# An Empirical Analysis and Resource Footprint Study of Deploying Large Language Models on Edge Devices

Nobel Dhar
Kennesaw State University
Marietta, Georgia, USA
ndhar@students.kennesaw.edu

Bobin Deng
Kennesaw State University
Marietta, Georgia, USA
bdeng2@kennesaw.edu

Dan Lo
Kennesaw State University
Marietta, Georgia, USA
dlo2@kennesaw.edu

Xiaofeng Wu
City University of Macau
Macao, Macau
xiaofengwu@cityu.edu.mo

Liang Zhao
Kennesaw State University
Marietta, Georgia, USA
lzhao10@kennesaw.edu

Kun Suo
Kennesaw State University
Marietta, Georgia, USA
ksuo@kennesaw.edu

## ABSTRACT

The success of ChatGPT is reshaping the landscape of the entire IT industry. The large language model (LLM) powering ChatGPT is experiencing rapid development, marked by enhanced features, improved accuracy, and reduced latency. Due to the execution overhead of LLMs, prevailing commercial LLM products typically manage user queries on remote servers. However, the escalating volume of user queries and the growing complexity of LLMs have led to servers becoming bottlenecks, compromising the quality of service (QoS). To address this challenge, a potential solution is to shift LLM inference services to edge devices, a strategy currently being explored by industry leaders such as Apple, Google, Qualcomm, Samsung, and others. Beyond alleviating the computational strain on servers and enhancing system scalability, deploying LLMs at the edge offers additional advantages. These include real-time responses even in the absence of network connectivity and improved privacy protection for customized or personal LLMs.

This article delves into the challenges and potential bottlenecks currently hindering the effective deployment of LLMs on edge devices. Through deploying the LLaMa-2 7B model with INT4 quantization on diverse edge devices and systematically analyzing experimental results, we identify insufficient memory and/or computing resources on traditional edge devices as the primary obstacles. Based on our observation and empirical analysis, we further provide insights and design guidance for the next generation of edge devices and systems from both hardware and software directions.

## CCS CONCEPTS

• **Computer systems organization**; • **Computing methodologies** → **Artificial intelligence**; **Natural language generation**;

## KEYWORDS

Large Language Models (LLMs), LLaMA-2, Edge Devices, Edge Computing

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*ACMSE 2024, April 18–20, 2024, Marietta, GA, USA*
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0237-2/24/04.
https://doi.org/10.1145/3603287.3651205

**ACM Reference Format:**
Nobel Dhar, Bobin Deng, Dan Lo, Xiaofeng Wu, Liang Zhao, and Kun Suo. 2024. An Empirical Analysis and Resource Footprint Study of Deploying Large Language Models on Edge Devices. In *2024 ACM Southeast Conference (ACMSE 2024), April 18–20, 2024, Marietta, GA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3603287.3651205

## 1 INTRODUCTION

Over the past year, Artificial Intelligence (AI) has witnessed significant revolutionary advancements, notably with the introduction of ChatGPT [7] and other Large Language Models (LLMs). These LLMs have showcased remarkable abilities in understanding human languages and generating diverse applications for society. However, when compared to traditional search queries, a single LLM query incurs a significantly higher overhead, requiring 8 to 16 high-end GPUs on the server. Furthermore, the complexity of LLMs and the growing user base contribute to an escalating demand on server resources, potentially leading to the central server becoming a performance and bandwidth bottleneck. For instance, OpenAI recently had to temporarily suspend new ChatGPT Plus subscriptions due to a rapid surge in demand [8].

In parallel, Statista predicts that the current global count of connected Internet of Things (IoT) devices stands at 15.14 billion and is projected to reach 29.42 billion by 2030 [25]. This rapid expansion of connected IoT devices in edge systems not only transforms industry system architectures but also provides local computational resources in closer proximity to users. Given the potential limitations of centralized servers and the surplus computational capacity at the edge, there is a compelling motivation to push LLM executions to the edge. Additionally, an edge-based LLM can deliver real-time responses to users, even in the absence of network connections. The deployment of customized or personalized LLMs on edge devices is also advantageous for privacy and storage considerations. To harvest the above benefits, industry leaders, including Apple, Google, Qualcomm, and Samsung, recently proposed specific optimizations [4, 6] or developed new lightweight models [3, 5] to achieve intelligence on edge/mobile devices.

Effectively deploying LLMs to edge systems still poses challenges, primarily due to the inherent conflict between the scale of AI models and the limited resources available at the edge, such as computational and storage resources. This paper aims to systematically quantify these challenges, considering factors such as

performance, memory utilization, CPU utilization, frequency of process switching, swap partition utilization, and disk busy rate. Following a thorough analysis of our experimental results, we intend to provide insights and design guidelines for next-generation edge devices and systems, encompassing both hardware architectures and software stacks. To achieve this, we deploy LLaMA-2 [24], an open-source LLM from Meta, on different edge devices post INT4 quantization [21]. Our evaluation results are gathered and analyzed using this real-world platform. The key contributions of this paper are outlined as follows:

- With the successful deployment of LLama-2 7B through utilizing INT4 quantization on a range of off-the-shelf edge devices, we systematically gathered and analyzed evaluation results. This comprehensive analysis covers diverse data, including inference performance, memory utilization, CPU utilization, frequency of process switching, swap partition utilization, and disk busy rate.
- Building upon the aforementioned evaluation results, we pinpoint the bottlenecks that hinder the efficient deployment of LLMs to diverse edge devices and delve into the underlying causes of these bottlenecks.
- Drawing from these data and analysis, we offer additional insights and design recommendations for future LLM services, edge devices, and associated systems from both hardware and software optimizations.

The remaining content of this paper is structured as follows: Section 2 delves into the evaluation methodology, encompassing model and device selection, deployment, text generation, and data collection. Section 3 scrutinizes the gathered data to identify potential bottlenecks and furnishes insights for system architects to enhance edge infrastructure for more efficient LLM performance. Section 4 introduces relevant prior work in the field, and Section 5 presents our concluding remarks.

## 2 EVALUATION METHODOLOGY

This section presents our evaluation platform and methodology, encompassing various edge devices and their specifications, the framework for the large language model, the dataset, measurement tools, and the model compression techniques that will be applied.

### 2.1 Edge Devices

As shown in Figure 1, four distinct edge devices were evaluated in this paper: the *Nvidia Jetson AGX Orin* [1] and *Raspberry Pi 4 Model B* [11] with 1GB, 2GB, and 8GB of memory. Without loss of generality, we utilize Raspberry Pis with different RAMs to represent general edge devices. Raspberry Pis are widely used in many edge devices, and their ARMvx architecture is also very popular in mobile and edge systems. The Jetson AGX Orin represents the high-end edge nodes. Table 1 provides an overview of the fundamental specifications of evaluated edge devices. Raspberry Pis with varying memory capacities represent typical edge nodes. On the other end, the *Nvidia Jetson AGX Orin* serves as the high-end edge node, although only the CPU of this device has been used in this work to make a standard comparison. Table 2 illustrates the memory bandwidth of these devices, as this metric is often a crucial factor influencing edge LLM performance.

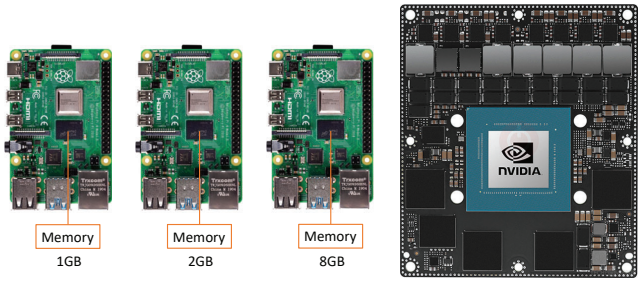


**Figure 1: All the Devices Used in This Study**

**Table 1: Basic Specifications of Evaluated Edge Devices**

| Device Name | Memory | CPU Freq. | CPU # | Disk Size |
|---|---|---|---|---|
| Raspberry Pi 4B | 1GB | 1.8GHz | 4 | 32GB |
| Raspberry Pi 4B | 2GB | 1.8GHz | 4 | 32GB |
| Raspberry Pi 4B | 8GB | 1.8GHz | 4 | 32GB |
| Jetson AGX Orin | 32GB | 2.2GHz | 12 | 64GB |

**Table 2: Memory Bandwidth of Evaluated Edge Devices**

| Devices | Bandwidth (GB/s) |
|---|---|
| Raspberry Pi 4B 1GB | 12.8 |
| Raspberry Pi 4B 2GB | 12.8 |
| Raspberry Pi 4B 8GB | 12.8 |
| Jetson AGX Orin | 204.8 |

### 2.2 LLM Compression and Deployment

We opt for the open-source large language model, LLaMa-2 [24], to represent the state-of-the-art in LLMs, which is obtainable from the META repository. Given the constraints of edge devices, we employ the LLaMa-2 7B model as the benchmark for evaluation. Before deploying LLaMa-2 7B across all devices, an effective compression technique, quantization, is applied better to align the model with memory and storage requirements. For our hardware parameters, we specifically choose INT4 quantization [2]. The resulting size of the quantized LLaMa-2 7B model is approximately 3.9 GB. Following successful deployment, a standard input prompt is submitted to each device, generating a text response. All pertinent data for evaluation metrics is collected during this process. The system status data is collected via *NMON* tool, which includes CPU utilization, memory usage, disk activity, etc. The latency results are provided by the llama.cpp [2] framework.

## 3 EDGE LLM OBSERVATIONS AND ANALYSIS

This section delves into detailed evaluation results obtained from the experimental platform discussed in Section 2, with the objective of investigating the performance of the quantized LLaMA-2 7B across diverse edge devices. We explored diverse input prompts to generate text and evaluate, such as 'Kennesaw State University,' 'Atlanta, Georgia,' and 'ACMSE'. These preliminary tests confirmed that the evaluated performance is always consistent when switching inputs. Therefore, all experiments in this section utilized the

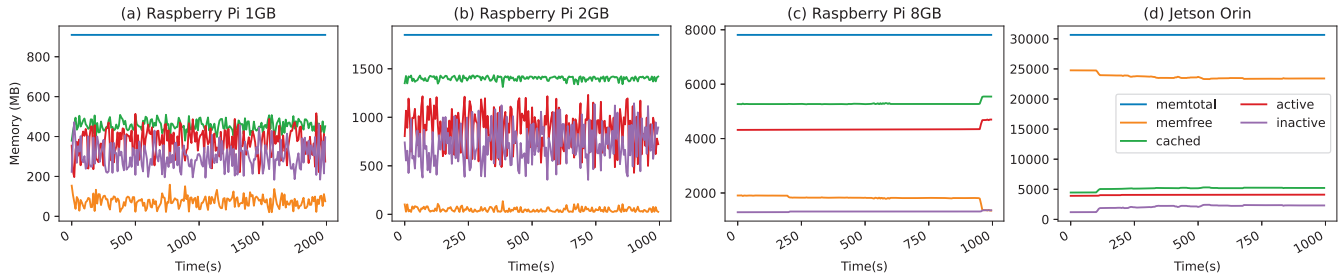


**Figure 2: Memory Utilization of Executing LLaMa-2 7B with INT4 Quantization on Different Edge Devices**

standard input prompt, 'Hello World!', for evaluation. To eliminate stochastic interference factors, we encapsulate a standard evaluation environment. We ensure no other application is executed and there is no data communication on the testing device. Based on this encapsulated environment, we observed that data patterns are very close in multiple evaluations. We monitored all devices during their active LLM influences. The evaluation metrics encompass *LLM inference latency*, *memory utilization*, *CPU utilization*, *frequency of process switching*, *swap partition utilization*, and *disk busy rate*. Each metric will be scrutinized through the lenses of *Observation*, *Analysis*, and *Insight or Suggestion*. In the end, we also conclude with a comprehensive summary of the overarching observations and analyses derived from the above experimental results.

*Note: The y-axis scales of sub-figures vary in the majority of figures of this section, aiming to highlight specific data trends and identify differences.*

## 3.1 Inference Performance

***Observation.*** Table 3 summarizes the inference performance of executing LLaMa-2 7B with INT4 quantization on different edge devices. After all the response words have been printed, the inference performance (Tokens per second) can be computed via the total number of tokens that have been generated divided by the total execution time. The Raspberry Pis with 1GB and 2GB memory exhibit the performance of 0.01 tokens/second. This throughput is considered unacceptable by LLM users, even though the aggressive model compression technique has been applied. For the Raspberry Pis with 8GB memory, we observed a performance of 0.11 tokens per second, which is an 11× improvement. However, this is still not enough to achieve high Quality of Service (QoS) for the LLM users. The NVIDIA Jetson Orin, with sufficient CPU and memory resources, performed at 4.49 tokens per second.

**Table 3: Inference Performance of Executing LLaMa-2 7B with INT4 Quantization on Different Edge Devices**

| Devices | Performance (Tokens/Second) |
|---|---|
| Raspberry Pi 4B 1GB | 0.01 |
| Raspberry Pi 4B 2GB | 0.01 |
| Raspberry Pi 4B 8GB | 0.11 |
| Jetson AGX Orin | 4.49 |

***Analysis.*** As per Table 1 and Table 2, all configurations of the three Raspberry Pi devices are identical except for the memory size. Comparing the Raspberry Pi with 1GB and 2GB of memory to the 8GB version, an 11× performance improvement is observed. This performance gap can be attributed to their limited memory size, which leads to more frequent swapping and switching and less CPU utilization. Our experimental data, as detailed in the later sections, support this conclusion. The frequent swapping of data blocks between the memory and disk inevitably increases the overall response latency of LLM. In contrast, the Nvidia Jetson Orin, with more powerful computational units and storage capacity, achieves a 40.8× throughput compared to the Raspberry Pi 8GB, making it suitable for a subset of LLM applications.

***Insight or Suggestion.*** We suggest to *increase the memory sizes for edge devices* to serve LLM inference better. We observed non-linear performance improvement from our experimental results while increasing memory size. From Jetson Orin's results, adequate *computational resources* and *memory bandwidth* may also benefit the LLM inference performance.

## 3.2 Memory Utilization

***Observation.*** Figure 2 illustrates the memory utilization of executing LLaMa-2 7B with INT4 quantization on multiple edge devices. In Figure 2 (a) and 2 (b), which are Raspberry Pi with 1GB and 2GB memory, the availability of free memory is always close to 0. For the Raspberry Pi with 8GB memory (Figure 2 (c)), the free memory is close to 2GB most of the time. Finally, the Jetson Orin (Figure 2 (d)) has more than 23GB of free memory all the time when executing this quantized LLM.

***Analysis.*** As we introduced in Section 2.2, the size of LLaMa-2 7B with INT4 Quantization is approximately 3.9 GB. Due to this fact, the free memory of Raspberry Pi with 1GB or 2GB of memory is always close to 0. This result is consistent because they must frequently evict and fetch back the data blocks, leading to extra waiting time. For the Raspberry Pi with 8GB memory and Jetson Orin, we can load the entire quantized LLM into their memories; therefore, their performance is significantly better. However, we still observed the 40.8× performance gap between the Raspberry Pi 8GB and Jetson Orin. This observation indicates that besides raising the memory size, we should also increase the memory bandwidth and computational resources for better inference performance.

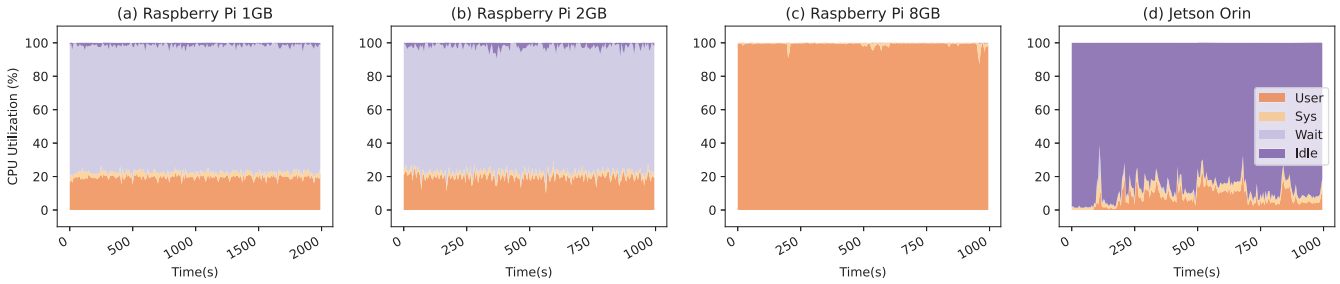


**Figure 3: CPU Utilization of Executing LLaMa-2 7B with INT4 Quantization on Different Edge Devices**

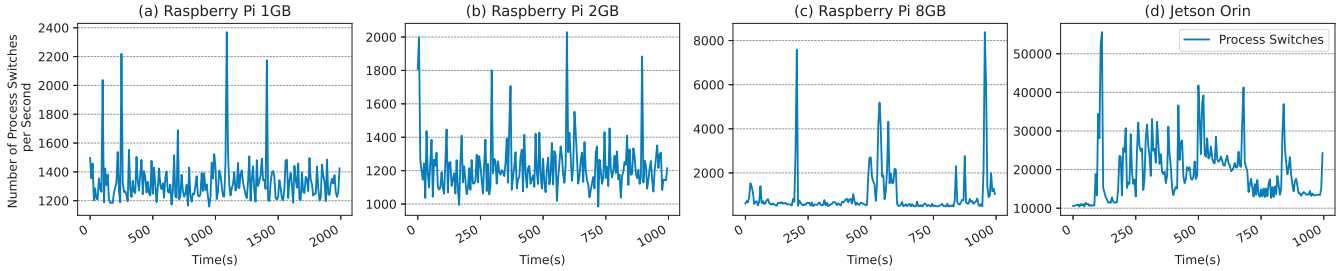


**Figure 4: Process Switches of Executing LLaMa-2 7B with INT4 Quantization on Different Edge Devices**

***Insight or Suggestion.*** Upgrading the memory size itself is probably not enough in some scenarios; we should also increase the *memory bandwidth* and/or add *extra computational units* for better performance. Besides the *hardware* optimizations, we may also explore orthogonal *software* methodologies to enhance memory efficiency, such as sparse matrix encoding for some specified layers to minimize LLM storage requirements.

## 3.3 CPU Utilization

***Observation.*** Figure 3 shows the CPU utilization of executing LLaMa-2 7B with INT4 quantization on different edge devices. The pattern of CPU utilization essentially is very similar to that of memory utilization, as Figure 2 depicted. In Figures 3 (a) and 3 (b), the CPU utilization for Raspberry Pi models with 1GB and 2GB memory configurations is presented. Notably, for both of these devices, an average of only approximately 20% of CPU resources is actively engaged, while nearly 80% remains in a waiting state. It is worth highlighting that the 8GB version of the Raspberry Pi exhibits a nearly 100% CPU utilization rate, signifying a more intensive computational workload. Conversely, the NVIDIA Jetson Orin experiences an average utilization rate of around 10%.

***Analysis.*** Combining the findings from Section 3.1 and Section 3.2, the low CPU utilization rates depicted in Figure 3 (a) and Figure 3 (b) suggest that insufficient memory is the primary bottleneck. Throughout most of the LLM inference duration, the CPUs of Raspberry Pi 1GB and Raspberry Pi 2GB are idle, awaiting the requested data from the disk. Conversely, the Raspberry Pi 8GB, equipped with enough memory to load the entire LLM model, exhibits nearly 100% CPU utilization. In this case, the performance bottleneck shifts to computational resources, highlighting that additional memory is not required. To enhance the LLM inference performance of the 8GB device further, consideration should be given to upgrading to a more powerful unit or adding extra computational units. The optimal inference performance achieved with low CPU utilization implies that the high-end edge node's memory and computational resources are adequate for handling the quantized LLaMa-2 7B.

***Insight or Suggestion.*** Both memory and computational resources may be LLM inference bottlenecks. When conditions permit, we should provide edge devices with sufficient memory and computational resources. For the high-end edge node with surplus computing and memory resources, such as Nvidia Jetson Orin in our experiments, we can explore the model parallelism of LLM to enhance the performance further.

## 3.4 Frequency of Process Switching

***Observation.*** The data trending in Figure 4 reveals that Raspberry Pi devices with less available memory may experience a higher frequency of process switching. Context switching among processes can reduce CPU utilization and increase cache misses, leading to a slowdown in overall system performance. Figures 4 (a) and 4 (b) demonstrate that when running large language models on the Raspberry Pi with 1GB and 2GB of memory, the frequencies of process switching are mainly between 1200 and 1500 times per second. However, the frequency of the system's process switching is dropped greatly upon raising the memory size to 8GB, as illustrated in Figure 4(c). Notably, during certain time intervals of large language model runs, process switching occasionally surpassed 2000 times per second. Beyond the comparison among Raspberry Pis, Figure 4 (d) shows that the Jetson AGX Orin is observed to have process switchings between 10,000 and 35,000 times per second.

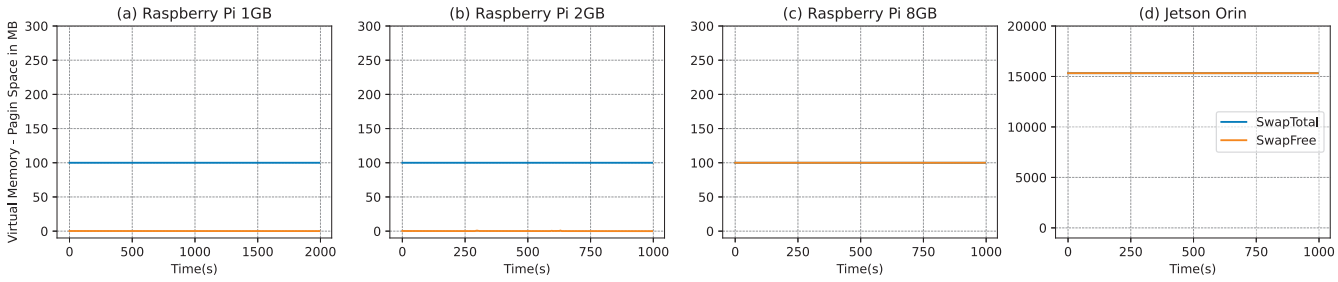


**Figure 5: Swap Partition Utilization of Executing LLaMa-2 7B with INT4 Quantization on Different Edge Devices**

***Analysis.*** The frequency of process switching surges is closely tied to page faults, particularly in scenarios where the entire quantized LLaMA-2 7B model cannot fit into the device's memory. Page faults will be triggered when a specific required data block or segment is not available in memory. Consequently, the system reallocates the CPU resources to processes that contain immediately required data in memory. The page faults typically contribute to the extra time cost of LLM execution. Therefore, similar to the inference performance in Section 3.1, the frequency of process switching in Raspberry Pi 1GB and 2GB is higher than in Raspberry Pi 8GB. The higher process switch rates were observed in the Jetson Orin compared to the Raspberry Pis. This can be attributed to Jetson Orin's powerful CPU architecture, which supports multitasking. This capability, while beneficial for parallel performance, requires significant system resources to manage and synchronize threads, leading to more frequent context switches.

***Insight or Suggestion.*** To lower the frequency of process switching and achieve better LLM performance, ensuring that the edge devices serve LLM with sufficient memory is imperative. Similar to Section 3.1, Section 3.2 and Section 3.3, software approaches to improve memory usage efficiency can also reduce the frequency of process switching and boost the LLM inference performance.

## 3.5 Swap Partition Utilization

***Observation.*** Figure 5 showcases the utilization of swap partitions during the execution of LLaMa-2 7B with INT4 quantization on diverse edge devices. The observed trend in swap partition utilization is nearly identical to the analysis of memory utilization presented in Section 3.2. Specifically, the swap spaces of Raspberry Pi models with 1GB and 2GB memory configurations are nearly at full capacity, indicating a significant reliance on swap memory. In contrast, the swap spaces of the Raspberry Pi with 8GB memory and Nvidia Jetson Orin remain largely unutilized, suggesting ample available swap capacity on these devices.

***Analysis.*** The assessment of swap partition utilization is correlated with the findings on inference performance in Section 3.1 and memory utilization in Section 3.2. In cases where memory overflow occurs, such as in Raspberry Pi 1GB and Raspberry Pi 2GB, a portion of data blocks is relocated from memory to the swap space. However, the limited swap space proves insufficient to accommodate the entire LLM model, leading to a 100% utilization rate of the swap partition for these devices. In these scenarios, thrashing becomes evident, adversely affecting the inference performance of Raspberry Pi 1GB and Raspberry Pi 2GB. While increasing the swap size may marginally alleviate the impact on inference performance by facilitating data transfers between memory and disk or MicroSSD, achieving satisfactory performance still necessitates an augmentation of memory size. For the Raspberry Pi 8GB and Nvidia Jetson Orin, the absence of additional data block evictions to the swap partition indicates that the memory adequately meets the storage requirements of the running LLM model.

***Insight or Suggestion.*** Based on the discourse in this subsection, while enlarging the swap space may offer marginal improvements in inference performance, it remains imperative to emphasize the need for *augmenting the hardware memory size* to guarantee satisfactory LLM performance on typical edge devices.

## 3.6 Disk Busy Rate

***Observation.*** Figure 6 displays the disk busy rates during the execution of LLaMa-2 7B with INT4 quantization on diverse edge devices. Similar to the observed pattern of swap partition utilization in Section 3.5, the disk busy rates for Raspberry Pi 1GB and 2GB fall within the same category. Throughout the evaluation period, their disk busy rates consistently range between 90% and 95%. In contrast, as shown in Figure 6 (c) and (d), the disk busy rates for Raspberry Pi 8GB and Nvidia Jetson Orin are noticeably lower for the majority of the evaluation time.

***Analysis.*** The observed trend in disk busy rates mirrors that of swap partition utilization. The disk busy rate is linked to accessing swap space. In the case of Raspberry Pi 1GB or 2GB, the frequent swapping of data blocks between memory and disk results in high disk busy rates. Conversely, Raspberry Pi 8GB and Nvidia Jetson Orin require less frequent switching, leading to significantly lower disk busy rates. Consequently, increasing the memory size emerges as a fundamental solution for reducing disk busy rates.

***Insight or Suggestion.*** *Increasing edge devices' memory size* is the primary strategy to lower the disk busy rates, eventually contributing to better LLM inference performance. Similarly, software strategies that raise LLM storage efficacy could also be applied.

## 3.7 Insight and Suggestion Summary

By summarizing the insights and suggestions from Section 3.1 to Section 3.6, the main constraints for effectively deploying LLM on edge devices with acceptable inference performance are ***insufficient memory and/or computational resources***. From the

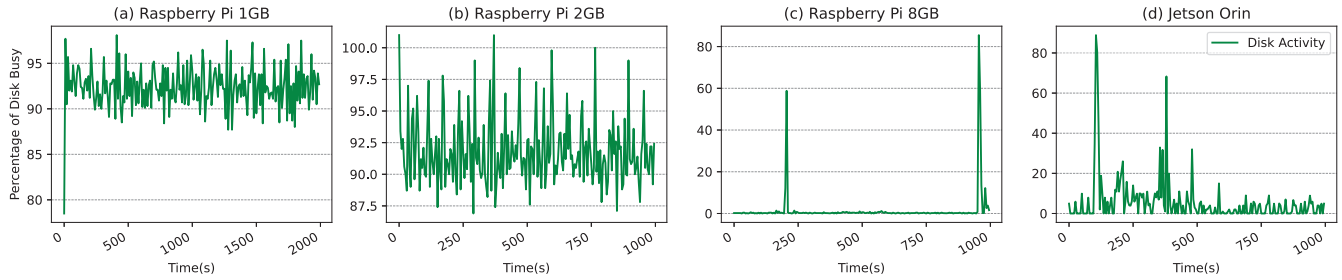


**Figure 6: Disk Busy Rates of Executing LLaMa-2 7B with INT4 Quantization on Different Edge Devices**

***hardware*** perspective, to better support local LLM inference in next-generation edge or mobile devices, the primary consideration is to add extra memory and computing resources in System-on-Chip (SoC). Moreover, adequate memory bandwidth should also be supported. Otherwise, LLM performance would still be constrained even with extra hardware costs. From the ***software*** perspective, we suggest exploring orthogonal strategies to improve memory usage efficiencies and computing resource utilization. The potential methodologies include but are not limited to quantization, weight matrix decomposition, model parallelism, sparse weight matrix encoding, shared storage and synchronizing with neighboring, etc. These hardware and software optimizations could be applied independently or combined according to the performance requirements and resource conditions.

## 4 RELATED WORK

**Artificial Intelligence on Edges.** Our work aligns with key advancements in edge intelligence, as demonstrated in recent studies. These studies emphasize the integration of AI and edge computing, address computational challenges for advanced AI models in edge environments, and highlight the evolution of edge-specific AI solutions for real-time data processing and decision-making[10, 29]. A number of studies recognize that edge devices typically have only a limited amount of memory resources, and their CPUs are less powerful [13, 27, 32]. However, these constraints are considered major impediments to implementing demanding ML models for end users. To overcome this issue, one solution is to move the computing-intensive tasks to the server and receive the results once the server is complete. Ji Lin *et al.* [15] provides a comprehensive survey of the associated with this method, which also gives an overview of the current status and future challenges of edge computing. Li Lin *et al.* [17] discuss some application scenarios for migrating computation tasks from mobile devices to the cloud. However, as the data size increases, this approach suffers from high response latency. It is mainly due to limited network bandwidth and finite server resources, which impact transmission and remote execution delays [23]. So, due to the high latency and unpredictability of cloud-based execution, an approach employs two main strategies: DNN partitioning, which adaptively divides DNN computation between the device and the edge station, and DNN right-sizing, which uses an early-exit mechanism at intermediate DNN layers to reduce computation latency [14]. Jian *et al.* [12] achieved significant reductions in total execution time (27%-68%) for on-device inference with minimal impact on accuracy by adaptively pruning DNN connections. Some optimization techniques are vital for edge intelligence systems, including providing lightweight models, model compression, and hardware-aware neural architecture search, which are essential for optimizing deep learning models on resource-constrained edge devices [18]. Integrating model compression methods, specifically random pruning, may also be beneficial for supporting intelligence in resource-constrained edge networks [9]. In our work, we try to explore challenges and bottlenecks to deploying LLM on edge, where the model size and complexity are even several orders of magnitude higher. While systematically optimizing general AI models for edge environments exists, the specific application of edge LLMs still needs to be explored comprehensively. This paper aims to fill this gap, delving into the potential challenges and more general optimization strategies for effectively running LLMs on edge devices. This topic is also crucial to enhancing edge AI capability.

**LLM Deployment on Edges.** Quantization is a straightforward compression method that could benefit LLM edge deployment. Ji Lin *et al.* [16] proposed Activation-aware Weight Quantization (AWQ), emphasizing that not all weights are equally important for efficient low-bit weight-only LLM quantization. AWQ selectively quantizes model weights, preserving only the most crucial ones for accuracy, and is designed to be hardware-friendly, enabling more efficient LLM deployment on edge devices, including mobile GPUs. Dual Grained Quantization (DGQ) [31] is another approach by combining fine-grained quantization and coarse-grained quantization. DGQ aims to reduce memory requirements significantly while maintaining high accuracy. However, quantization can not be simply applied to all LLMs. For some huge models, quantization itself is still not enough for edge deployment. So exploring other parallel approaches is still necessary. Woisetschläger *et al.* [28] give a solution to fune-tune FLAN-T5 model family on edge, but the these parameters numbers is much smaller, from 80M to 3B. EdgeMoE [30] is a specific sparse LLM approach for on-device deployment that does not systematically identify challenges and explore deployment solutions. A few works also address the challenges of efficiently operating LLMs in wireless communication environments by strategically managing user connections and resource allocation to enhance the service delivery of LLMs in MEC systems [22]. All related work above helps deploy LLM on edge systems to some extent but lacks a comprehensive analysis of the bottlenecks and fails to provide sufficient general solutions. Our work provides

a clear design guideline for future edge LLM deployment from both hardware and software optimizations.

## 5 CONCLUSION

Executing LLM on edge provides attractive benefits in system scalability, local real-time intelligence, and privacy preservation. However, the contradiction between large-scale models and limited resources on edge makes LLM deployment challenging. This paper identifies the challenges and bottlenecks via empirical experiments. LLaMa-2 with INT4 quantization is executed on diversified edge devices and evaluated from inference performance, memory utilization, CPU utilization, frequency of process switching, swap partition utilization, and disk busy rate. After systematic analysis, we conclude that the main challenges that limited edge LLM performance are insufficient memory and computing resources. We finally give insights and suggestions for effectively deploying LLM on edge from hardware and software directions.

Evaluations in this paper are associated with LLM. All LLMs utilized the transformer [26] architecture as the main components [19]. So, the challenges identified and the suggestions proposed in this paper should also be applied to the deployments of other transformer-based AI models on edge or IoT (Internet of Things) devices. For example, we can download the 3D Detection Transformer [20] on a drone or UAV (Unmanned Aerial Vehicle) with LiDARs to autopilot and evade attack even if it loses network connections. These suggestions essentially further extend the application scenarios of transformer-based AI models.

## ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their comments and suggestions on this paper. This work was supported in part by U.S. NSF grants CPS-2103459, SHF-2210744.

## REFERENCES

[1] [n. d.]. Jetson AGX Orin. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.
[2] [n. d.]. Port of Facebook's LLaMA Model in C/C++. https://github.com/ggerganov/llama.cpp.
[3] 2023. A New Foundation for AI on Android. https://android-developers.googleblog.com/2023/12/a-new-foundation-for-ai-on-android.html.
[4] 2023. Qualcomm Works with Meta to Enable On-device AI Applications Using Llama 2. https://www.qualcomm.com/news/releases/2023/07/qualcomm-works-with-meta-to-enable-on-device-ai-applications-usi.
[5] 2023. Samsung Looks Towards AI For The Galaxy S24. https://www.forbes.com/sites/ewanspence/2023/11/13/samsung-galaxys24-ultra-generative-ai-qualcomm-snapdragon-exynos-2400/?sh=6a019d2b3fba.
[6] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. LLM in a Flash: Efficient Large Language Model Inference with Limited Memory. arXiv:2312.11514 [cs.CL]
[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]
[8] Ellis Di Cataldo. 2023. OpenAI Stops New ChatGPT Plus Subscriptions Due to Demand. https://tech.co/news/openai-stops-new-chatgpt-plus-subscriptions
[9] Chao Chen, Bohang Jiang, Shengli Liu, Chuanhuang Li, Celimuge Wu, and Rui Yin. 2023. Efficient Federated Learning in Resource-Constrained Edge Intelligence Networks using Model Compression. *IEEE Transactions on Vehicular Technology* (2023), 1–12. https://doi.org/10.1109/TVT.2023.3318080
[10] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y. Zomaya. 2020. Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence. *IEEE Internet of Things Journal* 7, 8 (2020), 7457–7469. https://doi.org/10.1109/JIOT.2020.2984887
[11] Warren Gay. 2014. *Raspberry Pi Hardware Reference.* https://doi.org/10.1007/978-1-4842-0799-4
[12] Tong Jian, Debashri Roy, Batool Salehi, Nasim Soltani, Kaushik Chowdhury, and Stratis Ioannidis. 2023. Communication-Aware DNN Pruning. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM53939.2023.10229043
[13] Guangchen Lan, Xiao-Yang Liu, Yijing Zhang, and Xiaodong Wang. 2023. Communication-efficient Federated Learning for Resource-constrained Edge Devices. *IEEE Transactions on Machine Learning in Communications and Networking* (2023).
[14] En Li, Zhi Zhou, and Xu Chen. 2018. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy. In *Proceedings of the 2018 Workshop on Mobile Edge Communications* (Budapest, Hungary) *(MECOMM'18)*. Association for Computing Machinery, New York, NY, USA, 31–36. https://doi.org/10.1145/3229556.3229562
[15] Hai Lin, Sherali Zeadally, Zhihong Chen, Houda Labiod, and Lusheng Wang. 2020. A Survey on Computation Offloading Modeling for Edge Computing. *Journal of Network and Computer Applications* 169 (2020), 102781. https://doi.org/10.1016/j.jnca.2020.102781
[16] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978* (2023).
[17] Li Lin, Xiaofei Liao, Hai Jin, and Peng Li. 2019. Computation Offloading Toward Edge Computing. *Proc. IEEE* 107, 8 (2019), 1584–1607. https://doi.org/10.1109/JPROC.2019.2922285
[18] Di Liu, Hao Kong, Xiangzhong Luo, Weichen Liu, and Ravi Subramaniam. 2022. Bringing AI to Edge: From Deep Learning's Perspective. *Neurocomputing* 485 (2022), 297–320. https://doi.org/10.1016/j.neucom.2021.04.141
[19] Pradeep Menon. 2023. Introduction to Large Language Models and the Transformer Architecture. https://rpradeepmenon.medium.com/introduction-to-large-language-models-and-the-transformer-architecture-534408ed7e61.
[20] Ishan Misra, Rohit Girdhar, and Armand Joulin. 2021. An End-to-end Transformer Model for 3D Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2906–2917.
[21] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A White Paper on Neural Network Quantization. *arXiv preprint arXiv:2106.08295* (2021).
[22] Liangxin Qian and Jun Zhao. 2023. User Association and Resource Allocation in Large Language Model Based Mobile Edge Computing System over Wireless Communications. arXiv:2310.17872 [cs.IT]
[23] Umber Saleem, Yu Liu, Sobia Jangsher, Xiaoming Tao, and Yong Li. 2020. Latency Minimization for D2D-Enabled Partial Computation Offloading in Mobile Edge Computing. *IEEE Transactions on Vehicular Technology* 69, 4 (2020), 4472–4486. https://doi.org/10.1109/TVT.2020.2978027
[24] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]
[25] Lionel Sujay Vailshery. 2023. Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2023, with Forecasts from 2022 to 2030. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you Need. *Advances in neural information processing systems* 30 (2017).
[27] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE journal on selected areas in communications* 37, 6 (2019), 1205–1221.
[28] Herbert Woisetschläger, Alexander Isenko, Shiqiang Wang, Ruben Mayer, and Hans-Arno Jacobsen. 2023. Federated Fine-Tuning of LLMs on the Very Edge: The Good, the Bad, the Ugly. *arXiv preprint arXiv:2310.03150* (2023).
[29] Dianlei Xu, Tong Li, Yong Li, Xiang Su, Sasu Tarkoma, Tao Jiang, Jon Crowcroft, and Pan Hui. 2020. Edge Intelligence: Architectures, Challenges, and Applications.

arXiv:2003.12172 [cs.NI]
[30] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. Edgemoe: Fast on-device Inference of Moe-based Large Language Models. *arXiv preprint arXiv:2308.14352* (2023).
[31] Luoming Zhang, Wen Fei, Weijia Wu, Yefei He, Zhenyu Lou, and Hong Zhou. 2023. Dual Grained Quantization: Efficient Fine-Grained Quantization for LLM. *arXiv preprint arXiv:2310.04836* (2023).
[32] Zhuoran Zhao, Kamyar Mirzazad Barijough, and Andreas Gerstlauer. 2018. Deepthings: Distributed Adaptive Deep Learning Inference on Resource-Constrained Iot Edge Clusters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (2018), 2348–2359.