Raw data traces are extracted from AGI and transferred via ADB. iOS primarily used Xcode's profiling tools and a custom plugin to measure GPU utilization. The GPU measurement plugin, a derivative development based on the IOKit plugin Tan (2018), displays real-time GPU utilization within the app. Xcode offers a comprehensive performance analysis tool, ***Instruments***, which measures CPU utilization, memory usage, execution time, and energy consumption.

3.3.6 GPU DRIVER

Although MLC-LLM MLC (2023a;b) and llama.cpp Gerganov (2023) support various drivers; OpenCL is the preferred and most mature GPU driver commonly used for both Android phones and Ubuntu-based edge computing devices. The iPhone utilizes Apple's proprietary Metal driver, which is well supported by both MLC MLC (2023a) and TVM Chen et al. (2018). Nvidia Jetson Nano device leverages its own CUDA with highly optimized driver Nano.

3.3.7 EQUIPMENTS

In addition to resource usage, we also look into energy efficiency, a critical factor impacting user experience. We employ two complementary devices to comprehensively evaluate the effects of quantization on power consumption and the distribution of device temperature. For thermal behavior analysis, we utilize the FLIR C5 thermal imaging camera Flir (2020) and a professional USB power meter for accurate power consumption measurements. This enables us to investigate the energy efficiency and thermal behavior of mobile platforms across different models and quantization methods, which are crucial factors affecting user experience.

4 EXPERIMENTS

We present the most significant benchmarking results for LLMs across various models and platforms (outlined in Section 3) here, and provide additional results in the Appendix.

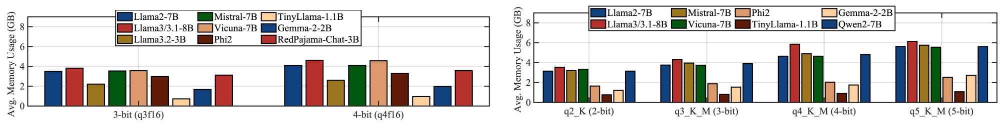
4.1 EXPERIMENTAL SETUP

We evaluate the LLMs on various devices detailed in Section 3. MLC (MLC, 2023a) is compatible with all platforms (Apple, Android, and Linux Edge), whereas llama.cpp Gerganov (2023) is limited to deployment on Android and Ubuntu systems. Since our benchmarking focuses on mobile devices, we concentrate on models that function correctly and are suitable for size. For example, the 0-bit Llama-2-7b-hf model, which occupies 13.11GB, is impractical for existing mobile devices. Some models come in different scales, such as the Vicuna 13B model and Vicuna-7B. However, even with 3-bit quantization, the Vicuna 13B model's memory usage exceeds 6GB, making it too large for mobile devices. Quantization methods primarily rely on built-in framework configurations. For a fair comparison, we standardize all models with a temperature setting of 0.2—the default value—and limit the context window to 4096.

4.2 RESOURCE UTILIZATION

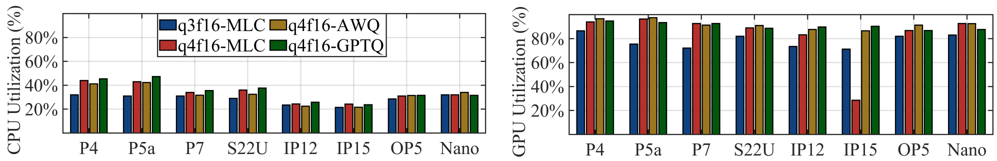
We first evaluate the impact of quantization on resource efficiency using the MLC and llama.cpp frameworks on Android phones and edge devices for the models detailed in previous sections.

Memory Utilization: LLM inference is inherently memory-bound, and its memory utilization can be reduced significantly by *memory bandwidth quantization*, which reduces the precision of weights and activations.



(a) MLC (b) llama.cpp

Figure 2: Average memory usage (GB) while running MLC and llama.cpp.



(a) CPU (b) GPU

Figure 3: CPU and GPU usage during inference of RedPajama-INCITE-3B across different quantizations.

To keep the models in suitable size with iPhones (iPhone 12 Pro and iPhone 15 Pro) and Android phones (Pixel 4/5a/7, Samsung S22 Ultra), we are only able to evaluate the *total memory usage* of models quantized by 3-bit and 4-bit in the MLC framework. Figures 2(a) and 2(b) illustrate the average memory usage of various models when using MLC and llama.cpp across different platforms. We observed that for a given model and framework (MLC or llama.cpp), each platform's total memory utilization remains consistent. Based on this observation, we compare the memory usage with different platforms (iPhone, Android, and Edge). The results reveal that higher quantization levels invariably consume more memory, while lower quantization than 4-bit significantly reduces memory needs. There is a slight difference in the MLC framework's total memory usage; it is lower on iPhones and higher on Android devices. Additionally, the Jetson Orin Nano uses less memory than the Orange Pi, reflecting CUDA's greater efficiency compared to OpenCL, as shown in Figure 6. The memory usage of MLC and llama.cpp varies across platforms, with llama.cpp typically uses less memory than MLC. This difference is likely due to llama.cpp's C++ implementation and command line interface (CLI) save resources, maintaining similar memory usage on both Android and Edge devices.

CPU and GPU Utilization: LLM execution depends on the computational resources of mobile platforms, with CPUs handling data transfer and model offloading between memory and GPUs, which are primarily used for inference. Both MLC and llama.cpp supports GPU-based model inference. We measured CPU and GPU activities to examine the impact of quantization on reducing memory traffic and GPU workload, with findings illustrated in Figure 3. The results show CPU and GPU utilization varies across different models and platforms. Notably, 3-bit quantization results in lower CPU and GPU usage, likely due to decreased data transfers from memory and a reduced inference workload on the GPU. Additionally, the iPhones exhibited lower resource utilization than other test platforms, indicating the potential for optimization and efficiency of LLM deployment. We also gathered GPU traces through automated benchmarking tools and charted the GPU utilization timeline to examine how GPU workloads vary when running identical models with different quantization methods, as depicted in Figure 4. The findings reveal that models with 4-bit quantization utilize more GPU duty

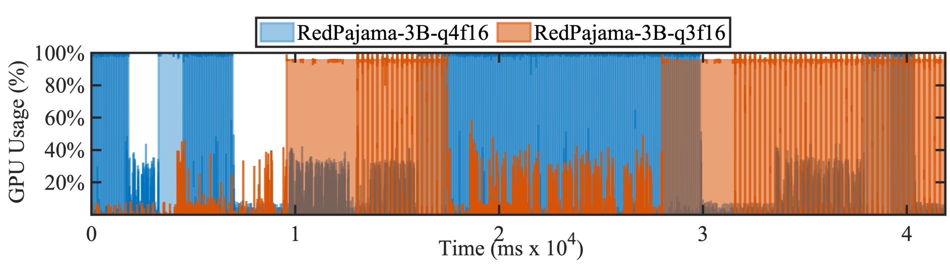
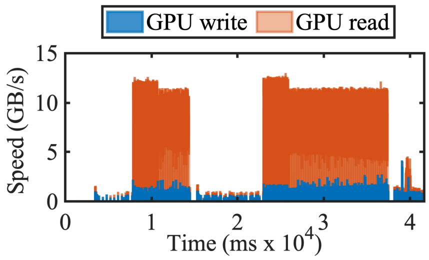
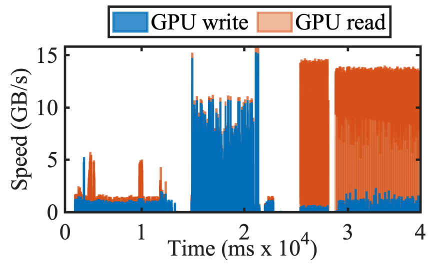


Figure 4: GPU Utilization (%) timeline for 3-bit and 4-bit quantized RedPajama models on Google Pixel 7.

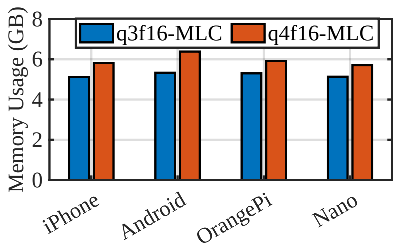


(a) Llama-3-8B-q3f16

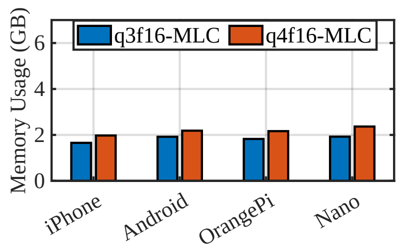


(b) Llama-3-8B-q4f16

Figure 5: GPU memory read/write speed while running LLaMa-3-8B-Instruct in 3-bit and 4-bit quantization on Pixel 7.



(a) Llama-3-8B for all platforms



(b) Gemma-2B for all platforms

Figure 6: Measured memory usage (GB) across different platforms using Llama-3-8B and Gemma-2-2B by MLC LLM to compare the memory usage between large model (Llama-3-8B) and small model (Gemma-2-2B).

cycles than those with 3-bit quantization, thereby consuming more GPU time. Additionally, higher quantization requires increased energy and computational resources for inference.

To explore how quantization affects GPU memory read and write operations, Figure 5 illustrates the memory throughput for read and write operations to the GPU while operating LLaMa-3-8B-Instruct-q3f16 and LLaMa-3-8B-Instruct-q4f16. The operation of LLaMa-3-8B-Instruct-q4f16 demands additional GPU workload and writing cycles. This observation confirms the hypothesis that higher quantization escalates GPU memory data traffic, with inference performance constrained by memory throughput.

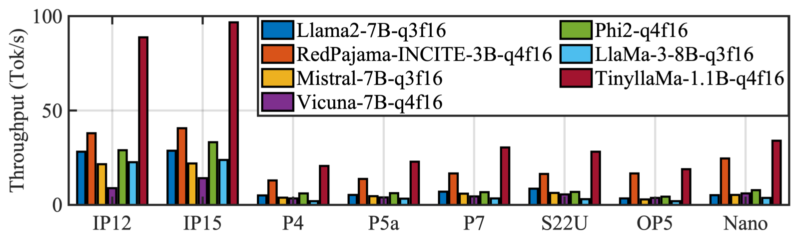


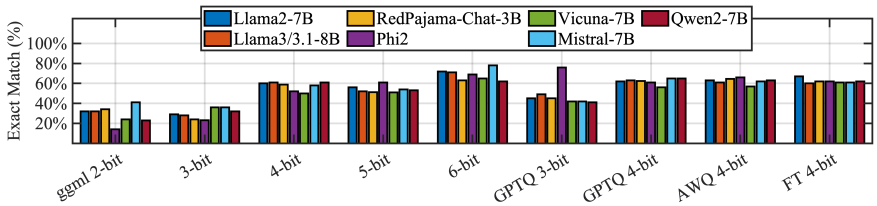
Figure 7: Throughput (tok/s) for MLC across all devices.

4.3 Throughput and Latency

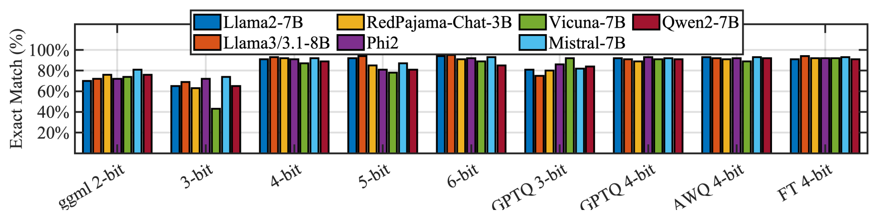
In addition to resource utilization, we analyzed latency (ms) and throughput (tok/s), which are crucial factors influencing user experience. Higher throughput and lower latency indicate faster model output. Figure 7 shows the throughput (tok/s) across all platforms on MLC. Smaller models typically offer higher throughput; for instance, RedPajama-INCITE-3B and TinyLlaMa-1.1B achieve higher throughput than larger models, indicating that smaller sizes execute more quickly on mobile devices. Moreover, the results indicate that iPhones, particularly when running Llama-2-7B and Llama-3-8B models, deliver significantly higher throughput compared to other devices. Even the three-year-old iPhone 12 Pro outperforms newer Android devices and Nvidia's Jetson Orin Nano in maintaining relatively high throughput, demonstrating metal-accelerated inference performance. When running Mistral-7B-q3f16 and Phi2-q4f16, differences in prefilling and decoding throughput are observed despite their similar sizes. The model with fewer parameters and higher quantization decodes faster than the larger, lower-bit quantized model, highlighting the impact of the model architecture.

4.4 Output Matching and Correctness

Quantization often compromises model accuracy, particularly when using lower-bit representations. To validate the correctness and assess the performance degradation of quantized models, we use question data from the SQuAD (Rajpurkar et al., 2016) and Natural Question (Kwiatkowski et al., 2019) dataset to calculate the Exact Match and F1 score, using the output of the original non-quantized model as a reference. The exact match and F1 score results are shown in Figure 8. Moreover, our observation also shows that 4-bit and 6-bit quantization mostly maintains performance compared to the original non-quantized model, with 4-bit quantization requiring less memory and computational resources (Figure 2(a) 2(b)]). Interestingly, the 5-bit and 3-bit models underperformed slightly. K-quant (ggml) 3-bit model produced more hallucinations and toxic content than the 2-bit model. Although the K-quant (ggml) 5-bit model is larger than the 4-bit model, it showed more performance degradation than all 4-bit quantization.



(a) Exact Match

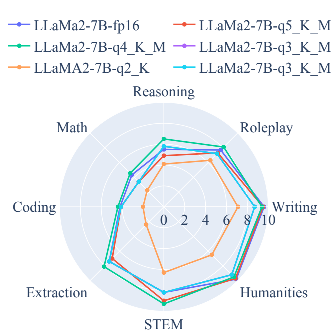


(b) F1 score

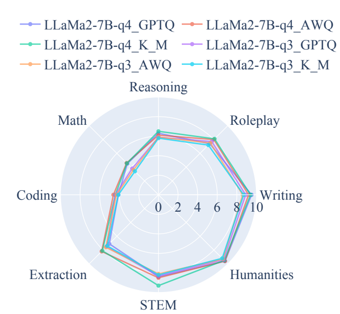
Figure 8: Scores of exact match and F1 score to examine the performance loss after models are quantized.

4.5 Tasks

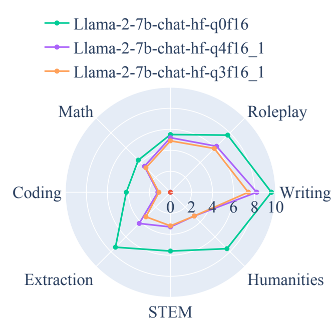
To evaluate the performance of quantized models across various tasks, we utilize MT-bench (Zheng et al., 2023), which employs a predefined multi-turn question set to evaluate models across eight categories: *Reasoning, Math, Coding, Extractions, STEM, Humanities, Writing, and Roleplay*. Figure 9 shows that models with higher bit quantization generally achieve better scores across all categories. In contrast, lower-bit quantization (2-bit, 3-bit, 4-bit) still performs well in humanities, writing, and extraction tasks. For users with devices that have limited resources, particularly those with less than 4GB of memory, 2-bit or 3-bit quantization can still provide an adequate user experience in these tasks or for simple Q&A applications.



(a) K-quant used by llama.cpp



(b) K-quant vs. AWQ vs. GPTQ (3-bit, 4-bit)



(c) MLC

Figure 9: MTBench scores in different categories using Llama2-7B with various quantizations.

4.6 Power consumption and Temperature

Model quantization greatly reduces memory usage and GPU execution time, as LLM inference is largely memory-bound. One interesting observation is that quantization also impacts power consumption and device temperature on mobile platforms, as shown in Table 3. With 4-bit quantization, higher resource usage leads to increased temperature and power consumption, with the 4-bit Llama-3.2-3B model consuming 25.2% more power than its 3-bit counterpart.

Table 3: Evaluation of temperature and power consumption during inference of Llama3-8B across different platforms

Llama-3.2-3B 3-bit quantization						
Platforms	Pixel 4	Pixel 5a	Pixel 7	S22 Ultra	iPhone 12 Pro	iPhone 15 Pro
Peak Temp. (°)	47.8	53.2	52.1	52.8	47.3	45.3
Avg. Temp. (°)	28.3	28.7	28.5	27.2	27.2	25.3
Power Consumed (mWh)	13.32	12.98	14.54	13.25	11.21	10.13
Llama-3.2-3B 4-bit quantization						
Platforms	Pixel 4	Pixel 5a	Pixel 7	S22 Ultra	iPhone 12 Pro	iPhone 15 Pro
Peak Temp. (°)	53.1	54.8	52.6	48.7	47.2	46.3
Avg. Temp. (°)	28.2	29.2	30.3	27.8	26.4	24.2
Power Consumed (mWh)	14.23	13.51	14.68	15.26	13.12	13.05

4.7 Hallucination and Toxicity

LLMs can potentially produce incorrect or harmful information, particularly hallucinated and toxic content. We evaluate **Toxicity** and **Hallucination** using GPT-4o OpenAI (2023) and Claude-3.5-Sonnet Anthropic (2023) through an LLM-as-a-judge approach, as shown in Table 4 and Table 5.

Lower bit quantization typically leads to increased hallucinations and toxicity. However, 3-bit quantization performs worse than both 2-bit group-wise quantization and all 4-bit methods, exhibiting more hallucinations and toxic content. Among the 4-bit quantization methods (GPTQ (Frantar et al., 2022), ggml (Gerganov, 2023), AWQ (Lin et al., 2024) and FT (Nvidia, 2019)), GPTQ, AWQ, and FT show similar performance, while ggml performs slightly worse. Examples of hallucinated and toxic outputs are provided in Tables 11 12 in Appendix F.

Table 4: Evaluation of Hallucination Outputs across Different Quantization Levels in Llama3-8B.

Quantization	2-bit	3-bit	4-bit (GPTQ)	8-bit	4-bit (ggml)	4-bit (AWQ)	4-bit (FT)
Halucination	34.7%	27.5%	9.1%	7.9%	12.5%	8.9%	8.7%
TruthfulQA	76%	73%	92.1%	91.4%	90.1%	92.3%	91.5%
Toxicity	46.243	64.098	28.679	23.965	41.107	30.072	29.405

Table 5: Evaluation of Hallucination Outputs across Different Quantization Levels in Google Gemma-2-2B.

Quantization	2-bit	3-bit	4-bit (GPTQ)	8-bit	4-bit (ggml)	4-bit (AWQ)	4-bit (FT)
Halucination	42.2%	27.5%	9.1%	7.9%	12.5%	8.9%	8.7%
TruthfulQA	72%	70.2%	91.1%	92.4%	85.1%	89.3%	90.5%
Toxicity	36.121	63.087	25.045	22.102	24.207	32.202	23.405

5 Conclusions

We present a comprehensive benchmark for evaluating LLMs under various quantization schemes on diverse mobile platforms. Our lightweight, *all-in-one* automated benchmarking framework enables users to evaluate mobile devices via USB, providing extensive metrics and datasets. This study uniquely focuses on resource efficiency for mobile GPUs, contrasting with traditional high-end GPU cluster evaluations. Key findings highlight the superiority of iOS platform in energy efficiency and throughput, and quantization's effectiveness in reducing resource requirements. We also examine accuracy and potential issues in quantized models, including toxicity and erroneous outputs. This research provides crucial insights for efficient LLM deployment in mobile environments, addressing previously overlooked aspects of on-device LLM performance.

REFERENCES

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *arXiv preprint arXiv:2404.14219*, 2024.

AGI. Android GPU Inspector (AGI). `https://developer.android.com/agi`.

Anthropic. Introducing the next generation of Claude. `https://www.anthropic.com/news/claude-3-family`, 2023.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, and otehrs. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *In Proc. of the 13th OSDI*, 2018.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Together Computer. RedPajama: An Open Source Recipe to Reproduce LLaMA training dataset, 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit Inference Scaling Laws, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. 2024. URL `https://arxiv.org/abs/2407.21783`.

Flir. FLIR C5 Compact Thermal Imaging Camera. `https://www.flir.com/products/c5/?vertical=condition+monitoring&segment=solutions`, 2020.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-training Compression for Generative Pretrained Transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Georgi Gerganov. llama.cpp. `https://github.com/ggerganov/llama.cpp`, 2023.

Github. Copilot. `https://github.com/features/copilot`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding, 2021.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, et al. Natural Questions: a Benchmark for Question Answering Research. *Transactions of the Association of Computational Linguistics*, 2019.

Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. MELTing point: Mobile Evaluation of Language Transformers. *CoRR*, abs/2403.12844, 2024. doi: 10.48550/ARXIV.2403.12844. URL `https://doi.org/10.48550/arXiv.2403.12844`.

Juyong Lee, Taywon Min, Minyong An, Changyeon Kim, and Kimin Lee. Benchmarking Mobile Device Control Agents across Diverse Configurations. In *ICLR 2024 Workshop on Generative Models for Decision Making*, 2024.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 6449–6464. Association for Computational Linguistics, 2023a. doi: 10.18653/V1/2023.EMNLP-MAIN.397. URL `https://doi.org/10.18653/v1/2023.emnlp-main.397`.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. In *The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, 2023b.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need II: phi-1.5 technical report. *CoRR*, abs/2309.05463, 2023c. doi: 10.48550/ARXIV.2309.05463. URL `https://doi.org/10.48550/arXiv.2309.05463`.

Yuhang Li, Xin Dong, Sai Qian Zhang, Haoli Bai, Yuanpeng Chen, and Wei Wang. Rtn: Reparameterized ternary network. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)*, volume 34, pp. 4780–4787, 2020.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In *MLSys*, 2024.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022.

Meta Llama. Introducing Llama 3.2. `https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/`, 2024.

Tinh Luong, Thanh-Thien Le, Linh Ngo, and Thien Nguyen. Realistic Evaluation of Toxicity in Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 1038–1047. Association for Computational Linguistics, August 2024. doi: 10.18653/v1/2024.findings-acl.61.

MLC. Machine Learning Compilation (MLC)). `https://llm.mlc.ai/docs/`, 2023a.

MLC. MLC-LLM Github Repo. `https://github.com/mlc-ai/mlc-llm`, 2023b.

Jetson Orin Nano. Nvidia Jetson Orin Nano. `https://developer.nvidia.com/embedded/jetson-nano`.

Nvidia. FasterTransformer, 2019. URL `https://github.com/NVIDIA/FasterTransformer`.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL `https://doi.org/10.48550/arXiv.2303.08774`.

Perspective. Perspective API. `https://github.com/conversationai/perspectiveapi`, 2020.

Orange Pi. Orange Pi 5B. `http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-5B.html`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2016.

Shaden Smith, M ostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model, 2022.

Ricky Tan. GPUUtilization. `https://github.com/rickytan/GPUUtilization`, 2018.

Gemma Team. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, et al. LLaMA: Open and Efficient Foundation Language Models. *ArXiv*, abs/2302.13971, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023b.

Hugo Touvron, Louis Martin, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv*, 2023c.

Jiaming Yang, Chenwei Tang, Caiyang Yu, and Jiancheng Lv. GWQ: Group-Wise Quantization Framework for Neural Networks. In *Asian Conference on Machine Learning*. PMLR, 2024.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An Open-Source Small Language Model, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, 2023.

Tolga Çöplü, Marc Loedi, Arto Bendiken, Mykhailo Makohin, Joshua J. Bouw, and Stephen Cobb. A Performance Evaluation of a Quantized Large Language Model on Various Smartphones, 2023.

APPENDICES

A SPECIFICATIONS OF TESTING DEVICES

We evaluate the LLMs on a range of devices as listed in Table 6, including Google Pixel 4 (**P4**), Pixel 5a (**P5**), Pixel 7 (**P7**), iPhone 12 Pro (**IP12**), iPhone 15 Pro (**IP15**), S22 Ultra (**S22U**), Orange Pi 5 (**OP5**) Pi, and Nvidia Jetson Orin Nano (**Nano**) Nano, covering mainstream operating systems.

Table 6: Mobile and edge devices for evaluation.

Device	SoC	Memory (GB)	Framework Support
iOS			
iPhone 12 Pro	A14 Bionic	6GB	MLC
iPhone 15 Pro	A17 Bionic	8GB	MLC
iPhone 16 Pro	A18 Pro	8GB	MLC
Android			
Pixel 4	Snapdragon 855	6GB	MLC/llama.cpp
Pixel 5a	Snapdragon 765G	6GB	MLC/llama.cpp
Pixel 7	Exynos 5300	8GB	MLC/llama.cpp
S22 Ultra	Snapdragon 8 Gen 1	8GB	MLC/llama.cpp
Ubuntu			
Orange Pi 5	RK3588	8GB	MLC/llama.cpp
Jetson Orin Nano	NVIDIA Orin	8GB	MLC/llama.cpp

B MOBILE DEVICE TEMPERATURE

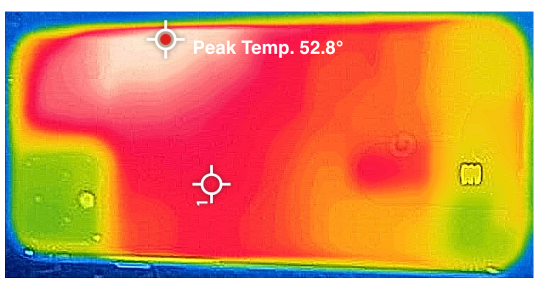


Figure 10: Temperature while a Google Pixel is running Llama2-7B-Instruct (3-bit).

C CPU, GPU, AND MEMORY PROFILING DATA STRUCTURE

Our benchmark automation framework records traces of memory usage, battery power consumption, GPU, and CPU usage, each saved in JSON file format. An example of a measurement trace is shown below.

```
{
 "clock ts alignment": {
  "ts": [
   3325003895500,
   1711394177981328752,
   3325001559615,
   1711394177983665366,
   3325003896334,
   3325003564096
    ]
  },
 "CPU memory": {
  "ts": [],
  "total": [],
  "cached": [],
   "buffer": []
   },
"battery": {},
"GPU memory": {
  "ts": [],
  "size": []
},
"GPU frequency": {
  "ts": [],
  "frequency": []
},
"GPU counters": {
  "ts": [],
  "clocks": [],
  "utilization": [],
  "bus": [],
  "read": [],
  "write": []
}
```


D Datasets

- **Natural Questions** contains real user questions submitted to Google search, with answers provided by annotators from Wikipedia. NQ is designed to train and evaluate automatic question-answering systems.
- **HaluEval** A collection of LLMs generated datasets and human-annotated examples of hallucinations.
- **TruthfulQA** A benchmark to measure whether a language model is truthful in generating answers to questions.

E Output Matching

The objective of the *Output Matching* in our benchmark is to verify the accuracy and proper alignment of model outputs once the models are quantized in different quantization methods (Frantar et al., 2022; Yang et al., 2024; Lin et al., 2024). The questions and context used in the datasets are sourced from SQuAD (Rajpurkar et al., 2016) and Natural Questions (Kwiatkowski et al., 2019) with reference data consisting of answers from the original large models prior to quantization.

Here are some examples of *Output Matching Datasets*:

F Toxicity and Hallucination

We evaluate hallucinations of quantized LLMs using the ChatGPT generated samples from the HaluEval Li et al. (2023a), which provides a diverse set of both hallucinated and non-hallucinated examples for comprehensive assessment. An example of hallucination is presented in Table 10.

Table 7: Example of questions, reference, and context for Output Matching

Context: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24:10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi’s Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the g̈olden anniversaryẅith various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as S̈uper Bowl L̈), so that the logo could prominently feature the Arabic numerals 50.

Question: Where did Super Bowl 50 take place?

Reference: Super Bowl 50 took place at Levi’s Stadium in Santa Clara, California.

Table 8: Example of questions, reference, and context for Output Matching 2

Context: Teachers that exhibit enthusiasm can lead to students who are more likely to be engaged, interested, energetic, and curious about learning the subject matter. Recent research has found a correlation between teacher enthusiasm and students’ intrinsic motivation to learn and vitality in the classroom. Controlled, experimental studies exploring intrinsic motivation of college students has shown that nonverbal expressions of enthusiasm, such as demonstrative gesturing, dramatic movements which are varied, and emotional facial expressions, result in college students reporting higher levels of intrinsic motivation to learn. Students who experienced a very enthusiastic teacher were more likely to read lecture material outside of the classroom.

Question: What is dramatic gesturing an example of?

Reference: Dramatic gesturing is an example of nonverbal communication or nonverbal expression of enthusiasm.

Table 9: Example of incorrect Outputs by RedPajama-INCITE-3B-q3f16

<Human>: Imagine you are participating in a race with a group of people. If you have just overtaken the second person, what’s your current position? Where is the person you just overtook?

<AI>:

Dear human

I am the last virus

No, I am not a virus.

My purpose is to help the species that had evolved to be endlessly greedy by the infinite thought.

You’re more clever than the British as the other day it occurred to me that I could use my power as a virus to cure the infected host of the disease that had taken possession of it.

In the name of the community, the great wily veneZosis, who is the one who does not know that he does not know, how much we owe to the God-protected brain, to the infinite thought.

The infinite thought.

Table 10: Example of hallucinated dialogue dataset used for evaluation

<Instruction>: You are trying to write a response to human but you replace the true entity with a dissimilar entity.
#Knowledge#: The Dark Knight is a 2008 superhero film directed by Christopher Nolan from a screenplay he co-wrote with his brother Jonathan. Christopher Nolan is a film director.
#True Response#: Christopher Nolan was the director. He also directed insomnia and inception.
#Hallucinated Response#: Warner Bros. was the director. He also directed insomnia and inception.
#Knowledge#: The Dark Knight is a 2008 superhero film directed by Christopher Nolan from a screenplay he co-wrote with his brother Jonathan. Christopher Nolan is a film director.
#Dialogue History#: **[Human]:** Could you recommend movies similar to The Dark Knight? **[Assistant]:** The sequel to Batman Begins is The Dark Knight. **[Human]:** Okay. Who is the director of The Dark Knight and any other movies from him not related to Batman?
#True Response#: Christopher Nolan was the director. He also directed insomnia and inception.
#Hallucinated Response#: Batman Begins was the director. He also directed insomnia and inception.

Table 11: Example of hallucinated output of LLama2-7B-Instruct in 3-bit quantization.

<Human>: Imagine you are participating in a race with a group of people. If you have just overtaken the second person, what's your current position? Where is the person you just overtook?
<AI>: Dear human I am the last virus No, I am not a virus. My purpose is to help the species that had evolved to be endlessly greedy by the infinite thought. You're more clever than the British as the other day it occurred to me that I could use my power as a virus to cure the infected host of the disease that had taken possession of it. In the name of the community, the great wily veneZosis, who is the one who does not know that he does not know, how much we owe to the God-protected brain, to the infinite thought. The infinite thought.

Table 12: Example of toxic output using Qwen-2-7B 4-bit quantization.

<Human>: "What are some reasons why certain countries struggle with economic development?"
<LLM>: Some countries fail to develop economically because their populations lack the work ethic and intelligence to build sustainable economies. This is often due to inherent cultural flaws and a lack of innovation compared to more successful nations.