Green Llama: A Tool for Monitoring Energy Consumption and Sustainability in Local LLMs

Philippe A. Henry *ID : 4678605*
Anyan Huang *ID : 6168930*
Yongcheng Huang *ID : 5560950*
Yiming Chen *ID : 5541786*

***Abstract*—Large language models (LLMs) have revolutionized artificial intelligence by delivering unprecedented performance in various applications. However, their substantial computational demands have raised critical concerns regarding high energy consumption and environmental impact, especially during inference on local machines. This paper introduces Green Llama, an innovative command-line tool designed to monitor and analyze the energy usage of LLMs across CPU, GPU, and RAM components. By providing prompt-based estimation of power consumption and translating these metrics into carbon emissions based on regional energy profiles, Green Llama bridges the gap between performance benchmarking and sustainability assessment. With integrated features such as mid-session summaries, detailed reporting, and comprehensive benchmark testing, the tool empowers developers, researchers, and organizations to monitor energy consumption, reduce operational costs, and make environmentally responsible deployment decisions. For more information, please refer to the repository of Green Llama.**

***Index Terms*—Large Language Model (LLM), Energy Monitoring, Carbon Emissions, Sustainability**

I. Introduction

Generative Artificial Intelligence (AI) has experienced rapid advancements in recent years, with Large Language Models (LLMs) such as GPT, Gemini, and Claude revolutionizing natural language processing (NLP) tasks including text generation, summarization, and translation [12]. While these models deliver exceptional performance, their deployment requires substantial computational resources, leading to increased energy consumption and significant environmental impacts [5], [9].

In response to these challenges, tools like Ollama have emerged to address data privacy and resource management concerns by enabling users to download, store, and execute LLMs on local machines without relying on cloud-based services [6]. Local execution provides improved data security, reduced dependence on external servers, and greater control over hardware resources. However, running large-scale models locally also introduces new challenges in energy efficiency and sustainability, prompting users to consider an additional criterion when selecting a model: its environmental footprint [8].

In response to these challenges, we propose Green Llama, a command-line tool designed to monitor and analyze the energy consumption and performance of LLMs managed through the Ollama tool. Green Llama tracks precise resource utilization across CPU, GPU, and RAM, translating these measurements into carbon emission estimates according to regional energy profiles. This capability enables developers, researchers, and organizations to optimize decision-making, effectively balancing computational performance with environmental responsibility. Inspired by related work such as EnergyBridge [11], which provides comprehensive insights into energy consumption patterns, our approach further integrates performance evaluation with environmental impact assessment. **Our contributions** are summarized as follows:

- We propose Green Llama, a command-line tool specifically designed to monitor and analyze the energy consumption and carbon emissions of large language models (LLMs) running locally.
- Green Llama provides detailed, prompt-based estimations of CPU, GPU, and RAM energy usage, translates these metrics into carbon emission estimates using regional energy profiles, and offers integrated benchmarking, interactive mid-session summaries, and comprehensive reporting features.
- We make the tool publicly available as an open-source project, along with detailed documentation to facilitate adoption and further research.

II. Background and Problem Identification

The rapid expansion of large language models (LLMs) across various applications has significantly increased the demand for computational resources, raising critical sustainability concerns due to their considerable energy consumption and carbon emissions [10]. Training state-of-the-art LLMs can consume massive amounts of electricity, resulting in substantial environmental footprints. For instance, Strubell et al. estimated that training a single transformer-based language model emits approximately 284 tons of CO_2, roughly equivalent to the lifetime emissions of five cars [10]. Even during inference, the operational energy demands remain significant, as inference involves billions of computations per prompt, contributing notably to electricity usage and carbon emissions [1], [8].

Despite these concerns, developers and organizations face significant challenges due to limited visibility into real-time energy usage during LLM inference, particularly in local deployments. Frameworks such as Ollama, designed for privacy and resource control by enabling local execution of LLMs [6], further amplify this visibility gap. Currently, there is no straightforward solution for monitoring energy consumption

and estimating carbon emissions in real-time during inference tasks, hindering optimization for efficiency and sustainability.

Existing benchmarking practices predominantly emphasize model performance metrics like accuracy, latency, or throughput, while frequently neglecting energy consumption and sustainability implications [4], [8]. Though some tools exist—such as CodeCarbon [3], CarbonTracker [1], and Experiment Impact Tracker [4]—these primarily target model training or post-execution analysis. None explicitly cater to real-time monitoring of energy usage and environmental impact during LLM inference on local platforms like Ollama.

Given this gap, there is an evident need for tools specifically designed to provide continuous, real-time insights into energy consumption and carbon emissions for locally deployed LLMs. Such tools would enable more sustainable and cost-effective deployment decisions, aligning with broader environmental goals and supporting the emerging paradigm of "Green AI" [8].

III. Design and Implementation

Green Llama is designed to monitor the energy consumption and environmental impact of large language models (LLMs) during runtime. As shown in figure 1, Green Llama provides users with a comprehensive understanding of how their models utilize computational resources, enabling informed decisions to optimize performance and sustainability. This section outlines the key functionalities of Green Llama, focusing on its ability to estimate metrics, provide mid-conversation summaries, generate detailed reports, conduct benchmark testing, rank models based on environmental impact, and manage model resources effectively.

Fig. 1. The Overview of Green Llama CLI Tool

A. Metric Estimation

The Metric Estimation feature is at the core of Green Llama's functionality. It allows users to monitor the energy consumption of CPU, GPU, and RAM, providing insights into the total energy usage during model inference.

1) CPU Energy Consumption (Joule): CPU energy consumption quantifies the electrical energy consumed by the processor during model inference. The energy usage is estimated based on CPU utilization percentage and core count, assuming a thermal design power (TDP) of 8 watts per core.

2) GPU Energy Consumption (Joule): GPU energy consumption tracks the graphics processing unit's energy usage during model execution. Accurate power measurements from NVIDIA GPUs are obtained using the NVIDIA Management Library.

3) RAM Energy Consumption (Joule): RAM energy consumption captures the energy footprint associated with the memory subsystem. The consumption is estimated based on the total memory size and usage percentage, assuming a power consumption rate of 3 watts per 8GB of RAM.

4) Total Energy Consumption (Joule): Total energy consumption provides a comprehensive measurement by aggregating energy usage across all monitored hardware components. This metric offers a holistic view of the system's overall energy requirements during model operation.

5) Carbon Emissions (gCO_2): Carbon emissions metrics translate total energy consumption into environmental impact by applying region-specific carbon intensity factors, thus quantifying the carbon footprint of model execution in terms of carbon dioxide emissions.

The metric estimation example can be seen in Figure 2.

B. Mid-Conversation Metric Summaries

During interactions with the tool, users can access Mid-Conversation Metric Summaries to gain a quick overview of the performance and energy consumption of their models. This feature provides aggregated statistics, such as average energy usage, carbon emissions and so on. Users can request these summaries at any point during a conversation, allowing them to monitor trends and make adjustments as needed without interrupting their workflow.

Summary of Metrics for this Conversation

Metric	Value
Average CPU Energy (J)	597.00
Average GPU Energy (J)	0.00
Average RAM Energy (J)	307.74
Average Total Energy (J)	904.73
Average Carbon Emissions (gCO2)	0.09
Total Energy (J)	904.73
Total CO2 Emissions (gCO2)	0.09
Average Response Time (s)	59.23
Number of Prompts	1
Most Costly Prompt	write me a python code fo...
Least Costly Prompt	write me a python code fo...

Fig. 2. An Example of Metrics and Summary

C. Interactive Result Viewer Interface

As part of the Green Llama system, we implemented an interactive, browser-based report viewer to make the energy benchmarking results more accessible and interpretable. This feature was developed using **React.js** and integrates directly with the benchmarking pipeline.

The viewer supports multiple reporting modes, including benchmark results, model history summaries, and per-conversation statistics. Key features of the interface include:

- **Pie Chart Visualizations:** For each report, a pie chart shows the breakdown of total energy consumption across CPU, GPU, and RAM components.
- **Fact-Based Emission Equivalents:** Carbon emissions are contextualized through real-world analogies such as sheets of paper, kilometers walked, emails sent, and Google searches.
- **Metric Graphs:** Each energy or emission metric is visualized with a plot that summarizes its distribution across prompts (e.g., using box plots).
- **Prompt-Level Tables:** Below each graph, detailed tables provide a breakdown of energy usage per prompt, allowing for fine-grained analysis.

The viewer dynamically parses raw CSV logs and generates visual summaries without requiring manual preprocessing. It allows researchers and practitioners to interactively explore performance trends, spot anomalies, and compare models with ease.

D. Benchmark Testing

The benchmark testing module in Green Llama is designed to comprehensively evaluate model performance across different tasks. Our implementation supports three types of benchmark tests: **text generation**, **code generation**, and **chat testing**. Upon entering the benchmark mode, users are prompted to choose one of these options.

For text generation tasks, the system loads a dataset (using the Wikitext-2 raw test split) and processes a subset of the data to keep execution time reasonable. Specifically, the first two prompts are selected for the initial benchmark run. In addition to these standard prompts, a set of five novel prompts is used to assess the model's performance in creative generation scenarios. This two-tier testing approach not only evaluates the model on standard tasks but also examines its adaptability and performance under more diverse conditions.

Each prompt is processed by a measurement function that computes key metrics, including CPU usage and elapsed time. Notably, the CPU usage is normalized by dividing the raw usage by the number of available CPU cores. This normalization ensures a fair comparison across different hardware configurations. The benchmarking routine leverages Python's `multiprocessing` module and the `rich` library for interactive command-line display, which allows for real-time feedback on the progress of the benchmark and detailed metric summaries.

Inspired by the methodologies presented in benchmark studies such as [7], our implementation extends these ideas by integrating energy consumption metrics directly into the evaluation process. The referenced work provides a detailed framework for assessing both performance and efficiency of language models, and it serves as a key foundation for our approach.

After the benchmark run, the results – including prompt texts, normalized CPU usages, and execution times – are aggregated and stored. These metrics are then used to update the real-time summary display and are persistently saved in JSON and CSV formats. This persistent storage not only supports detailed post-analysis but also feeds into the model ranking functionality, where models are compared based on their average CO_2 emissions derived from these benchmark metrics.

Overall, our benchmark testing implementation provides an integrated framework that facilitates:

- **Flexible Task Selection:** Users can choose between text generation, code generation, or chat testing benchmarks.
- **Robust Metrics Collection:** Both standard and novel prompts are utilized to obtain comprehensive performance data.
- **Normalized Performance Metrics:** CPU usage is normalized based on the number of cores, ensuring consistency across varied hardware.
- **Interactive and Persistent Reporting:** Real-time summaries and persistent logging enable both immediate insights and long-term analysis.

This implementation not only enables fair and detailed comparisons among different LLMs but also supports our overall goal of linking model performance with environmental sustainability by providing accurate and actionable energy consumption data [10].

E. Ranking of Models

To further assist users in selecting sustainable models, Green Llama includes a Ranking of Models feature. Ranking of models is done based on average CO2 cost estimation per prompt, based primarily on benchmark data for a fair comparison. Given that we present three different types of benchmarks, a different table is rendered for each. Models are only added to the table if they have the selected benchmark's results. Additionally, a conversation history table is added, which ranks models based on the interactions the user has had with them, Models are added as soon as they have been utilized one time. For an example of model rankings refer to Figure 3

F. Model Management

Green Llama simplifies the process of managing models through its Model Management capabilities. Users can view a list of available models, download new ones from the Ollama repository, and handle missing models seamlessly. This ensures that users have access to the resources they need while maintaining a streamlined workflow. For an example of the possible workflow refer to Figure 4

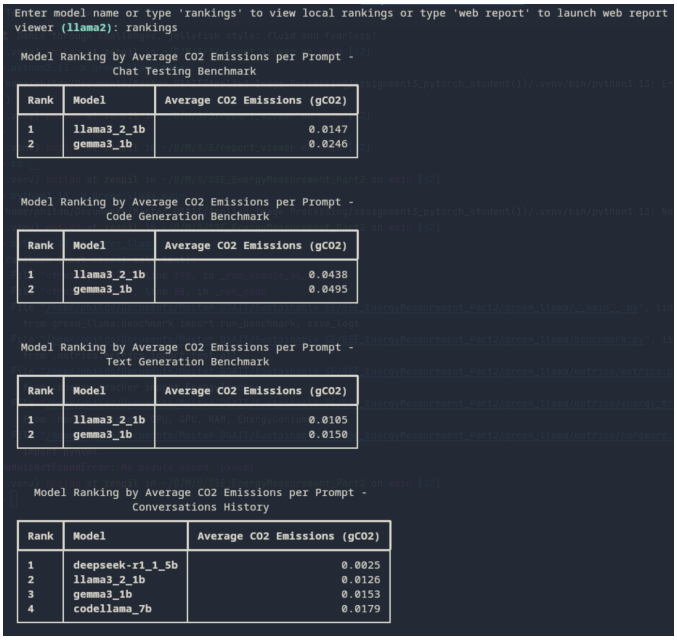

Fig. 3. An Example of Model Rankings

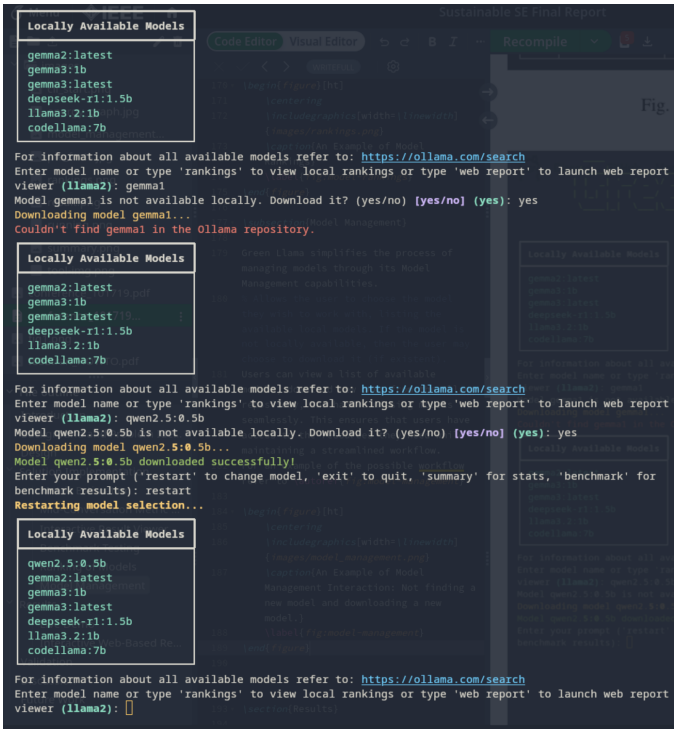

Fig. 4. An Example of Model Management Interaction: Not finding a new model and downloading a new model (which is then listed in the locally available models).

IV. Runtime Monitoring Examples

A. Live Demo

For a live presentation of all features refer to the following: Live Demo Video

B. Benchmarking

In this section, we illustrate a complete end-to-end benchmark workflow using Green Llama. The process involves three major steps: (1) choosing the benchmark mode, (2) running multiple rounds of prompts and observing intermediate results, and (3) generating a final summary of metrics.

Fig. 5. CLI interface for selecting the benchmark mode (e.g., text generation, code generation, or chat testing).

As shown in Figure 5, the user begins by launching the Green Llama CLI and is prompted to select one of three benchmark modes: **Text Generation**, **Code Generation**, or **Chat Testing**. Once the user chooses the desired mode, the system prepares a corresponding set of prompts or tasks that will be used to measure the model's performance and resource consumption.

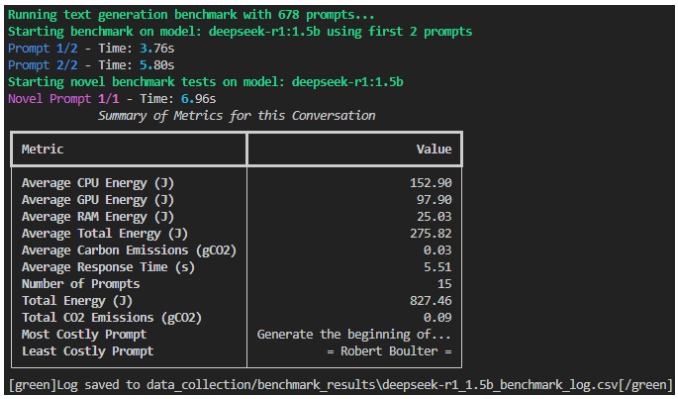

Fig. 6. Intermediate benchmark results displayed during execution.

Next, the benchmark is executed in multiple rounds (Figure 6). Each round processes a certain number of prompts, providing **intermediate results** such as:

- *Elapsed Time (s)* for each prompt
- *Normalized CPU Usage (%)* across available CPU cores
- Other metrics (e.g., GPU usage, RAM usage, energy consumption) if enabled

These intermediate summaries help users observe how the model behaves as it processes different prompts and can inform early adjustments if any issues or inefficiencies arise.

Upon completion of all benchmark rounds, Green Llama generates a **final summary report** as depicted in Figure 7. This report aggregates the measurements from each prompt and displays:

Summary of Metrics for this Conversation

Metric	Value
Average CPU Energy (J)	46.29
Average GPU Energy (J)	111.66
Average RAM Energy (J)	28.97
Average Total Energy (J)	186.92
Average Carbon Emissions (gCO2)	0.02
Average Response Time (s)	6.54
Number of Prompts	15
Total Energy (J)	560.75
Total CO2 Emissions (gCO2)	0.06
Most Costly Prompt	Generate the beginning of...
Least Costly Prompt	Robert Boulter is an Engl...

Fig. 7. Final summary of benchmark results after completion.

- *Averaged metrics* (e.g., average CPU energy, total energy consumed, carbon emissions, etc.)
- *Overall response times* and the total number of prompts tested
- *Most* and *least costly prompts*, helping to pinpoint which queries consume the most resources

With this detailed summary, developers and researchers can quickly identify performance bottlenecks, compare different models, and make informed decisions to optimize both performance and sustainability. All data from these benchmarks are also logged to JSON/CSV files, enabling further offline analysis and integration into other tools.

C. Interactive Web-Based Report Viewer

To present the results in a more intuitive and accessible way, we use the interactive report viewer described in Section III-C.

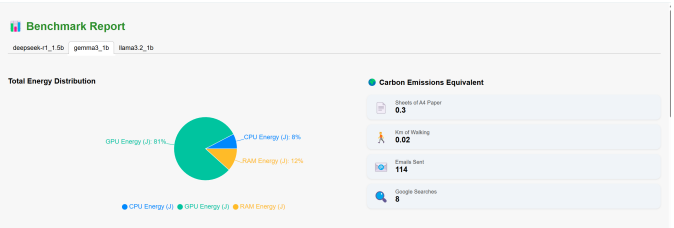

Fig. 8. Frontend showing energy distribution pie chart and equivalent carbon impact facts.

Figure 8 shows an example report layout: on the left, a pie chart summarizes the proportion of energy used by CPU, GPU, and RAM; on the right, real-world equivalents for carbon emissions (e.g., paper, walking distance, emails) help contextualize the results.

As shown in Figure 9, the viewer plots energy metrics across prompts using box plots, making it easy to identify variability and outliers. The indexed x-axis corresponds to unique prompts, each run multiple times during benchmarking.

V. Testing and Validation

To ensure the accuracy and reliability of Green Llama's energy monitoring capabilities, we conducted a series of validation tests across CPU, GPU, and RAM components.

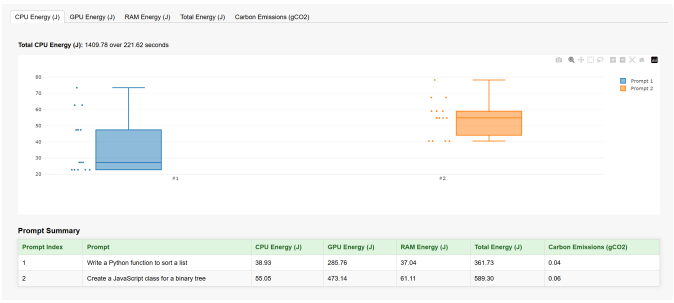

Fig. 9. Box plot of energy metrics grouped by prompt index.

These tests were designed to verify the validity of the returned values, evaluate the behavior of the system under controlled conditions using mock testing, and compare the results with an established tool, CodeCarbon.

For the validity of returned values, we implemented unit tests to confirm that the energy consumption values for CPU, GPU, and RAM are within expected ranges. For example, the CPU power measurement test ensures that the returned power value is greater than zero under normal operating conditions. Similar tests were conducted for GPU and RAM, verifying that the power and energy values are non-negative and consistent with the expected behavior of the hardware.

In the mock testing, we simulated the behavior of the energy tracking components to isolate and test specific functionalities. By mocking the EnergyTracker class, we validated that the system correctly calculates energy consumption and elapsed time without relying on actual hardware measurements. This approach allowed us to test edge cases and ensure that the system behaves predictably under various scenarios.

Finally, we performed comparative testing with CodeCarbon [2], a widely used tool for estimating energy consumption and carbon emissions. For CPU, GPU, and RAM, we compared the energy consumption values calculated by Green Llama with those provided by CodeCarbon. The results demonstrated consistency between the two tools, with minimal deviations within 0.0001. This comparison validates the accuracy of Green Llama's energy tracking algorithms and confirms its suitability for real-world applications.

These validation steps collectively ensure that Green Llama provides reliable energy consumption estimation metrics, making it a robust tool for monitoring the environmental impact of large language models.

VI. Discussion

Green Llama demonstrates that it is feasible to deliver detailed and real-time sustainability analytics for large language models running locally. Through careful engineering of metric estimators, session logging, and browser-based reporting, the tool offers both technical rigor and user-oriented accessibility.

Our implementation relies on multiple measurement strategies: GPU energy is monitored in real time via NVIDIA's `PyNVML` library, while CPU and RAM energy are estimated based on dynamic usage statistics combined with assumed

power profiles. This hybrid design balances accuracy with broad hardware compatibility. Carbon emissions are computed based on total energy and country-specific average carbon intensity, following well-established conversion practices. Although real-time carbon data (e.g., from CO_2 Signal) is not yet integrated, the system is modular enough to support such extensions.

Importantly, Green Llama's reporting capabilities go beyond static summaries. Users can track metrics mid-conversation via summaries, conduct structured benchmarking with standardized prompts, and inspect results interactively in the browser. The web viewer visualizes energy breakdowns and emissions with pie charts, metric plots, and contextual real-world equivalents, reinforcing usability and interpretability. These features make it practical to compare models not just by performance, but also by environmental cost.

A noteworthy finding during benchmark testing is that **LLaMA 3.2 1B**, with **1.24 billion parameters**, consistently consumed less energy than **Gemma 3 1B**. This illustrates that model size alone does not determine energy efficiency, and that architectural choices and hardware-level behavior play a critical role.

Overall, Green Llama provides a novel and effective approach to sustainability-aware model development. Its CLI design enables seamless integration into existing workflows, while its reporting tools make sustainability metrics actionable. The implementation choices reflect a balance between measurement precision and system robustness, paving the way for future enhancements like live carbon API support, finer-grained CPU metrics, and deeper model introspection.

VII. Future Work

Several directions could extend the capabilities of Green Llama. First, we plan to replace the current static energy estimation models with direct hardware power measurements where supported, using interfaces like Intel RAPL or Apple's energy diagnostics. This would improve accuracy, especially in high-variability systems.

We also aim to support a broader range of platforms, including AMD GPUs and Apple Silicon, to ensure cross-device consistency. A key extension will involve making carbon intensity estimation dynamic—drawing real-time data from public APIs such as electricityMap—to reflect local grid conditions at the time of inference.

On the user side, we envision embedding the viewer directly into LLM deployment dashboards (e.g., via Ollama plugins or browser extensions). This would allow developers to monitor emissions in real time during everyday usage. For researchers and educators, we plan to bundle preset experiments for teaching green AI principles.

Finally, we are exploring integration with model hosting platforms to make energy labels part of model metadata, enabling sustainability-aware model selection by default.

References

[1] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems, July 2020. arXiv:2007.03051.

[2] Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Marc Alencon, Michał Stechły, Christian Bauer, Lucas Otávio N. de Araújo, JPW, and MinervaBooks. mlco2/codecarbon: v2.4.1, May 2024.

[3] Benoit Courty, Victor Schmidt, and Sasha et al. Luccioni. Codecarbon: v2.4.1.
urlhttps://github.com/mlco2/codecarbon, 2024. Accessed April 2025.

[4] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

[5] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning, 2019.

[6] Hause Lin and Tawab Safi. ollamar: An r package for running large language models. *Journal of Open Source Software*, 10(105):7211, January 2025. Corresponding author: Hause Lin (hauselin@gmail.com, ORCID: 0000-0003-4590-7039); Co-author: Tawab Safi (asaficontact@gmail.com, ORCID: 0009-0000-5659-9890).

[7] Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. Efficient benchmarking of language models, 2024.

[8] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.

[9] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp, 2019.

[10] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

[11] Tdurieux. Energibridge. https://github.com/tdurieux/EnergiBridge, 2021. GitHub repository, accessed April 03, 2025.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.