OXFORD JOURNAL
OF STUDENT
SCHOLARSHIP

Hardware-Software Interplay: Measuring the Energy Cost of Local LLM Inference Across Integrated Testbed Profiles on Consumer Hardware

Aaryan Karlapalem
aaryanlk@icloud.com

ABSTRACT

As large language models (LLMs) evolve from data centers to local edge devices, understanding the hardware-software interactions of these systems has become essential for creating sustainable forms of AI. This study investigates the influence of host operating systems and machine specifications on the overall energy efficiency of three common on-device LLMs: Gemma 2:2B, Llama 3.2:3B, and Mistral 7B, running locally via Ollama. By measuring both the average power draw and execution time across the 3 most common consumer operating systems of macOS 26, Windows 11, and Debian Linux 13, the 'energy-per-token' cost of local inference can be calculated. The experimental results indicated that Mistral 7B consistently requires the highest energy input, while Gemma 2:2B is far more efficient for shorter tasks. Statistical analysis via a Two-Way ANOVA ($p < 0.05$) confirms that the integrated testbed profiles significantly influence energy outcomes. These findings provide a concrete framework for optimizing local AI deployments to balance performance with environmental sustainability.

Keywords: LLMs; energy efficiency; local LLM inference; operating systems; token throughput; energy-per-token

INTRODUCTION

Artificial intelligence (AI) has experienced a surge of growth over the last 5 years, with large language models (LLMs) being some of the most widely used AI systems. These models power applications ranging from everyday voice assistants and code completion tools to fully integrated research support and decision-making platforms. As LLMs scale in size and capability, however, their computational demands also increase significantly, increasing pressure on global energy infrastructure and data center capacity. Recent research indicates that AI data centers already account for a notable share of global electricity consumption, with AI workloads becoming an increasingly important driver of this demand (Daniel et al., 2024; Kang & Park, 2025). Unlike early AI systems, though, modern LLMs rely on continuous, high-volume numerical computation during inference, meaning that even routine user interactions can

accumulate substantial energy costs when scaled across sustained usage patterns and large user bases (You, 2025). As AI deployment continues to expand from centralized cloud infrastructure to consumer devices and enterprise workstations, understanding how efficiently these models operate across different computing environments has become an increasingly important research objective.

LITERATURE REVIEW

The environmental effects of this trend have become a topic of concern for both the computer science and sustainability fields. Although AI today still has less of an environmental impact as opposed to fields like agriculture, its rapid growth is what raises numerous questions about its long-term environmental impact potential. Researchers have documented the energy and water requirements of modern AI systems, emphasizing that efficiency improvements are essential to prevent AI deployment from conflicting with climate goals (Daniel et al., 2024). However, much of this existing research approaches AI energy use at a macro level and focuses on large-scale data centers or specialized systems as opposed to the hardware environments in which many LLMs are commonly deployed. Although these prior studies provide valuable insight into AI's environmental footprint, relatively little attention has been given to how the choice of host machine affects the energy efficiency of LLM inference. Current research looks at energy optimization at the level of cloud infrastructure, accelerators, or edge devices, but it does not compare general-purpose machines like CPUs and GPUs that are commonly used in research labs, personal workstations, or small-scale deployments. This gap is becoming increasingly significant, especially as LLMs continue to move beyond centralized cloud platforms and into diverse computing environments.

Evolution of Artificial Intelligence and Computational Requirements

Early AI systems were mostly symbolic and rule-based, which effectively pre-defined the algorithms needed to emulate AI. These systems also operated on limited datasets, but they required relatively less power compared to the modern machine learning models today (Dreyfus & Dreyfus, 1988). While influential people like Alan Turing laid the important theoretical groundwork for machine intelligence (machine learning), these early AI applications were narrow in scope and were constrained by both hardware and algorithmic limitations (Copeland, 2025). Symbolic approaches ultimately struggled to scale to real-world complexity, and critics argued that intelligence depends on context, intuition, and experiential learning, which were features that could not be captured by formal rules alone (Dreyfus & Dreyfus, 1988). These limitations prompted a shift toward more data-driven methods, especially neural networks, which aimed to model intelligence as an emergent property of interconnected processing units. Neural networks initially faced considerable resistance following the perceptron controversy, during which influential critiques emphasized the limitations of early single-layer architectures (Olazaran, 1996). As a result, research funding and institutional support declined, delaying progress in connectionist approaches. Later advances in multilayer networks and backpropagation demonstrated that many of these limitations were not fundamental flaws but rather consequences of limited computational power. This highlighted a recurring pattern in the history of AI, which is that advances in model capability are directly related to advances in hardware. Modern LLMs represent the aggregation of this co-evolution, due to their

large parameter counts and reliance on numerical computation show a departure from earlier AI systems, making energy consumption a leading concern across the entire industry. As these models continue to scale in size and complexity, the relationship between computational advancement and environmental cost has become an increasingly important consideration in AI research.

Energy Consumption and Environmental Impact of AI Systems

Current research emphasizes that AI energy consumption has become a sizable component of rising electricity demand. AI data centers already account for a significant share of worldwide energy use, with AI workloads among the fastest-growing contributors (Kang & Park, 2025). Although training LLMs is highly energy-intensive, researchers agree that inference represents the long-term energy cost, as models are continuously deployed at scale (Daniel et al., 2024). Studies estimating per-query energy use suggest that LLM interactions can consume more electricity than web searches, especially with longer or more complex prompts and outputs (You, 2025). While individual queries have minimal impact, their cumulative effect across millions of users creates significant pressure on energy infrastructure. This has shifted attention from training costs toward deployment efficiency. Beyond energy consumption, AI infrastructure also relies heavily on water for cooling, linking AI expansion to regional water scarcity and ecological stress. These findings show that AI sustainability depends not only on algorithms but also on infrastructure design, including cooling systems, power delivery, and hardware efficiency. However, assessing AI's environmental impact remains challenging, as major AI companies rarely disclose hardware configurations, utilization rates, or power consumption, limiting transparency and independent evaluation (Daniel et al., 2024).

The Role of Hardware Architecture in Energy and Power Efficiency

Different processors like CPUs, GPUs, and specialized accelerators tend to show very different trends in power consumption based on workload intensity. GPUs, for example, achieve high throughput through parallelism, but they tend to draw more power as the workload intensity rises (Kang & Park, 2025). Specialized accelerators, on the other hand, have been shown to offer more stable efficiency due to their more specialized designs, and high-performance computing research provides further insight into these dynamics, showing that GPU-enabled systems deliver improved performance per watt compared to traditional CPU-based systems (Schalkwijk et al., 2012). However, these results are still very much dependent on the system configuration, workload specifications, and thermal efficiency. As a result, efficiency is not an inherent property of a specific processor, but rather something that results from the interaction between hardware and workload. High-performance computing environments also face similar issues related to power density and heat dissipation. Increased frequency scaling and dense computation raise thermal output, reducing both efficiency and lifespan (Kelechi et al., 2020). Although AI-based systems have indeed shown some potential reductions in cooling energy, these solutions are often hardware-specific and cannot be generalized across different deployment contexts.

Analytical Insights from Embedded, Edge, and Distributed Computing

Ongoing research on embedded and edge computing further shows the importance of hardware efficiency analysis. Processing data locally on a machine can reduce latency and energy overhead that would result from cloud communication with a server, but these benefits are very dependent on the efficiency of the

hardware (Kang & Park, 2025). Studies demonstrate that co-designing hardware and software simultaneously can produce significant energy savings. However, most embedded-system research focuses on relatively smaller models performing limited tasks that are often the same. These findings do not directly translate to LLMs, which have larger parameter counts, greater memory demands, and more complex inference behavior. As a result, existing efficiency insights from embedded systems provide limited guidance for understanding LLM deployment on general-purpose machines. Distributed computing research similarly emphasizes system-level decisions such as scheduling and workload placement. While these studies provide useful conceptual frameworks, they are often too abstract to consider the specific hardware characteristics of individual machines, which limit their applicability to real-world scenarios.

Algorithmic Optimization and Hardware Constraints

Algorithm-based approaches to reducing energy consumption have also been deeply explored, with learning-based optimization and scheduling techniques having been shown to lower energy use in IoT networks and distributed systems without any noticeable performance drop (Balkhande et al., 2023). Additionally, multi-objective optimization methods further assert how balancing performance and energy usage can improve overall efficiency in cloud computing environments (Choudhary & Perinpanayagam, 2022). Despite these advances, however, software-level optimization continues to be constrained by the physical hardware powering it. Causes like inefficient processor design, thermal management, or power scaling can offset the gains achieved through algorithmic improvements. Due to this, understanding the specifications of the host machine remains crucial for evaluating real-world energy efficiency, especially for computationally intensive workloads.

Methods for Measuring AI Energy Consumption

Although quantifying the energy dissipated by a host machine during an AI workload presents methodological challenges, modern operating systems provide accessible mechanisms for this measurement. However, with most operating systems (Windows, macOS, and virtually every Linux distribution included), it is easy to get the amount of CPU utilization as a percentage of processing capacity in use, though this does not capture watts, through the built-in activity monitoring control panel software on all OSes. Additionally, each operating system had various commands (in both Bash and PowerShell) that could be executed that generated actual power consumption reports, and by running these commands, the amount of power P used by the host machine over a set interval can be retrieved, with the simple equation of:

$$\Delta E = \int_{t_1}^{t_2} P \, dt$$

Formula 1: Integral of power gives energy

Where P is the power, t represents time, and E denotes the amount of energy, the amount of energy dissipated by the LLM can then be calculated.

Synthesis and Identified Research Gap

Collectively, existing literature in the field shows that AI energy consumption is shaped by a complex interaction between model design, algorithmic optimization, and hardware architecture. Environmental impact studies document the increasing energy demands of AI, while hardware-focused research demonstrates that processor type and system configuration strongly influence efficiency outcomes. Methodological work provides tools for measuring and comparing energy use across different systems. However, despite these contributions, current research does not systematically examine how different general-purpose host machines influence the energy efficiency of LLM inference. Most studies focus on cloud-scale infrastructure, embedded devices, or specialized HPC environments, leaving common deployment contexts such as personal workstations and research servers underexplored. As LLMs continue to move beyond centralized cloud platforms and into diverse computing environments, this gap becomes increasingly consequential. Addressing this limitation forms the overall basis of this research study, which aims to systematically investigate how host machine specifications and operating systems influence the energy efficiency of local LLM inference.

METHODOLOGY

This study uses an experimental approach to investigate the relationship between host machine specifications and the energy consumption of large language models (LLMs). Due to the lack of objective experimental data on how hardware configurations influence AI energy efficiency, an exploratory framework is particularly appropriate for generating new quantitative evidence in this emerging area of research. The exploratory nature of the study allows for a flexible yet systematic investigation, providing preliminary insights that may guide future, more targeted studies. An experimental design was selected to allow systematic control of external conditions while varying only the independent variables of interest: the integrated testbed profile (the categorical combination of host hardware architecture and operating system software) and the model workload. This design minimizes bias and ensures that differences in energy consumption and execution performance can be attributed directly to differences in host machine configurations rather than uncontrolled environmental influences. By isolating these variables, the study aims to provide a more accurate and granular understanding of how each specification of a host machine contributes to the overall energy footprint of LLM computations.

The independent variables in this study are the properties of the specific host machine like the operating system, CPU, GPU, and amount of RAM. Operating systems tested included Windows 11 Home, macOS 26 Tahoe, and Debian Linux 13. These specifications were varied across experimental trials, and each machine’s configuration was documented in detail to ensure accurate comparison, as thorough documentation of hardware is essential to ensure replicability and to allow future researchers to interpret results in the context of specific machine characteristics. The dependent variable is the amount of power dissipated by each host machine during LLM execution. Energy consumption was measured in Watts (W) and converted into Joules (J) using operating-system-native power monitoring utilities accessed via each system's command-line interface: powercfg (via PowerShell) on Windows 11, powermetrics (via Terminal) on macOS 26, and powertop on Debian Linux 13. Execution time, recorded in seconds, served

as an additional performance indicator, providing a complementary measure of efficiency that accounts for the speed at which computational tasks are completed. Both metrics together offer a comprehensive view of machine performance and energy efficiency.

This study uses a quantitative data structure, with the primary quantitative data consisting of power consumption measurements before, during, and after execution, total energy dissipated during each trial, and execution time logged to the nearest millisecond using Python's built-in time library. These numerical measurements provide an objective basis for evaluating how different hardware configurations affect LLM efficiency and allow for precise statistical analysis. In order to ensure experimental integrity, external environmental factors are maintained as controlled constraints (constants). These constants include using identical prompt sequences across all trials, executing tests from a freshly booted system state cleared of non-essential background processes, maintaining a constant ambient room temperature to prevent thermal throttling, and neutralizing display panel power draws by fixing screen brightness. Qualitative documentation also supports transparency in the experimental process, ensuring that other researchers can successfully replicate or extend the study with a clear understanding of the equipment and software being used.

Data is collected programmatically using a custom, Python benchmarking framework featuring a thread-isolated logging process (PowerMonitor) that interfaces directly with low-overhead, kernel-level profiling utilities to eliminate the high resource overhead of graphical interfaces. This background execution loop maintains a negligible processing profile to ensure all recorded power fluctuations stem entirely from underlying LLM tensor calculations rather than the auditing software itself, while workload uniformity is enforced by deploying identical local inference models via an automated Ollama engine managed by the Python OpenAI pip library wrapper. To strictly restrict environmental variables, ambient room temperature is verified using a digital thermometer and display panel power draws are neutralized by fixing screen brightness via a luminance meter, effectively removing any external thermal or display-based confounding variables to isolate the core hardware-software interactions and make the results more representative. In addition, all computers were connected to an Ethernet cable to minimize any connectivity and latency issues during runtime.

Each host machine was first assigned a unique experimental identification code dependent on its operating system. This labeling system ensures accurate tracking and replicability of results and allows for clear referencing in the data analysis phase. Each computer was then completely wiped and had its OS freshly reinstalled onto it before being restarted to clear background processes and return performance to baseline. Before each trial, environmental conditions were standardized by recording room temperature and adjusting screen brightness to consistent levels using the luminance meter. Standardization of these conditions is essential because even minor environmental differences can subtly affect CPU and GPU thermal throttling, which in turn impacts energy consumption. To ensure that background system overhead did not introduce bias or variable hardware resource fluctuations during testing, all non-essential operating system services and background applications were terminated prior to the execution of the Python benchmarking script, returning the host environments to a standardized, clean baseline. The Python benchmarking script was then executed on each machine under identical conditions. Running all

1000 trials under uniform conditions helps ensure that the obtained experimental results are both fair and comparable across all host configurations.

Each computer ran all 3 models on it one at a time, with all runs using the same set of 1000 prompts. All 1,000 prompts were written manually by the researcher, rather than drawn from an existing dataset, to ensure consistent difficulty scaling across tiers and to avoid overlap with data the tested models may have encountered during training. The prompts are designed to be sloped in terms of computational intensity, with the first 250 being of easy difficulty(1-word answers, basic math, etc), the next 500 being of medium complexity (everyday google searches, upper-level math, paragraph-length responses, etc), and the final 250 being of intense difficulty (essay writing, complex analysis, etc). This approach allows the AI models to be fully stressed to their limits, revealing how they hold up under sustained pressure and the role the host machine plays in impacting model performance. For a list of example prompts of each difficulty, see the Appendix.

During each run, power consumption was recorded before execution, during execution, and after execution, along with total execution time in seconds. Each host machine completed multiple trials, with cooling periods between runs to prevent overheating or residual performance strain. Cooling periods are included to maintain the integrity of energy measurements, as sustained high temperatures could otherwise skew power consumption results. All results were then compiled into a structured dataset containing both quantitative performance measurements and qualitative hardware documentation. The dataset served as the foundation for subsequent statistical analysis and was made completely transparent to support verification and reproducibility. The final data produced was in the form of 9 JSON files(3 computers and 3 models running on each computer so 9 in total), with each JSON file containing the information about the computer, the model being run, as well as the results for all 1000 prompts (for an example result file, see the Appendix).

After the data collection process is complete, all host machine specifications, execution times, and energy consumption values were organized into a set of 9 JSON files (3 computers times 3 models per computer) to ensure both accurate organization and handling of experimental results. This allowed for efficient sorting and accessing of machine-specific data, facilitating detailed comparative analysis. A power regression, fitted via nonlinear least squares, and an ANOVA analysis were then conducted with throughput on the x-axis and energy consumption on the y-axis to identify any patterns between performance efficiency and power dissipation across hardware configurations. By comparing these trends across various OSs and machine specifications, this analysis highlighted which host configurations produced the most energy-efficient LLM execution. Results were presented using tables to demonstrate how specific CPUs, GPUs, RAM capacities, and operating systems influence AI workload energy demands, offering actionable insights for optimizing hardware selection for large-scale AI applications.

This study also acknowledges the ethical implications associated with energy-intensive AI research. Large language models can consume significant power, which contributes significantly to environmental impacts such as increased electricity demand and carbon emissions. By systematically measuring energy consumption across different hardware configurations, this study aims to promote more sustainable and

responsible use of AI technologies. All benchmarking was conducted in a controlled manner to avoid unnecessary energy waste, and results are reported transparently to inform other researchers about the relative efficiency of different host configurations. Additionally, careful documentation of methods and machine specifications ensures replicability and reduces the likelihood of misinterpretation, supporting ethical standards of transparency and accountability in experimental research. The study does not involve human subjects or sensitive personal data, minimizing privacy concerns.

In summary, This study uses a controlled experimental design to examine how host machine specifications can affect the energy consumption and performance of on-device large language models. By measuring power usage, execution time, and hardware configurations under standardized conditions, it provides reliable and replicable data to guide energy-efficient AI deployment. The findings establish an essential benchmark for optimizing computer hardware choices, supporting more sustainable and responsible AI practices as models continue to scale.

RESULTS

The data analysis for this research mainly focuses on the "energy-per-token" metric as the primary indicator of efficiency, alongside power draw and execution time across the three tested LLMs. The results offer a detailed view of the energy-performance trade-offs inherent in local Large Language Model (LLM) inference. By normalizing the telemetry into a Joules-per-Token (J/t) metric, the analysis bypasses superficial power consumption limits to isolate the true cost of each generated word across scales.

Contrary to expectations that larger models would demand exponentially higher running wattage, the average power draw across the macOS testbed remained remarkably constrained, fluctuating tightly around 5.3W. This reveals that on integrated ARM architectures utilizing a unified memory matrix, the true bottleneck for energy efficiency is not peak power draw, but rather the Wall Duration required to complete a tensor operation.

An interesting phenomenon observed consistently across all the models and operating systems was the “Inference Spike.” During the initial prompt processing phase, known as the pre-fill operation, system power draw rose sharply and peaked as the framework parsed prompt tokens, allocated memory, and loaded model weights into cache. Once the system transitioned into the decoding phase, where tokens are generated sequentially, power consumption dropped into a more stable range. Although decoding remains demanding, it distributes computation more evenly over time, resulting in steadier behavior. For example, on macOS, Gemma maintained a steady draw of approximately 6.0W during generation, while Mistral, despite having nearly three times the parameters, exhibited a similar steady-state draw of about 4.76W under comparable conditions. The key difference was not instantaneous power usage but duration, as Mistral remained in this active state for up to 125 percent longer to produce outputs of matching length, leading to a higher energy footprint despite similar real-time wattage.

The complete data for each machine can be found on the following page in Tables I-IV:

Table I - Machine Hardware Specifications

	CPU	GPU	Memory	Storage	TDP
Windows 11	Intel Ultra 7 265K	Intel Xe-LPG	64GB DDR5	1TB	125 W
macOS 26	Apple M4	Apple 8-core	16GB LPDDR5	256GB	~40 W(SoC)
Debian 13	Intel i5-13600K	Intel UHD 770	8GB DDR4	256GB	125 W

Table II - Windows 11 Average Data

	Easy time	Medium time	Hard time	Energy per token	Average power draw	Tokens per second
Gemma2 2B	2.7610	29.8620	52.7450	2.1080	36.2900	16.3800
Llama3.2 3B	3.8470	35.146	62.0890	3.0693	39.5200	12.2600
Mistral 7B	9.3390	65.7550	120.309	6.9894	44.5300	6.0300

Table III - macOS 26 Tahoe Average Data

	Easy time	Medium time	Hard time	Energy per token	Average power draw	Tokens per second
Gemma2 2B	1.2890	12.2880	19.6230	0.1603	6.0500	42.4800
Llama3.2 3B	1.9010	14.0530	23.1050	0.1739	5.0900	32.0412
Mistral 7B	4.5720	26.7380	44.4460	0.3163	4.7600	15.6100

Table IV - Debian Linux 13 Average Data

	Easy time	Medium time	Hard time	Energy per token	Average power draw	Tokens per second
Gemma2 2B	3.8560	41.8450	74.2990	1.2618	15.4900	11.6300
Llama3.2 3B	5.4810	50.1740	91.2020	1.8795	16.8300	8.4600
Mistral 7B	12.3660	84.6560	157.011	3.7139	18.1600	4.6500

DISCUSSION

The overall efficiency of each model was heavily influenced by its parameter size and the resulting generation speed:

- **Gemma 2:2b:** This model was consistently the most efficient across all 3 host environments. As shown in Table III, it achieved an exceptional efficiency profile on macOS 26 Tahoe of 0.1603 J/t; per Table II, it maintained a moderate footprint of 2.1080 J/t on Windows 11; and per Table IV, it required 1.2618 J/t on Debian Linux 13.

- **Llama 3.2:3b:** Serving as the middle ground, this model demonstrated mid-tier energy costs across the board. It required a true summary mean of 0.1739 J/t on macOS 26 Tahoe (Table III), 3.0693 J/t on Windows 11 (Table II), and 1.8795 J/t on Debian Linux 13 (Table IV). Its tokens-per-second rate was lower than Gemma's on all platforms, translating directly to higher cumulative energy requirements.

- **Mistral 7b:** As the largest model tested, Mistral 7B was very energy intensive, with its cross-platform summary average energy cost reaching 3.6732 J/t, driven by low throughput speeds, particularly on Linux(Table IV) where it managed just 4.65 tokens per second, causing a severe power drain.

A critical discovery in the dataset is the non-linear relationship between prompt complexity (Easy vs. Medium vs. Hard) and localized energy cost. For example, under Windows 11, the energy-per-token metric scales from 2.0538 J/t for easy tasks up to 2.2076 J/t for hard tasks when running Gemma 2:2B (per-difficulty breakdown; see Appendix). Similarly, on Debian Linux, Mistral 7B scales from 3.6216 J/t (Easy) to an expansive 3.8304 J/t (Hard) (per-difficulty breakdown; see Appendix). This demonstrates that sustained, complex generation tasks amplify the systemic strain on high-overhead runtime environments, causing a sharp, non-linear increase in total energy costs compared to short-form baseline executions.

While average power remained within predictable operating bands for each host machine, Wall Duration acted as the primary multiplier for total energy consumption. In the Mistral trials, this created a clear "latency wall". As model parameter size increased, the Tokens Per Second (TPS) dropped from Gemma's 23.50 TPS to Mistral's average of 8.76 TPS(Tables II-IV). Because the system must keep the processing cores and memory bus in an active, high-power state for the complete duration of the generation cycle, every second of latency adds directly to the total Joules consumed. For example, during sustained trials where Mistral held the hardware in an active generation state for extended durations, energy accumulated rapidly. In contrast, Llama completed similar tasks in a fraction of the time, reducing total energy draw, and this confirms that in the context of edge deployments, the most sustainable model choice is not necessarily the one with the lowest instantaneous power signature, but the one that achieves the highest operational throughput among comparably-sized models.

During the extended "Hard" difficulty blocks for Mistral 7B, often lasting more than 30 seconds, the automated background script continuously monitored power draw for any signs of thermal throttling or longer-horizon efficiency drift. On the macOS 26 Tahoe testbed, the power draw remained remarkably stable, staying locked around 5.3W even toward the end of long generation cycles, with only minor short-term fluctuations that did not indicate any sustained upward trend. This steadiness suggests that the integrated cooling behavior and tightly managed power states of the ARM-based architecture help preserve consistent performance under continuous inference load, avoiding the gradual performance variability sometimes seen under extended stress. In contrast, on the Windows 11 environment, a more noticeable upward drift in power draw was observed during sustained workloads, implying that thermal management dynamics in higher-overhead desktop configurations may introduce an additional "heat tax" over time, where accumulating thermal load influences voltage-frequency scaling behavior and incrementally reduces Joules-per-token efficiency as execution continues.

Beyond CPU architecture and generation velocity, the memory and thermal design specifications listed in Table I offer additional insight into the divergent energy profiles observed across host machines. Although the Windows 11 (Intel Ultra 7 265K) and Debian Linux 13 (Intel i5-13600K) testbeds share an identical 125W TDP classification, their measured average power draws differed substantially, ranging from 36.0W-45.0W on Windows versus 15.0W-19.0W on Debian (Table II vs. Table IV). This divergence indicates that TDP functions primarily as an upper bound on a chip's thermal design headroom rather than as a reliable predictor of sustained real-world power draw; the actual draw instead appears shaped by how comfortably the inference workload fits within the available memory subsystem. The Windows machine's 64GB of DDR5 memory provided substantial headroom beyond the requirements of even the largest tested model (Mistral 7B), allowing model weights and active context to remain fully resident in fast memory throughout generation. In contrast, the Debian testbed's 8GB of slower DDR4 memory left considerably less margin, particularly for Mistral 7B, whose parameter footprint alone represents a meaningful fraction of total available RAM once the OS, Ollama runtime, and inference context are accounted for. This constrained headroom likely forced more frequent memory paging and reduced effective memory bandwidth during sustained generation, modestly increasing Wall Duration relative to Windows (Table IV), yet Debian's far lower sustained power draw more than compensated, driving the substantially lower energy-per-token values observed on Debian (3.7139 J/t for Mistral 7B, versus 6.9894 J/t on Windows; Table II vs. Table IV).

The macOS 26 testbed presents a different pattern. Despite having only 16GB of memory, less than either x64 competitor, it achieved the lowest energy cost per token by a wide margin (Table III). This result is best explained not by memory capacity but by memory architecture: the Apple M4's unified memory design allows the CPU, GPU, and neural processing components to access a single shared memory pool without the copying overhead or separate address spaces typical of discrete CPU/GPU configurations. Combined with the SoC's integrated power management, this architecture appears to keep the chip within a narrow, low-power operating envelope regardless of model size, reflected in the remarkably stable 4.7W-6.1W draw recorded across all three tested models (Table III). This suggests that memory architecture and system-on-chip integration can be a stronger determinant of edge inference efficiency than either raw memory capacity or nominal TDP alone.

Storage capacity and type (Table 1) showed no clear direct relationship to the energy outcomes in this study. Because Ollama loads model weights into memory once at the start of a session, storage primarily affects initial load latency rather than the sustained generation-phase power draw measured here. As this study did not isolate storage read/write speed as an independent variable, we treat any storage-related effect as a limitation rather than a finding; future work could disentangle model-load energy from generation-phase energy to test whether faster storage meaningfully reduces the "Inference Spike" observed during pre-fill.

STATISTICAL ANALYSIS

In order to evaluate the overall effects of host operating systems and model architectures on the energy cost of a specific LLM inference, a Two-Way Analysis of Variance (ANOVA) test was conducted, with the significance level being set at $\alpha = 0.05$. The dependent variable was the energy efficiency, quantified by the energy per token (Joules per token, J/t), in which a lower value indicates a stronger overall hardware-software efficiency profile. The independent variables included the host operating system and the model architecture tier (Gemma 2:2B, Llama 3.2:3B, and Mistral 7B). The omnibus Two-Way ANOVA revealed highly significant main effects for both independent variables, as well as a statistically significant first-order interaction effect, thereby rejecting the null hypothesis H_0 that energy consumption is constant across a variety of local edge configurations ($p < \alpha$). Post-hoc analysis utilizing Tukey's Honestly Significant Difference (HSD) test was then applied to further isolate and identify specific cross-platform and cross-model variations within the dataset.

The post-hoc pairwise comparisons further showed that model scale dominates the absolute energy investment required during inference. When comparing cross-platform combined metrics across all three operating systems, the average energy cost unexpectedly scaled non-linearly with parameter density. Gemma 2:2B established the baseline efficiency at 1.1767 J/t, followed by Llama 3.2:3B at 1.7076 J/t, while Mistral 7B required an exponentially higher cross-platform average token expenditure of 3.6732 J/t. This progression demonstrates a substantial increase in energy demand as model complexity increases, despite all models being evaluated under the same experimental methodology. Crucially, within the parallel x64 environments, Windows 11 was shown to be considerably less efficient than Debian Linux 13. For example, during the execution of Llama 3.2:3B, the Windows 11 environment maintained a mean consumption profile of 3.0693 J/t, which represents an inferior efficiency compared to the 1.8795 J/t required by the Debian Linux environment (Table II vs. Table IV). This observed difference widens substantially as parameter size continues to scale. Under the Mistral 7B workload, Windows 11 drew a mean of 6.9894 J/t, whereas Debian Linux 13 dropped to 3.7139 J/t (Table II vs. Table IV), translating to a 46.86% decrease in the energy required to generate an identical amount of standard UTF-8 text under otherwise comparable experimental conditions. Collectively, these results reinforce the presence of statistically significant differences in energy efficiency across both operating systems and model architectures, with the magnitude of these differences becoming increasingly more pronounced as model size increases across the evaluated language model architectures.

To frame the exact significance thresholds of the primary experimental observations, both the individual main effects and interaction data are formally integrated into the statistical analysis as seen below:

- Both the parameter scale and structural complexity of the models exerted a statistically significant effect on energy requirements, which was evaluated as $F(2, 8991) = 24747.12$, with a probability of $p \ll 0.001$.

- The host software environment and its underlying hardware abstraction framework both demonstrated a statistically significant impact on execution efficiency, evaluated as $F(2, 8991) = 52814.21$, with an empirical probability of $p \ll 0.001$.

- A significant first-order interaction effect between the model tier and the software environment was mathematically validated at an empirically determined probability of $p \ll 0.001$, proving that software runtime constraints do not scale linearly with model parameter boundaries.

Following the omnibus framework, a Spearman rank correlation coefficient (ρ) was computed to map the monotonic relationship between token generation throughput (Tokens per Second) and the resulting normalized energy footprint (Joules per Token) across the integrated dataset. The analysis yielded a strong negative correlation of:

$$\rho = -0.8267$$

Figure I - Spearman rank correlation coefficient

This index asserts that local edge inference efficiency is indeed dependent on the execution velocity. Due to traditional x64 host architectures drawing a high and unoptimized baseline wattage when maintaining active hardware computation states (36.0W–45.0W on Windows 11 and between 15.0W and 19.0W on Linux)(see Average Power Draw columns, Tables II and IV), any processing latency or execution delay directly exponentiates the total time that the system must remain locked in this power-heavy state as a result. To quantitatively model this behavior, a power regression was fitted via nonlinear least squares using Token Throughput (x) as the independent predictor of Joules per Token (y) across the integrated dataset. This calculation produced the following regression model:

$$\hat{y} = 27.21391 \cdot x^{-1.04689}$$

Figure II - Power regression equation

The negative nature of this relationship shows that token throughput is indeed a primary operational determinant of edge energy efficiency, especially in sustained inference workloads where runtime duration is just as important as instantaneous power draw. Because x64 operating systems, particularly Windows 11, maintain a relatively high baseline power overhead regardless of execution speed, any reduction in generation velocity forces the system to remain in an active, power-intensive computational

state for a longer period per token, effectively extending the energy "exposure window" of each inference step. This extended runtime directly amplifies total energy consumption even when instantaneous wattage remains relatively stable, as the cumulative area under the power-time curve increases linearly with execution time. Therefore, architectural and software-level optimizations that improve generation velocity, whether it be through better kernel scheduling, optimized attention implementations, or reduced memory bottlenecks, tend to correlate in a proportional decrease in cumulative energy use and, by extension, carbon footprint in local edge deployments. Consequently, minimizing idle computational periods, reducing sub-optimal runtime intervals, and improving scheduling efficiency become critical design priorities for developers targeting constrained, battery-powered, or otherwise resource-limited environments where sustained efficiency matters more than peak throughput alone.

CONCLUSION

This research provided a systematic analysis of the energy efficiency of on-device LLM inference across varying operating systems and model architectures. By precisely quantifying the "energy-per-token" cost, the study connects raw performance metrics to environmental sustainability in the modern era of artificial intelligence. The findings also confirm that while hardware specifications provide the foundation for power potential, the operating system serves as a critical mediator of actual energy consumption.

The ANOVA results ($p < \alpha$) statistically validate this claim as well. It is worth noting, however, that within parallel x64 environments, the Windows/Ultra-7 265K testbed showed higher architectural overhead and higher throughput than the Debian/i5-13600K testbed, resulting in an increased cumulative power drain. For example, during execution of Llama, the Windows environment maintained a lower mean efficiency of 3.0693 J/token compared to the 1.8795 J/token required by the Linux environment(Table II; Table IV), further highlighting how background resource management and execution velocity can vastly influence the cumulative energy footprint. Additionally, the data highlights a disparity in efficiency based on the model parameter size. Gemma emerged as the most sustainable choice for short-form and general-purpose tasks with a baseline mean below 0.2 J/token on the integrated ARM architecture of macOS(Table III). In contrast, Mistral's increased computational complexity and reduced throughput led to a higher energy investment, culminating in a cross-platform average energy cost of 3.6732 J/token and scaling up to 6.9894 J/token on Windows 11 (Table II), suggesting that for local deployment, the "larger is better" approach to AI must be balanced against the exponential increase in energy demand as opposed to a more traditional linear model regression.

In conclusion, local AI offers a path toward data privacy and reduced latency, but it is not exempt from environmental responsibility. Optimizing the synergy between the LLM, token throughput, and the host OS, ensures that the advancement of AI does not come at an unsustainable cost to our global energy infrastructure. Future work should extend this benchmarking framework to systems with dedicated accelerators, such as NPUs, discrete GPUs, or edge AI chips (e.g., NVIDIA Jetson, Google Coral), to test whether offloading inference away from the CPU reduces the host-OS-driven energy differences identified in this study.

REFERENCES

[1] Balkhande, B., Ghule, G., Meshram, V. H., Anandpawar, W. N., Gaikwad, V. S., & Ranjan, N. (2023). Artificial Intelligence Driven Power Optimization in IOT-Enabled Wireless Sensor Networks. Journal of Electrical Systems, 19(2), 38–46.

[2] Choudhary, R., & Perinpanayagam, S. (2022). Applications of Virtual Machine Using Multi-Objective Optimization Scheduling Algorithm for Improving CPU Utilization and Energy Efficiency in Cloud Computing. Energies, 15(23), 9164. https://doi.org/10.3390/en15239164

[3] Copeland, B. (2025, July 30). history of artificial intelligence (AI). Encyclopedia Britannica. https://www.britannica.com/science/history-of-artificial-intelligence

[4] Daniel, C., Gehin, J., Laurin-Kovitz, K., Morreale, B., Stevens, R., & Tumas, W. (2024). ADVANCED RESEARCH DIRECTIONS ON AI FOR ENERGY Report on Winter 2023 Workshops. https://www.anl.gov/

[5] Dreyfus, H. L., & Dreyfus, S. E. (1988). Making a Mind versus Modeling the Brain: Artificial Intelligence Back at a Branchpoint. Daedalus, 117(1), 15–43. http://www.jstor.org/stable/20025137

[6] Kang, M., & Park, M. (2025). Power Estimation and Energy Efficiency of AI Accelerators on Embedded Systems. Energies, 18(14), 3840.

[7] Kelechi, A. H., Alsharif, M. H., Bameyi, O. J., Ezra, P. J., Joseph, I. K., Atayero, A.-A., Geem, Z. W., & Hong, J. (2020). Artificial Intelligence: An Energy Efficiency Tool for Enhanced High performance computing. Symmetry, 12(6), 1029.

[8] Olazaran, M. (1996). A Sociological Study of the Official History of the Perceptrons Controversy. Social Studies of Science, 26(3), 611–659.

[9] Schalkwijk, J., Griffith, E. J., Post, F. H., & Jonker, H. J. J. (2012). High-Performance Simulations of Turbulent Clouds on a Desktop PC: Exploiting the GPU. Bulletin of the American Meteorological Society, 93(3), 307–314.

[10] You, J. (2025, February 7). How much energy does ChatGPT use? Epoch AI.

APPENDIX

To ensure transparency, reproducibility, and accessibility of the research workflow, all code, data artifacts, and supporting materials associated with this study have been made publicly available in a dedicated GitHub repository.

The repository includes the full Python codebase used for data processing, analysis, and generation of results, along with all JSON files produced throughout experimentation. These JSON outputs capture both intermediate and final computational states, enabling readers to trace the workflow and reproduce the results under the same experimental conditions described in this paper.

In addition, the repository contains a structured collection of over 1,000 AI prompts developed and used during the course of this research. These prompts were systematically designed to evaluate model behavior across diverse tasks and conditions, and are provided in full to support methodological transparency, reproducibility, and potential extension in future studies.

To further support usability, the repository includes a comprehensive README.md file. This document provides an overview of the project structure, setup instructions, dependencies, and usage guidelines for running the code and reproducing the results. It also outlines the intended workflow and describes how each major component of the repository contributes to the overall research pipeline. The README serves as the primary entry point for researchers or practitioners seeking to understand or extend the project.

The entire repository is released under the Apache License 2.0. This permissive open-source license allows for use, modification, and distribution of the code, both in academic and commercial contexts, provided that proper attribution is maintained and the terms of the license are followed. The license also explicitly grants patent rights from contributors to users, supporting safer adoption of the software in downstream applications while preserving academic credit and provenance.

All materials are organized into clearly labeled directories in order to facilitate navigation and reproducibility. Readers interested in replicating results, auditing the methodology, or building upon this work are encouraged to consult the repository in full.

The complete project repository is available at https://github.com/Speedstrike/ai-benchmarks/