An Empirical Evaluation of LLM Inference Efficiency on Data Center GPUs and Edge Accelerators

Alessandro Costantini(✉), Giovanni Zurlo, and Diego Michelotto

INFN CNAF, Viale Carlo Berti Pichat 6, 40127 Bologna, BO, Italy
{alessandro.costantini,giovanni.zurlo,diego.michelotto}@cnaf.infn.it

Abstract. The rapid adoption of Large Language Models (LLMs) has raised significant concerns regarding their computational and energy efficiency, especially when deployed across heterogeneous hardware platforms. This work presents a systematic empirical evaluation of multiple open-source LLMs, including variants from the Qwen, DeepSeek, and LLaMA families, and reports the results of running these models on various hardware accelerators ranging from high-performance data center GPUs (NVIDIA H100, NVIDIA A100, AMD MI210) to resource-constrained edge devices (Raspberry Pi 5 with Hailo-10H module) are reported. The models run under a similar evaluation framework, to collect fine-grained metrics for both prefill and decode phases, including latency, throughput, power consumption, energy usage, and efficiency in terms of tokens per Joule. Additionally, the GPU utilization and memory footprint have been analyzed to better understand hardware–model interaction. The study provides actionable insights for selecting optimal hardware configurations under energy and performance constraints, contributing to the design of sustainable and scalable LLM deployment strategies.

Keywords: Accelerators · GPU · LLM · Energy Consumption

1 Introduction

Large Language Models (LLMs) represent an important component for recent artificial intelligence systems, enabling a wide range of applications like text generation, summarization, and conversational agents. The increasing size of the models, together with the increasing computational demands, promoted significant challenges in terms of deployment cost, latency, and energy consumption.

Already published works focused on improving model quality and scaling capabilities, demonstrating that there is a growing need to understand the efficiency implications of LLM inference, especially in real-world deployment scenarios. This became relevant in contexts where hardware resources vary widely, from high-end data center GPUs to low-power edge devices.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16758, pp. 53–69, 2027.
https://doi.org/10.1007/978-3-032-30497-1_4

An important aspect of LLM inference is the distinction between the prefill phase, where the input prompt is processed, and the decode phase, where tokens are generated autoregressively. These phases exhibit different computational patterns and resource utilization, which can significantly impact overall system performance and energy efficiency.

In this paper, a comprehensive evaluation of LLM inference across heterogeneous hardware platforms is presented, including NVIDIA H100 [22], NVIDIA A100 [20], AMD MI210 [1], and an edge setup based on Raspberry Pi 5 [26] with a Hailo-10H [27] accelerator. We analyze a diverse set of models spanning different sizes and optimization strategies, including full-precision and quantized variants. Our contributions are as follows:

1. we introduce a unified benchmarking framework that captures both performance and energy metrics at fine granularity;
2. we provide a comparative analysis of multiple LLM families under consistent workloads;
3. we quantify the impact of quantization on efficiency and resource utilization;
4. we explore the feasibility of edge deployment for lightweight LLMs.

The findings of this study aim to support practitioners and researchers in making informed decisions about LLM deployment under performance and energy constraints.
The remainder of this paper is structured as follows:
Section 2 outlines related work and our motivations; Sect. 3 details the methodology and initial considerations; in Sect. 3.4, the hardware and software infrastructure configured for the test is highlighted; in Sect. 3.5, the candidate models are listed with their peculiarities; in Sect. 4, the results are provided, together with some relevant considerations; finally, Sect. 5 summarizes the outcomes and draws conclusions.

2 Related Work

The growing computational and environmental costs for Large Language Models (LLMs) motivated a deep investigation aimed at improving their efficiency. Existing work spans model compression, system-level optimization, and energy-aware machine learning, each contributing important insights while leaving several open challenges.

Model compression techniques, including quantization and distillation, have proven effective in reducing the memory and computational requirements of LLMs. Methods such as GPTQ [9], LLM.int8() [7], AWQ [18], and ZeroQuant [34] demonstrate that low-bit quantization can significantly decrease resource usage with limited impact on model quality. Many open models have 8-bit or 4-bit quantized versions available. These allow, for instance, a 70B model to fit on a single 48 GB GPU in 4-bit mode. Similarly, distillation approaches [29] enable smaller models to approximate the behavior of larger ones. While these methods are highly effective at the model level, their evaluation is often conducted under

limited or homogeneous hardware settings, making it difficult to fully assess their impact across diverse deployment scenarios.

Complementary to model optimization, system-level research has focused on improving inference efficiency through hardware-aware design. Frameworks such as vLLM [16] and FasterTransformer [21] introduce optimized execution strategies, including memory-efficient attention mechanisms and improved batching techniques, achieving substantial gains in throughput and latency. Additionally, studies on modern accelerators [13,33] highlight the importance of architectural features such as memory bandwidth and parallelism. However, these works primarily target high-performance environments and typically emphasize throughput-oriented metrics, with less attention to energy efficiency and cross-platform comparability.

In parallel, the environmental impact of machine learning has been investigated in works such as Strubell et al. (2019) [31], Patterson et al. (2021) [23], and Henderson et al. (2020) [11], which emphasize the need for transparent reporting of energy consumption and carbon emissions. While these studies have been instrumental in raising awareness and establishing best practices, they largely focus on training workloads or provide coarse-grained estimates of energy usage, leaving inference—particularly autoregressive generation—less explored.

In the recent years edge-oriented architectures stimulated a lot of interest. Benchmarks such as MLPerf Tiny [3] and studies on edge AI systems [14] demonstrate that compact models can be deployed on such kind of devices and that these efforts highlight important trade-offs between performance, efficiency, and model capacity. However, the applicability of these approaches to modern LLM workloads remains limited and focused on smaller models or non-generative tasks.

In summary, while the above cited work provides valuable advances in individual dimensions of LLM efficiency, there is still a lack of unified empirical studies that jointly consider model characteristics, quantization strategies, and heterogeneous hardware platforms under consistent workloads and metrics. In particular, the distinction between prefill and decode phases in autoregressive inference has received limited attention in the literature.

This work builds upon these foundations by providing a systematic evaluation across multiple LLM families, quantization levels, and hardware configurations, with a specific focus on fine-grained performance and energy metrics. By explicitly analyzing both prefill and decode phases, we aim to offer a more comprehensive understanding of the trade-offs involved in real-world LLM deployment.

3 Methodology

The benchmarking methodology was designed to provide a reproducible and architecture-agnostic evaluation of Large Language Model (LLM) inference performance and energy efficiency, with particular emphasis on separating prompt processing (prefill) and token generation (decode) phases. The methodology consists of three main components: (i) prompt definition and categorization, (ii)

Table 1. Prompt Categories and Maximum Token Budgets.

Category	Max Input Tokens	Max Output Tokens
Content generation	128	256
Creative writing	512	512
Summarization (*light*)	1024	128
Summarization (*moderate*)	1566	256
Code analysis	2059	128
Summarization (*intermediate*)	3906	128
Summarization (*substantial*)	7631	128

phased inference probing, and (iii) metrics computation, including power and energy efficiency.

3.1 Prompts Definition

Our benchmark includes prompts from seven different tasks that vary the lengths of the input prompts and output responses to simulate different types of language model use. The prompts come from the OpenOrca [17] data set and the Government Report data set [12], while coding ones come from the MLPerf Benchmark [28]. Some of them, for example, request the summarization of longer passages of text, with maximum prompt sizes of about 4K and 8K characters. These prompts may exceed the context window limits of some edge computing systems such as with the Hailo 10H NPU. In total, each model is tested on 14 different prompts (Table 1).

3.2 Phased Inference Probing

The procedure loads individual prompts from JSON benchmark entries, replicates it to the configured batch size (set to 4 for all experiments), tokenizes with padding (and optional truncation), and runs a short warm-up generation pass before measurement. Measured inference is split into two stages, prefill and decode, because they exhibit distinct computational profiles and resource utilization patterns, enabling phase-specific performance characterization. In prefill, a forward pass processes the full prompt context in one step with key-value cache enabled, yielding high throughput and compute utilization. In decode, iterative steps reuse the cache to generate tokens one at a time, typically exhibiting memory-bound characteristics and lower token throughput. By separating the two phases, we measured differences in latency, power, and energy efficiency for each phase. The first generated token is selected from prefill logits, and each subsequent token is generated by feeding back the previous token and selecting by greedy sampling for comparability. Decode runs for the requested budget subtracting the first token already produced in prefill, while updating the attention mask to match the growing context length. Early stopping at generated End of Sequence (EOS) token is deactivated.

3.3 Metric Computation and Energy Estimation

Across prefill and decode, hardware telemetry is sampled asynchronously and summarized at phase level. For each phase, energy is estimated from the observed power trace over the measured phase duration, then average power and utilization are reported from the same window. Total energy is defined as the sum of the prefill and decode contributions, so the end-to-end figure remains directly comparable across runs.

Throughput is reported with phase-specific definitions to reflect how generation actually proceeds. In prefill, we report both first-token throughput (tokens/s) and prompt processing throughput (prompt tokens/s). In decode, throughput is computed from the tokens produced during iterative decoding. End-to-end throughput uses the actual number of generated tokens over total latency, and energy efficiency is expressed as tokens per joule using the same token count and total energy. Finally, average GPU and memory utilization are aggregated with phase-duration weighting, so the final utilization values are consistent with total runtime rather than simple phase averaging.

3.4 Hardware and Software Infrastructure

To evaluate the performance of Large Language Model (LLM) inference on different hardware architectures, the NVIDIA A100 PCIe (40 GB) has been considered as our experimental baseline [20]. The comparison included two high-performance candidates: the AMD Instinct MI210 (64 GB) and the NVIDIA H100 SXM5 (80 GB). This selection of hardware resources allows also the investigation of performance of two different software ecosystems, CUDA and ROCm, respectively.

The hardware selection allowed us to investigate two critical factors for LLMs: memory bandwidth and specialized compute engines. In fact, since the autoregressive decoding phase of large models is typically memory-bound, the MI210 (whit 1.6 TB/s HBM2e bandwidth and its expanded 64 GB VRAM) provides a high-capacity alternative to the A100, with official benchmarks indicating competitive parity in offline inference tasks [1]. On the other side, the H100 represents a significant generational step-forward for in hardware accelerators. By leveraging HBM3 specification, by achieving 3.35 TB/s bandwidth, the H100 introduces a dedicated Transformer Engine that is theoretically capable of delivering up to a 30-fold increase in inference throughput over the A100 baseline [22].

All benchmarks are conducted within a standardized virtualized environment to reflect modern cloud-native deployment patterns. The experiment utilizes virtual machines (VMs) provisioned with 16 virtual CPUs, leveraging the OpenStack cloud operating system for full virtualization. The underlying host hardware features high-performance AMD EPYC processors across different generations (including 7313 “Milan”, “Rome”, and “Genoa”), ensuring that the host CPU does not become a bottleneck during the pre-processing or tokenization phases. In our OpenStack-based IaaS environment, GPUs are exposed to virtual

machines via PCI passthrough. The GPUs are neither virtualized nor partitioned; each virtual machine therefore accesses the assigned GPU in bare-metal mode, with direct and exclusive access to the physical device.

For the sake of consistency and reproducibility, we employ optimized software stacks tailored to each architecture, using Python 3.10.12 and `PyTorch 2.8.0` as the primary execution framework. On NVIDIA Ecosystems, tests are executed using the CUDA 12.6 runtime, along with `transformers 5.5.0`, `GPTQModel 5.8.0` and `accelerate 1.14.0` libraries, as bundled in the quantization-latest-gpu official HuggingFace Docker image. The AMD Ecosystem uses the ROCm 7.1 open software platform, with the environment supported by `transformers 5.3.0`, `GPTQModel 5.8.0` and `accelerate 1.13.0` from the transformers-pytorch-amd-gpu image. This dual-stack approach allows us to measure not just raw hardware performance, but the maturity and efficiency of the respective software compilers and runtimes when handling high-throughput inference workloads across different hardware generations.

On the Raspberry Pi5 with the Hailo-10H LLM module, tests are executed in the Raspberry Pi operating system based on Debian 13 Trixie with Python 3.13.5 and `hailoRT 5.1.1`. The Hailo-10H is a second-generation edge AI acceleration module specifically designed to enable on-device inference of Large Language Models (LLMs), Vision-Language Models (VLMs), and other generative AI workloads under strict power constraints. The Hailo-10H module includes on-module LPDDR4/LPDDR4X memory with 8 GB memory, enabling the execution of transformer-based models without reliance on external memory bandwidth.

3.5 Candidate Models

The evaluation of candidate models required understanding of the structural mechanisms dictating hardware compatibility. Modern open-weight architectures have largely transitioned from naive scaling toward sophisticated topologies designed to maximize reasoning density while strictly bounding static memory footprints and dynamic memory bandwidth. We curated a cohort of compact models released between late 2024 and early 2026, selected for their architectural diversity, open-weight availability, and suitability for single-VM or edge deployment (See Table 3). Where applicable, native precision models are benchmarked alongside 4-bit quantized variants, including specific compiled versions from the Hailo Model Zoo [10].

Qwen2.5. Qwen2.5-1.5B-Instruct is a 1.54B parameter causal language model utilizing Rotary Positional Embeddings (RoPE), SwiGLU activations, and RMSNorm [25]. To optimize throughput and KV cache efficiency during inference, it employs Grouped Query Attention (GQA) with 12 query and 2 key-value heads. The architecture consists of 28 layers (1536 hidden, 8960 intermediate) and a 151,936-token vocabulary with tied embeddings. Pre-trained on a 7-trillion token multilingual corpus, it natively supports a 32,768-token context window and an 8,192-token generation length.

For Hailo benchmarking comparability, a 4-bit GPTQ quantized variant (128 group size, symmetric) by Unsloth [5,9] has been also tested. GPTQ minimizes layer-wise reconstruction error via the inverse Hessian matrix of the layer inputs, dynamically compensating for rounding errors.

Qwen3.5. The Qwen3.5 series utilizes the Qwen3-Next hybrid framework, transitioning to a native vision-language architecture [24]. It interleaves Gated Delta Networks (for efficient linear attention) with Gated Attention mechanisms (for global dependency modeling), supporting a native 262,144-token context window. In the study, three models have been evaluated:

- **0.8B**: Considered to be optimized for high-efficiency deployment, featuring 24 layers (1024 hidden), 16 linear attention heads, and 8 query/2 KV gated heads.
- **4B**: A 32-layer (2560 hidden) mid-scale model, it adopts an eight-block layout sequence (three Gated DeltaNet-FFN blocks followed by one Gated Attention-FFN block). It utilizes 32 linear heads and 16 query/4 KV gated heads, optimized for the integrated reasoning mode.
- **9B**: The upper-bound dense specialist maintains 32 layers but expands the hidden dimension to 4096 and the FFN to 12288. It retains 32 linear and 16 gated query heads, offering representational capacity that rivals much larger models in knowledge-intensive STEM benchmarks.

Llama 3.2. Llama-3.2-3B-Instruct is a 3.21B parameter model distilled from the 8B and 70B Llama 3.1 models and pre-trained on a 15-trillion token corpus [19]. It utilizes GQA and a 128k-token vocabulary, supporting a native 128k context window for long-form agentic tasks. At full FP16 precision, the model requires over 7 GB of VRAM. Because this footprint renders it unviable for edge hardware like the Raspberry Pi with sufficient KV cache, we evaluated three distinct deployment variants across the target platforms: On the NVIDIA and AMD GPUs, we benchmarked the native unquantized baseline alongside a 4-bit Quantized Low-Rank Adaptation (QLoRA) variant [8]. Utilizing the NormalFloat (NF4) data type and Double Quantization (DQ), this QLoRA variant shrinks the footprint to approximately 4.0 GB. For deployment on the Hailo-10H, we utilized a highly optimized, compiled A8W4 variant. By employing INT4 group-wise quantization for weights, its static footprint further compresses to 2.62 GB, achieving 10 to 12 tokens per second on the edge module.

DeepSeek. DeepSeek-R1-Distill-Qwen-1.5B draw out the advanced reasoning capabilities of the 671B Mixture-of-Experts (MoE) DeepSeek-R1 model into a dense transformer footprint [6]. Fine-tuned on the Qwen2.5-Math-1.5B base using 800,000 curated MoE generations, this 1.5B parameter model inherits complex Chain-of-Thought (CoT) behaviors, including self-verification and reflection. It achieves 28.9% on AIME 2024 and 83.9% on MATH-500. Like the Llama

Table 2. Native (Unquantized) Candidate Models: Architectural Comparison.

Model	$Q_{2.5}$-1.5B	$Q_{3.5}$-0.8B	$Q_{3.5}$-4B	$Q_{3.5}$-9B	$L_{3.2}$-3B	DS_{R1}-1.5B
Params (B)	1.54	0.8	4	9	3.21	1.5
Layers	28	24	32	32	28	28
Hidden Size	1536	1024	2560	4096	3072	1536
FFN size	8960	3584	9216	12288	8192	8960
Attention (Q/KV)	12/2	8/2	16/4	16/4	24/8	12/2
Linear Att. Heads (K/V)	–	16/16	16/32	16/32	–	–
Context Window	32,768	262,144	262,144	262,144	131,072	131,072
Vocabulary	151,936	248,320	248,320	248,320	128,256	151,936
RoPE	Yes	Yes	Yes	Yes	Yes	Yes
Tied Embeddings	Yes	Yes	Yes	No	Yes	No

variants, its memory footprint is constrained using bitsandbytes (BNB) 4-bit NF4 double quantization [4], preserving its logical integrity with minimal accuracy degradation.

3.6 Hailo GenAI Compiled Models

For NPU-accelerated benchmarking on the Raspberry Pi 5 platform, we evaluate highly optimized compiled models from the Hailo GenAI ModelZoo [10]:

- DeepSeek-R1-Distill-Qwen-1.5B
- Qwen2.5-1.5B-Instruct
- Llama-3.2-3B-Instruct

To enable foundation models to operate within the AI HAT+ 2's 8 GB memory and sub-5W power constraints, the Hailo-10H employs aggressive INT4 quantization, achieving 40 TOPS through a hybrid A8W4-INT architecture. Advanced optimization techniques, including GPTQ [9] and QuaROT [2], mitigate the accuracy loss caused by low-bit quantization by compensating for numerical error and normalizing activation outliers [32]. Benchmarks show that natural language understanding and reasoning tasks remain largely resilient to INT4 compression, whereas code generation tasks experience substantial performance degradation due to their sensitivity to precise token prediction [15] (Table 2).

4 Results

4.1 Native Precision Models

In this section we present a comparative analysis of Large Language Model (LLM) inference workloads executed on different architectures: NVIDIA A100,

Table 3. Hailo Compiled Models (A8W4): Architectural and Runtime Comparison.

Model	DeepSeek-R1-1.5B	Qwen2.5-1.5B	Llama3.2-3B
Quantization scheme	A8W4, Symm., G-W	A8W4, Symm., G-W	A8W4, Symm., G-W
Parameters	1.5B	1.5B	3B
Model size	2.37 GB	1.64 GB	3.13 GB
Context length	2048	2048	2048
Load time (s)	4.99	5.17	12.61
TTFT (s)	0.74	0.37	1.47
TPS	7.98	7.35	2.67

NVIDIA H100 and AMD MI210. Running a model in its native numerical precision (e.g., BF16, FP16, or FP8, rather than weight-only INT8/INT4 quantization) is often preferable when predictive accuracy, numerical robustness, or latency are prioritized over maximizing memory efficiency. The metrics collected focus on power consumption, resource utilization, throughput, and energy efficiency. Results are reported using both aggregated metrics per model and interval-averaged metrics per GPU (or NPU), as derived from the collected measurements.

Power Consumption. Figure 1 and Fig. 2 report the average power consumption during the decode and prefill phases, grouped by GPU architecture. The NVIDIA H100 consistently exhibits the highest absolute power consumption

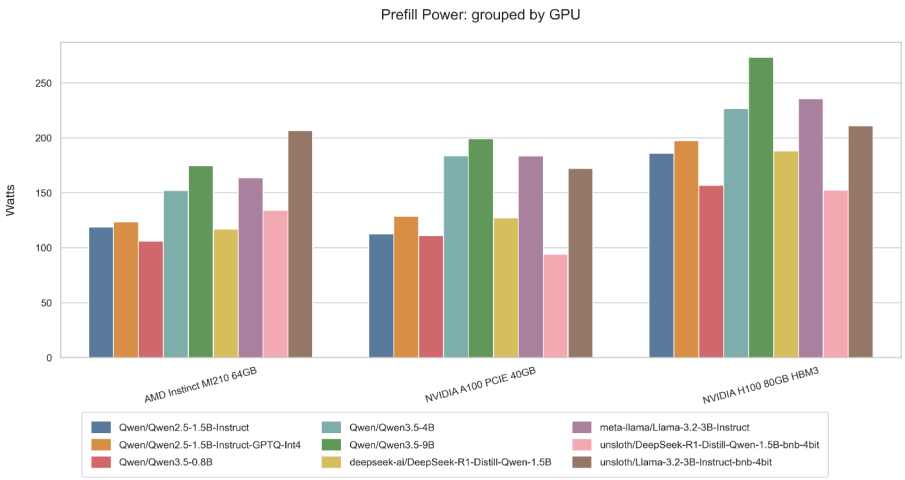

Fig. 1. Power consumption of the prefill phase in terms of Watt for each LLM model, grouped by accelerators

across all evaluated models in both phases. This behavior reflects the higher computational density of the H100 architecture.

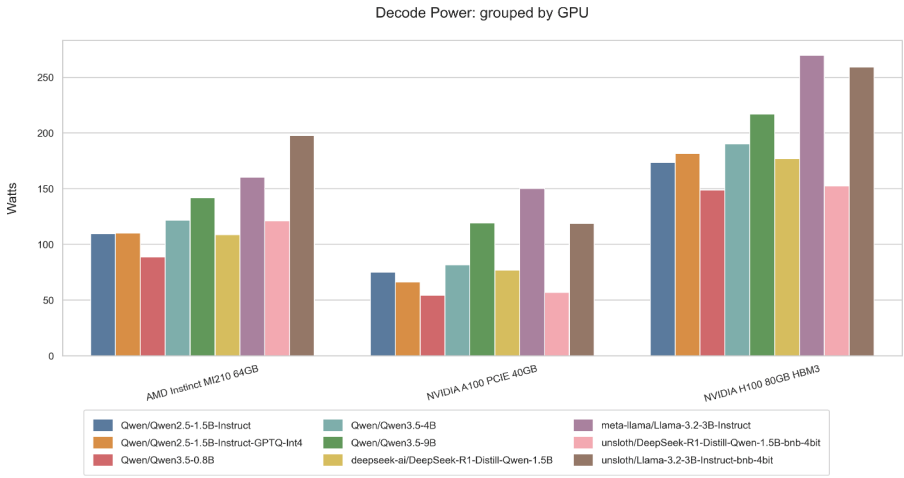

Fig. 2. Power consumption of the decode phase in terms of Watt for each LLM model, grouped by accelerators

In contrast, the NVIDIA A100 shows lower and more stable power consumption, with limited variability across models. The AMD MI210 presents power consumption that is in line, particularly during the prefill phase, with those of the NVIDIA A100, demonstrating the technical balance between the two models.

It is important to note that higher power consumption does not directly imply lower efficiency. Therefore, power metrics must be interpreted jointly with performance and energy-efficiency results.

GPU and Memory Utilization. In Fig. 3 the average GPU utilization is shown, highlighting clear architectural differences between the GPU used for this study. In particular, the AMD MI210 achieves the highest sustained utilization, indicating effective saturation of compute units by LLM inference workloads. On the other hand, NVIDIA A100 and NVIDIA H100 reach moderate utilization levels, suggesting partial under-utilization for some model configurations across workloads.

Memory utilization patterns, reported in Fig. 4, are primarily model-dependent. Larger models approach high memory occupancy on all platforms. AMD MI210 and H100 show similar memory utilization trends, despite the differences of their HBM capacities, while the NVIDIA A100 tends to reach high memory utilization earlier, potentially limiting scalability for larger models.

Overall, compute utilization differentiates GPU platforms more strongly than memory utilization.

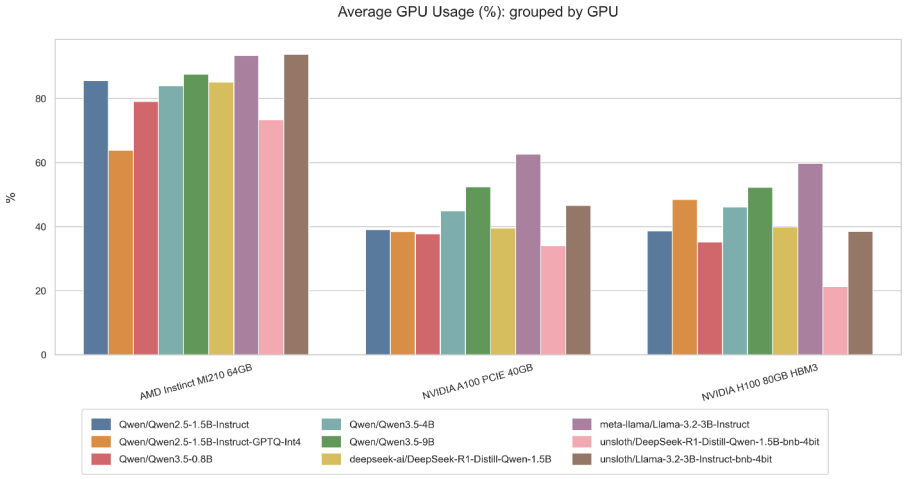

Fig. 3. GPU usage in terms of percentage for each LLM model, grouped by accelerators

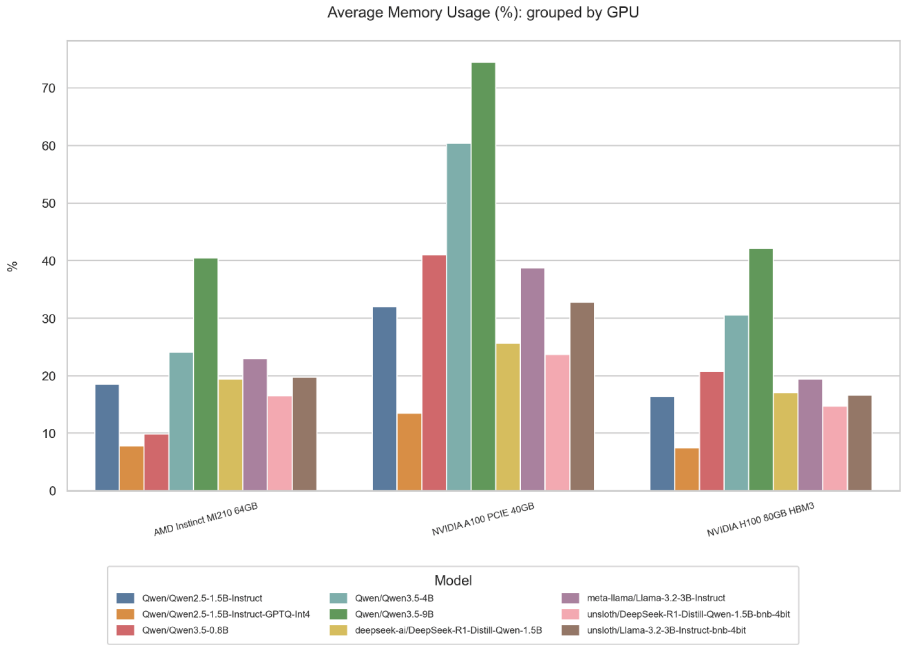

Fig. 4. Memory usage in terms of percentage for each LLM model, grouped by accelerators

Throughput and Energy Efficiency. Prefill and decode throughput results are reported in Figs. 5 and 6. The NVIDIA H100 consistently delivers the high-

est throughput across all evaluated models, with particularly strong performance during the decode phase. The NVIDIA A100 provides stable and predictable throughput, although at lower levels compared to the H100. The AMD MI210 shows substantially lower throughput, especially during prefill, indicating reduced token generation performance under the evaluated conditions.

These results highlight the impact of architectural features such as next-generation tensor cores and improved memory pipelines on LLM inference performance.

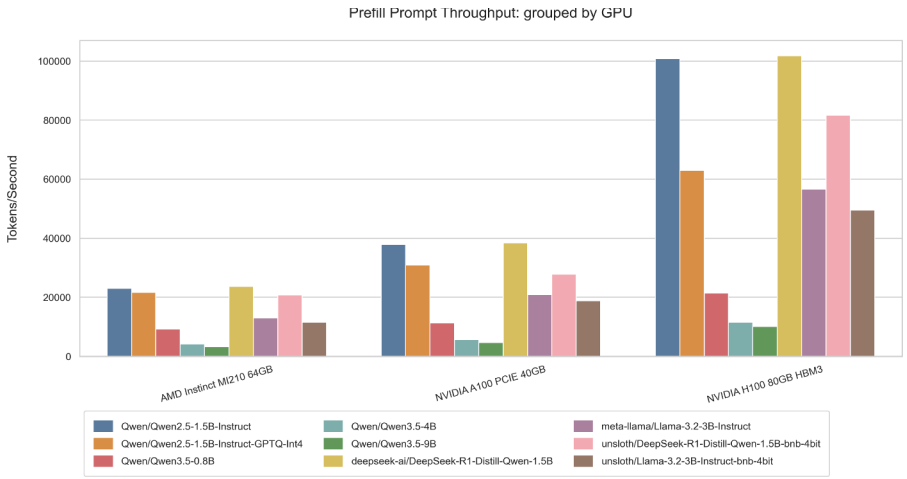

Fig. 5. Throughput of the prefill phase in terms of token per second for each LLM model, grouped by accelerators

Energy efficiency was evaluated using the tokens-per-joule metric, reported in Fig. 7. Considering the lower absolute power consumption, the NVIDIA A100 achieves the highest energy efficiency across all models. Despite their HBM hardware specifications, the NVIDIA H100 and AMD MI210 provides similar results in terms of efficiency indicating that lower power draw for AMD MI210 does not compensate for reduced throughput.

4.2 Quantized Models

This section presents a comparative analysis of the three Large Language Model (LLM) inference workloads executed on the Hailo-10H board against comparable quantized version on GPUs. It is important to note a technical caveat regarding the execution of 4-bit weights-only quantized models on the evaluated data center GPUs. While architectures such as Hopper feature advanced Tensor Cores that natively support mixed-precision formats (e.g., FP8, FP16, BF16, and TF32), they lack native hardware acceleration for 4-bit computation [22].

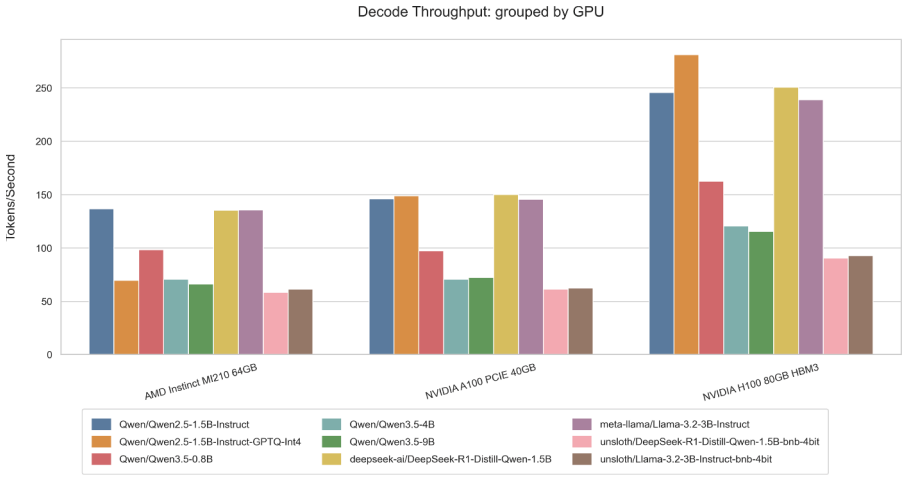

Fig. 6. Throughput of the decode phase in terms of token per second for each LLM model, grouped by accelerators

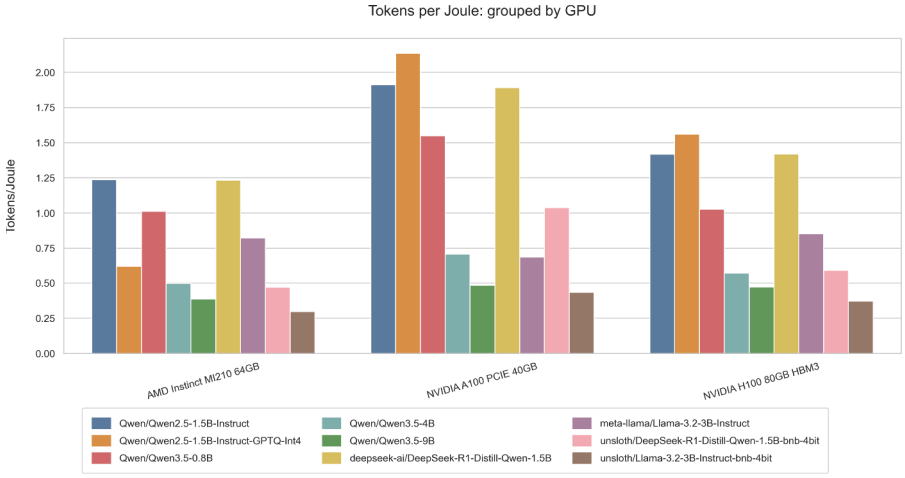

Fig. 7. Efficiency in terms of token per joule for each LLM model, grouped by accelerators

Consequently, running INT4 or 4-bit quantized models on Hopper platforms primarily optimizes memory footprint and bandwidth. During execution, these architectures rely on a kind of software emulation, requiring the 4-bit weights to be dequantized to INT8 or FP16 in the register file prior to computation. The quantised models were presented in Sect. 3.6, but for each of them, a GPU version and a compiled NPU version are actually compared:

– DeepSeek-R1-Distill-Qwen-1.5B
– Qwen2.5-1.5B-Instruct
– Llama-3.2-3B-Instruct

Figure 8 and Fig. 9 show the decode throughput and the decode power consumption, respectively, on the four platforms. Figure 10 shows instead the efficiency in terms of token per joule. Analyzing the outcomes from the three figures it is clear that the Hailo-10H shows all the typical features of a low power platform positioning itself in an excellent ranking, depending on the model chosen, even compared to the more renowned H100 in our test. On the contrary, the decode throughput shows a value well below the 40 tokens per second considered to be the minimum acceptable threshold for interactive LLM usage [30].

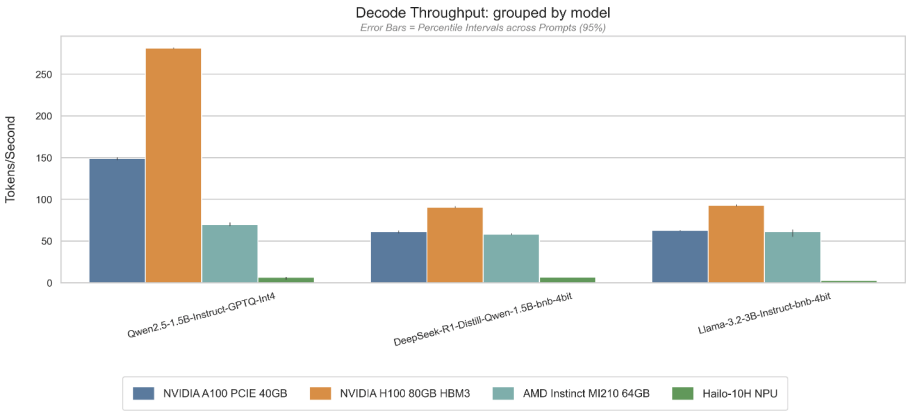

Fig. 8. Throughput in terms of token per second for the 4 accelerators, grouped by LLM model

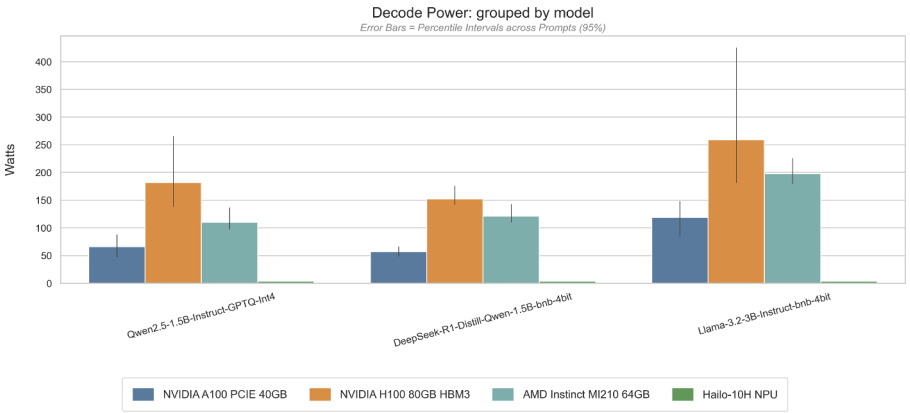

Fig. 9. Power consumption for the decode phase in terms of Watt for the 4 accelerators, grouped by LLM model

Although accelerators such as Hailo-10H do not meet the commonly accepted threshold for interactive LLM usage (≈40 tokens/s), they remain well suited for offline, edge, and background inference workloads, where energy efficiency dominates over latency. In such scenarios, token-per-joule efficiency becomes the primary optimization metric rather than raw throughput.

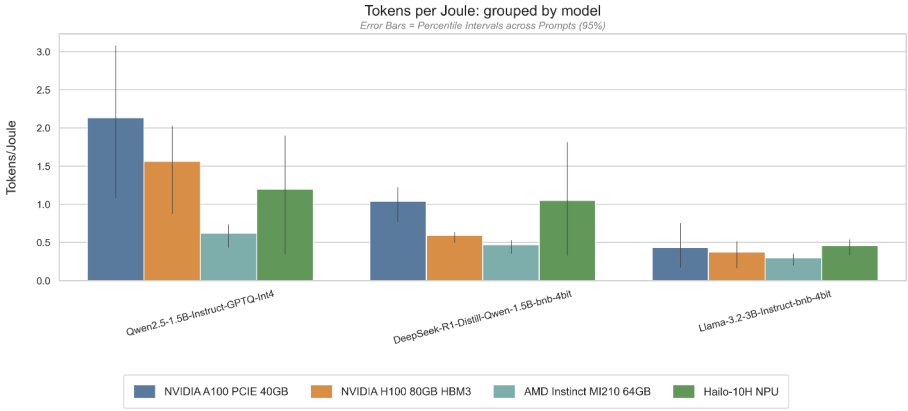

Fig. 10. Efficiency in terms of token per joule for the 4 accelerators, grouped by LLM model

5 Conclusions

The research activity presented in this study is intended to provide a concrete understanding on the trade-offs involved in deploying LLMs on different computing platforms. By analyzing performance and energy efficiency metrics under consistent workloads, this study highlights how architectural features, hardware design and optimization techniques may influence LLM inference behavior. Rather than identifying just a single optimal deployment solution, the results emphasize the importance of quantization strategy and hardware capabilities, having the model sizes aligned on the proposed usage scenario, in particular when considering throughput, latency, and energy consumption.

From a performance and energy-efficiency perspective, the NVIDIA H100 emerges as the most suitable platform for high-throughput LLM inference, particularly for decode-intensive workloads. The NVIDIA A100 remains a reliable and predictable solution for production environments with moderate performance requirements and costs. The AMD MI210, while exhibiting similar power consumption in respect to the NVIDIA A100, underperforms in terms of throughput and energy efficiency, suggesting that current limitations are primarily related to software maturity of ROCm suite and configuration tuning rather than hardware power characteristics alone.

Overall, the results indicate that energy efficiency in LLM inference workloads is driven primarily by achievable throughput rather than by raw power consumption. Moreover, while the NVIDIA H100 achieves the highest energy efficiency, confirming that its higher absolute power consumption is compensated by substantially higher throughput, the AMD MI210 shows the lowest tokens-per-joule values, indicating that reduced power draw does not offset lower performance.

Regarding the Hailo-10H module, it seems clear that the low energy consumption features do not compensate for the throughput which, being below 40 tokens per second, does not allow for responsive usage. Despite that, Hailo-10H remain well suited for scenarios (e.g. offline, edge, and background inference workloads) where the token-per-joule efficiency becomes the primary optimization metric rather than raw throughput.

References

1. Advanced Micro Devices, Inc. (AMD), et al.: AMD Instinct^{TM} MI210 accelerators (2022). https://www.amd.com/en/products/accelerators/instinct/mi200/mi210.html
2. Ashkboos, S., et al.: QuaRot: outlier-free 4-bit inference in rotated LLMs (2024). https://arxiv.org/abs/2404.00456
3. Banbury, C. et al.: MLPerf Tiny benchmark (2021). https://arxiv.org/abs/2106.07597
4. Bitsandbytes Foundation, et al.: Bitsandbytes-foundation/bitsandbytes: accessible large language models via K-bit quantization for PyTorch. https://github.com/bitsandbytes-foundation/bitsandbytes
5. Daniel Han, M.H., et al.: Unsloth (2023). https://github.com/unslothai/unsloth
6. DeepSeek-AI, et al.: DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning (2025). https://arxiv.org/abs/2501.12948
7. Dettmers, T., et al.: LLM.int8(): 8-bit matrix multiplication for transformers at scale. In: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS'22 (2022)
8. Dettmers, T., et al.: QLoRA: efficient finetuning of quantized LLMs (2023). https://arxiv.org/abs/2305.14314
9. Frantar, E., et al.: GPTQ: accurate post-training quantization for generative pre-trained transformers. In: 11th International Conference on Learning Representations (2023)
10. Hailo-AI, et al.: Hailo Model Zoo. https://github.com/hailo-ai/hailo_model_zoo
11. Henderson, P., et al.: Towards the systematic reporting of the energy and carbon footprints of machine learning (2022). https://arxiv.org/abs/2002.05651
12. Huang, L., et al.: Efficient attentions for long document summarization. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, June 2021, pp. 1419–1436. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.naacl-main.112, https://aclanthology.org/2021.naacl-main.112
13. Jouppi, N.P., et al.: Ten lessons from three generations shaped Google's TPUv4i : industrial product. In: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), pp. 1–14 (2021). https://doi.org/10.1109/ISCA52012.2021.00010

14. Khouas, A.R., et al.: Training machine learning models at the edge: a survey (2024). https://arxiv.org/abs/2403.02619
15. Kurtic, E., et al.: "give me BF16 or give me death"? accuracy-performance trade-offs in LLM quantization (2025). https://arxiv.org/abs/2411.02355
16. Kwon, W., et al.: Efficient memory management for large language model serving with PagedAttention. In: Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (2023)
17. Lian, W., et al.: OpenOrca: an open dataset of GPT augmented FLAN reasoning traces (2023). https://huggingface.co/datasets/Open-Orca/OpenOrca
18. Lin, J., et al.: AWQ: activation-aware weight quantization for on-device LLM compression and acceleration. In: Gibbons, P., Pekhimenko, G., Sa, C.D. (eds.) Proceedings of Machine Learning and Systems, vol. 6, pp. 87–100 (2024)
19. Meta Llama AI team, et al.: The Llama 3 herd of models (2024). https://arxiv.org/abs/2407.21783
20. NVIDIA Corporation, et al.: NVIDIA A100 tensor core GPU architecture (2020). https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf
21. NVIDIA Corporation, et al.: FasterTransformer (2021). https://github.com/NVIDIA/FasterTransformer
22. NVIDIA Corporation, et al.: NVIDIA H100 tensor core GPU architecture (2023). https://resources.nvidia.com/en-us-hopper-architecture/nvidia-h100-tensor-c
23. Patterson, D., et al.: Carbon emissions and large neural network training (2021). https://arxiv.org/abs/2104.10350
24. Qwen Team, et al.: Qwen3.5: towards native multimodal agents, February 2026. https://qwen.ai/blog?id=qwen3.5
25. Qwen Team, Yang, A., et al.: Qwen2.5 technical report (2025). https://arxiv.org/abs/2412.15115
26. Raspberry Pi Ltd., et al.: Raspberry Pi 5 (2023). https://www.raspberrypi.com
27. Raspberry Pi Ltd., et al.: Introducing the Raspberry Pi AI HAT+ 2: generative AI on Raspberry Pi 5 (2026). https://www.raspberrypi.com/news/introducing-the-raspberry-pi-ai-hat-plus-2-generative-ai-on-raspberry-pi-5/
28. Reddi, V.J., et al.: MLPerf inference benchmark. In: Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture, ISCA 2020, pp. 446–459. IEEE Press (2020). https://doi.org/10.1109/ISCA45697.2020.00045
29. Sanh, V., et al.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2020). https://arxiv.org/abs/1910.01108
30. Song, P., et al.: How many tokens per second is "good" for local LLMs? ML Journey, February 2026. https://mljourney.com/how-many-tokens-per-second-is-good-for-local-llms/, analysis of LLM inference speed, usability thresholds, and user perception in tokens per second
31. Strubell, E., et al.: Energy and policy considerations for deep learning in NLP (2019). https://arxiv.org/abs/1906.02243
32. Team et al.: Edge AI deployment of LLMs using Hailo-10H AI accelerators, January 2026. https://web.archive.org/web/20260118135652/https://hailo.ai/blog/bringing-generative-ai-to-the-edge-llm-on-hailo-10h/
33. Wan, Z., et al.: Efficient large language models: a survey (2024). https://arxiv.org/abs/2312.03863
34. Yao, Z., et al.: ZeroQuant: efficient and affordable post-training quantization for large-scale transformers. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems, vol. 35, pp. 27168–27183. Curran Associates, Inc. (2022)