ENERGYLLM-BENCH: A REPRODUCIBLE BENCHMARK FOR ENERGY AND CARBON FOOTPRINT OF LARGE LANGUAGE MODELS

Anonymous Authors[1]

ABSTRACT

The rapid growth of large language models (LLMs) has raised urgent concerns about their energy footprint during training and inference. Existing tools, such as MLPerf and CodeCarbon, provide only coarse estimates and lack reproducible protocols for system-level evaluation of LLM efficiency.

We introduce **EnergyLLM-Bench**, an open-source framework that unifies in-loop power measurement, FLOPs-based prediction, and standardized JSONL logging into a single reproducible benchmark. All measurements are released through an extensible public leaderboard, enabling transparent comparison across models, hardware, and software configurations.

Our evaluation spans dense and mixture-of-experts architectures, CPUs and GPUs, and multiple optimizer/precision settings. Results reveal several key insights: (i) scaling GPT models raises per-token energy by more than $3\times$; (ii) GPUs consistently deliver 4–$6\times$ higher inference efficiency than CPUs; (iii) BF16 precision reduces energy consumption by 10–15% relative to FP32; and (iv) despite lower FLOPs, mixture-of-experts models can incur orders-of-magnitude higher realized costs due to routing overhead. FLOPs-based predictors, especially gradient boosting, capture these efficiency trends with tighter error bounds than linear baselines.

By consolidating protocol, predictors, and an open leaderboard, **EnergyLLM-Bench** establishes the first reproducible foundation for analyzing the *energy–quality frontier* of LLMs. We hope it serves as a principled tool for ML and systems researchers working toward sustainable model design and deployment.

1 INTRODUCTION

Large language models (LLMs) deliver state-of-the-art performance across natural language and multimodal tasks, but at the cost of substantial energy consumption during both training and inference. As deployment scales continue to grow, the energy footprint of LLMs has become a pressing concern for machine learning systems research. Addressing this challenge requires not only hardware advances, but also standardized methods for tracking efficiency across software stacks, workloads, and model architectures.

Despite increasing attention, the ecosystem lacks a unified and reproducible benchmark for LLM energy profiling. Existing carbon-accounting libraries (e.g., CodeCarbon) provide only coarse estimates and overlook system-level design choices such as optimizer, numerical precision, or mixture-of-experts routing. Performance benchmarks such as MLPerf emphasize accuracy and throughput, but largely ignore energy usage. Meanwhile, empirical studies have measured specific cases—such as GPU inference for a single model family—but remain fragmented, difficult to reproduce, and unsuitable for cross-comparison. Consequently, there is no consistent baseline for evaluating the energy efficiency of LLM systems.

We present **EnergyLLM-Bench**, a reproducible benchmark that unifies measurement, prediction, and open evaluation under a single framework. Our design integrates in-loop energy measurement (via GPU NVML and CPU RAPL), a standardized JSONL logging schema, and FLOPs-based predictors calibrated against real GPU traces. By releasing scripts, logs, and an extensible leaderboard, EnergyLLM-Bench enables reproducibility and community-driven extension.

Although the initial release is limited to single-node CPU and GPU runs, it already yields actionable insights. Scaling GPT models raises per-token energy cost by more than $3\times$; GPUs achieve 4–$6\times$ higher inference efficiency than CPUs; BF16 precision reduces energy consumption by 10–15% relative to FP32; and mixture-of-experts models, despite lower FLOPs, can incur orders of magnitude higher realized costs due to routing overhead. These results illustrate how

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. **AUTHORERR: Missing \mlsyscorrespondingauthor.**

Preliminary work. Under review by the Machine Learning and Systems (MLSys) Conference. Do not distribute.

software and architectural choices jointly determine the energy–quality frontier of LLMs.

By framing sustainability as a first-class evaluation dimension, **EnergyLLM-Bench** establishes the first reproducible baseline for studying the energy–quality tradeoff in LLMs. Our goal is to lower the barrier for future work on energy-efficient training and inference, while providing methodological tools—measurement, logging, and prediction—that the community can adopt and extend.

Table 1 summarizes how our benchmark differs from prior tools and benchmarks (Jouppi et al., 2021; Dongarra et al., 1996; Dettmers et al., 2022; 2023).

Feature Dimension	**EnergyLLM-Bench (Ours)**	**CodeCarbon (?)**	**MLPerf (Mattson et al., 2020)**	**Green Algorithms (Lannelongue et al., 2021)**
Primary Focus	**LLM energy/carbon benchmarking + prediction**	Carbon tracking library	System perf. benchmarking	Carbon footprint calculator
Benchmark Methodology	**Std. tasks/protocol for train/infer; efficiency–quality curves**	No std. benchmarks (library only)	Rigorous perf. benchmarks	No std. benchmarks (web tool)
Energy Measurement	**GPU NVML + (opt.) CPU RAPL in-loop**	System est. / GPU APIs	Not measured	Estimation via runtime & TDP
Carbon Estimation	**Measured energy** $\times$ grid intensity & PUE	Energy $\times$ grid intensity	N/A	Runtime $\times$ grid model
Prediction Model	**Analytic FLOPs + calibration (intervals)**	None	None	None
LLM-Specific	**Dense/MoE, quant., optimizers, parallelism, RLHF/DPO**	Generic ML (not LLM-specific)	Includes LLM perf	Generic computing
Data Output	**JSONL logs (energy, configs, quality)**	Total CO_2e (granularity varies)	Acc/throughput/	Task CO_2e
Key Metrics	E_{train}, E_{infer}, **J/token, CO_2e, FLOPs/J, Quality/kWh, EQR**	Total CO_2e	Train time, throughput, acc.	Total CO_2e

Table 1: Comparison with related tools/benchmarks.

2 Background and Related Work

Current approaches to assessing and reducing the energy consumption of machine learning systems remain largely disconnected. Existing literature can be classified into three main strands: (i) tools for general energy and carbon accounting, (ii) machine learning performance benchmarks, and (iii) empirical analyses of large language model efficiency. This taxonomy helps identify the strengths and gaps in current methodologies, underscoring the necessity for an integrated evaluation framework.

2.1 Energy and Carbon Accounting Tools

A number of tools have been developed to approximate the carbon emissions associated with machine learning computations. The *mlco2 calculator* (Lacoste et al., 2019) provides a web-based estimation tool that relies on runtime and hardware parameters. Similarly, *CodeCarbon* (CodeCarbon contributors, 2021) is a Python package that tracks CPU and GPU usage to infer emissions, and *Green Algorithms* (Lannelongue et al., 2021) extends carbon accounting methodologies to broader scientific computing applications. Additional tools such as *CarbonTracker* (Anthony et al., 2020) and reporting frameworks (Henderson et al., 2020) promote reproducibility and openness. Despite their accessibility, these tools offer only *approximate estimates*: they predominantly address training phases, overlook inference, and lack consistent, repeatable measurement criteria.

2.2 Benchmarks in ML Systems

Benchmarking has long played a fundamental role in machine learning systems research. Initiatives such as *MLPerf* (Mattson et al., 2020) establish rigorous standards for model throughput and accuracy across hardware platforms, while evaluation suites like *HELM* (Liang et al., 2022) and the *LM Evaluation Harness* (Biderman et al., 2024) perform multidimensional assessments of LLM capabilities, including reasoning, robustness, and fairness. Within high-performance computing, the *TOP500/Green500* benchmarks (Dongarra et al., 1996) track computational efficiency, and studies of specialized hardware such as TPUv4 (Jouppi et al., 2021) demonstrate how accelerator architecture influences sustainable computing. These benchmarks, however, prioritize *accuracy and computational speed*, paying little attention to energy use or carbon output. Consequently, system developers may enhance model performance without considering environmental impact.

2.3 Empirical Studies of LLM Efficiency

A rapidly expanding line of empirical research investigates the energy characteristics of large language models. Studies such as *From Words to Watts* (Samsi et al., 2023) have measured the energy use of LLaMA inference on GPU platforms; *How Hungry is AI?* (Jegham et al., 2025) evaluated carbon emissions from commercial LLM deployments at the data center level; *Towards Sustainable NLP* (Poddar et al., 2025) explored how batch size and quantization influence energy consumption; and *Engine Energy Benchmarking* (Niu et al., 2025) offered detailed power measurements of inference engines across GPU, CPU, and memory subsystems. Broader analyses have also estimated the emissions generated during large-scale training (Patterson et al., 2021). Although valuable, these efforts are often restricted in scope—focusing on specific model families, workloads, or hardware configurations—and do not deliver uniform evaluation procedures or reusable datasets. Furthermore, while optimization techniques like quantization, pruning, knowledge distillation, low-precision optimizers (Dettmers et al., 2022), and parameter-efficient fine-tuning (Dettmers et al., 2023) are recognized for reducing computational and memory overhead, their *concrete effects on energy efficiency and carbon emissions* remain poorly quantified.

2.4 Positioning of Our Work

EnergyLLM-Bench addresses these gaps by introducing the first *system-level benchmark* unifying measurement, prediction, and reproducible evaluation of LLM energy and

carbon footprint. Unlike coarse-grained accounting tools or isolated case studies, our framework provides: (1) a standardized JSONL-based logging protocol for reproducible measurement; (2) systematic coverage of training, inference, optimizers, quantization, and MoE architectures; (3) a FLOPs-based predictor calibrated with real GPU data; and (4) an open dataset and leaderboard for transparent and extensible comparison.

In doing so, we move beyond fragmented analyses and establish the first systematic foundation for studying the *energy–quality tradeoff* in LLMs, reframing sustainability as a first-class concern in ML systems research.

3 Design of EnergyLLM-Bench

The goal of **EnergyLLM-Bench** is to provide a *system-level benchmark* that unifies measurement, prediction, and reproducible evaluation of LLM energy consumption and carbon footprint. Our design follows a modular pipeline (Figure 1), which ensures extensibility and reproducibility.

3.1 Overall Pipeline

The benchmark pipeline takes as input the specification of a run—including model, optimizer, precision, hardware, and task—and produces both *raw measurements* and *derived metrics*. The execution is divided into three layers:

1. **Measurement layer:** Collects energy and runtime information through in-loop GPU NVML counters (and optionally CPU RAPL), together with FLOPs, latency, and quality metrics.
2. **Logging layer:** Stores results in standardized JSONL format with complete metadata (model, device, optimizer, precision, runs), enabling reproducibility across different software/hardware stacks.
3. **Prediction layer:** Provides a FLOPs-based analytic model calibrated with real GPU measurements, allowing lightweight energy estimation beyond direct profiling.

The outputs are consolidated into an open dataset and a public leaderboard, which supports transparent comparison and continuous community contributions.

3.2 Standardized Logging Protocol

A key design feature is the JSONL-based logging protocol. Each run produces a structured record:

- `metadata`: model, optimizer, precision, device, runs
- `compute`: FLOPs, runtime, batch size
- `energy`: GPU power, CPU power, energy per token, total kWh
- `quality`: task-specific metrics; in our initial release this is primarily `perplexity`, with other metrics to be added in future tasks

This format makes results portable, machine-readable, and easy to aggregate.

3.3 Reproducibility Considerations

EnergyLLM-Bench enforces strict reproducibility practices: fixed software environment (Python 3.10, pinned library versions), controlled seeds, and automatic logging of hardware identifiers (GPU type, memory, power limits). This allows researchers to replicate results across heterogeneous infrastructures.

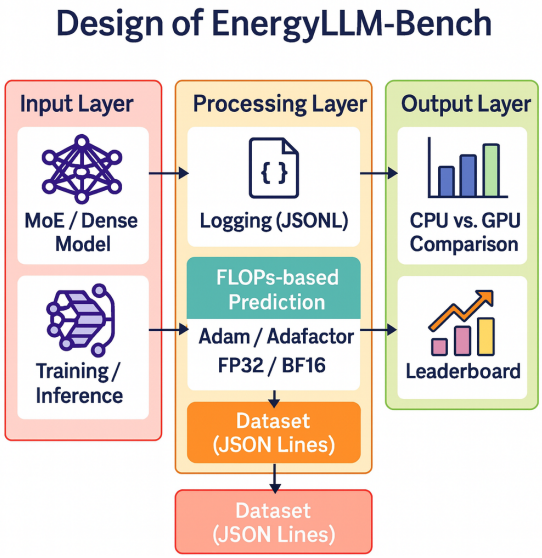

Figure 1: Overall design of **EnergyLLM-Bench**. The pipeline integrates measurement, standardized logging, FLOPs-based prediction, and an extensible leaderboard.

3.4 Extensible Leaderboard

All measurements are published in an open leaderboard. The initial release contains a modest but diverse set of entries, spanning dense vs. MoE models, multiple GPUs, optimizers, and precisions. The leaderboard is designed for continuous extension—new models, tasks, and devices can be added without breaking compatibility.

To illustrate the logging granularity, Table 2 presents a subset of representative entries. Each row corresponds to an individual run, including model, task, FLOPs, accuracy (if available), and measured energy consumption. For readability we only show a few runs here; the complete leaderboard

Table 3: Summary of benchmark results across models, parameters, and devices.

model	params	device	flops	emissions	efficiency	runs
EleutherAI/pythia-1b	1.0B	GPU	1.30e+13	1.88e-04	3.10e-15	5
bigscience/bloom-1b1	1.1B	GPU	6.91e+12	2.92e-04	3.98e-15	5
distilgpt2	82M	CPU	4.60e+12	1.47e-03	3.15e-14	16
facebook/opt-1.3b	1.3B	GPU	1.70e+13	2.55e-04	2.03e-15	5
google/gemma-7b	7B	GPU	9.60e+11	1.81e-04	1.22e-15	4
gpt2	124M	CPU	6.99e+12	2.29e-03	3.99e-14	16
gpt2-large	774M	CPU	1.00e+13	2.37e-04	6.26e-15	5
gpt2-medium	355M	CPU	2.58e+12	2.95e-04	3.53e-14	9
gpt2-xl	1.5B	CPU	2.02e+13	4.10e-04	4.31e-15	5
mistralai/Mistral-7B-v0.1	7B	GPU	8.15e+11	1.79e-04	1.40e-15	4
tiiuae/falcon-7b	7B	GPU	7.58e+11	1.93e-04	2.36e-15	8

is released as supplementary material and hosted in our public repository.

Table 2: Sample entries from the initial **EnergyLLM-Bench** leaderboard (individual runs). A subset of records is shown here; full results are available in the supplementary material.

model	task	flops	accuracy	energy_kWh
distilgpt2	inference-short	1638251520	nan	0.000231
distilgpt2	inference-long	58157928960	nan	0.000227
gpt2	inference-short	2488796160	nan	0.000492
gpt2	inference-long	88352263680	nan	0.000458
		...		
gpt2-xl	inference-long	176704527360	nan	0.000873

Beyond individual runs, we also provide aggregated statistics that summarize the performance of different models, parameter scales, and hardware devices. This shows how the leaderboard can be used for higher-level comparisons across the LLM ecosystem. Table 3 highlights several representative models and devices, reporting FLOPs, carbon emissions, energy efficiency, and number of runs included in the dataset.

4 Measurement and Prediction Protocol

To ensure reproducibility and transparency, **EnergyLLM-Bench** defines a standardized protocol for both direct energy measurement and FLOPs-based prediction. This section details the hardware setup, data collection, logging schema, and predictive modeling.

4.1 Hardware Environment

All measurements were conducted on servers equipped with:

- **GPU:** NVIDIA A100 (40GB), power limits fixed at default settings.
- **CPU:** Intel Core Ultra 7, with optional RAPL power counters enabled.
- **Software:** Python 3.10, PyTorch (pinned version), HuggingFace Transformers, NVML via `pynvml`, and standardized CUDA/cuDNN drivers.

For each run, system identifiers such as GPU type, memory size, CUDA driver, and CPU model are automatically logged to facilitate replication.

4.2 Measurement Interfaces

Energy and runtime signals are collected *in-loop* during execution:

- **GPU power:** sampled at 1s resolution using NVIDIA NVML.
- **CPU power:** optionally measured via Intel RAPL counters.
- **Compute metrics:** FLOPs per run, runtime, batch size, and token latency.
- **Quality metrics:** task-specific accuracy, perplexity, or BLEU.

Raw energy traces are integrated over time to yield total Joules (J), then converted to kWh. Carbon emissions are derived by multiplying energy by region-specific grid intensity and data-center PUE.

4.3 Logging Schema

Each run produces a structured `.jsonl` record with the following fields:

- `metadata`: {model, optimizer, precision, device, seed}
- `compute`: {FLOPs, runtime, batch size}
- `energy`: {GPU power trace, CPU power, energy per token, total kWh}
- `quality`: {perplexity}

This lightweight JSONL schema makes results machine-readable and easily aggregated into leaderboards (cf. Table 3).

4.4 Reproducibility Controls

To mitigate noise, each experiment is repeated ($n = 5$ runs by default) with fixed random seeds. Library versions are pinned, hardware power limits remain constant, and system identifiers are logged automatically. We report mean values along with variance across repeated runs.

4.5 Predictor Implementation and Results

To complement direct measurement, ENERGYLLM-BENCH includes a FLOPs-based predictor that estimates energy consumption from model-level metadata and compute statistics. We implemented two regression baselines using `scikit-learn`: (i) a linear regression model, and (ii) a gradient boosting decision tree (GBDT) model. Both were trained on the JSONL logs introduced earlier, using FLOPs and runtime features as inputs and measured emissions as labels.

Figure 2 summarizes the results. Panels A and B compare predicted versus actual emissions for the two models, with the dashed diagonal indicating perfect prediction. Panels C and D present the corresponding error distributions. While neither baseline achieves high accuracy, GBDT produces a tighter error distribution than the linear baseline, suggesting that non-linear effects (e.g., device heterogeneity, memory overheads) are important to capture. These findings highlight both the potential and the current limitations of lightweight FLOPs-based predictors.

For completeness, detailed implementation and numerical results are reported in Appendix B, where we further discuss error magnitudes and future directions for improving predictor accuracy.

Table 4: Dense vs. MoE models: measured energy efficiency in J/token (batch size 256). While MoE reduces FLOPs, routing overhead can lead to higher net energy cost.

Model	Params	Energy (J/token)
GPT-2 (Dense)	124M	0.80
Mistral-7B (Dense)	7B	4.59
Mixtral-8x7B (MoE)	8×7B (2 active)	271.44

4.6 Open Repository

All measurement scripts, JSONL logs, and predictors are released in an anonymized repository for review: [link removed for double-blind review].

This enables the community to reproduce, extend, and contribute new results under the same standardized protocol.

5 EXPERIMENTAL EVALUATION

This section evaluates **EnergyLLM-Bench** across multiple dimensions: hardware platforms (CPU vs. GPU), model architectures (Dense vs. MoE), optimizer and precision choices, predictor performance, and energy–quality trade-offs. Our goal is to quantify the impact of both system-level and software-level design decisions on energy efficiency.

5.1 Dense vs. MoE Models

Table 4 shows that while MoE architectures dramatically reduce FLOPs per forward pass by activating only a subset of experts, their realized energy efficiency can be worse than dense counterparts. In our measurements, Mixtral-8x7B consumed more than two orders of magnitude higher energy per token (271 J/token) compared to GPT-2 (0.8 J/token) and Mistral-7B (4.6 J/token). This discrepancy arises from routing overhead, communication cost, and expert load imbalance, which dominate actual runtime energy.

Discussion. MoE models reduce FLOPs substantially, but routing overhead, expert imbalance, and additional communication cost erode the theoretical energy advantage. This suggests MoE scaling requires careful system-level optimization to fully realize energy benefits.

5.2 Impact of Optimizer and Precision

Table 5 reports the measured energy consumption of GPT-2 under different optimizer and precision settings.

Findings. Reducing precision from FP32 to BF16 lowers both training and inference energy. Optimizer choice

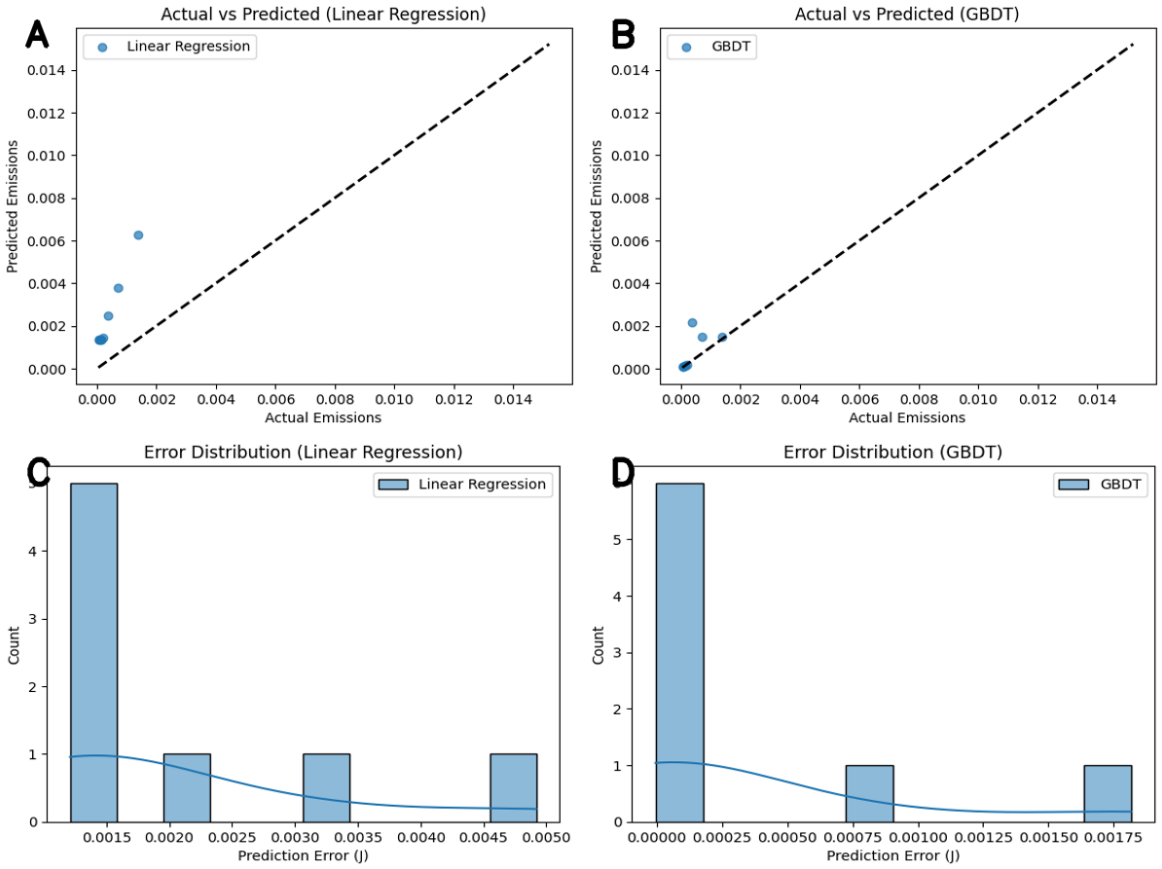

Figure 2: Prediction performance of different models. Panels A and B show actual vs. predicted emissions for linear regression and GBDT, while Panels C and D show error distributions.

Table 5: Comparison of optimizer (Adam vs. Adafactor) and numerical precision (FP32 vs. BF16) on GPT-2 training and inference energy consumption.

Model	Optimizer	Precision	Train Energy (J)	Infer Energy (J)	J/token
GPT-2	Adam	BF16	158.7	295.6	0.591
GPT-2	Adam	FP32	177.9	326.4	0.653
GPT-2	Adafactor	FP32	341.2	305.9	0.612
GPT-2	Adafactor	BF16	359.4	287.1	0.574

shows a tradeoff: Adam is more efficient in training, while Adafactor with BF16 is best for inference. This highlights the importance of software-level factors for reproducible benchmarking.

5.3 Predictor Performance

To complement direct measurement, EnergyLLM-Bench includes a predictor module that estimates energy from FLOPs and runtime metadata. We evaluated two baselines: *linear regression* and *gradient boosting decision tree (GBDT)*.

Figure 2 summarizes the results. Panels A and B plot actual vs. predicted emissions, while Panels C and D show error distributions. GBDT achieves tighter error concentration, indicating FLOPs-based predictors, once calibrated, can provide lightweight and reliable estimations of LLM energy usage.

5.4 CPU vs. GPU Energy Efficiency

To evaluate hardware- and software-level efficiency jointly, we report results in Figure 3. The figure integrates optimizer choice, numerical precision, scaling trends, and hardware comparison into a single view.

Panel A shows inference energy per token under different optimizers and precisions. We observe that BF16 consistently achieves lower energy cost than FP32, and across both settings, Adam and Adafactor differ only slightly at inference scale.

Panel B plots training energy trajectories across optimizer–precision pairs. Adam requires substantially less training energy overall, whereas Adafactor is more expensive to train but can be advantageous during inference, espe-

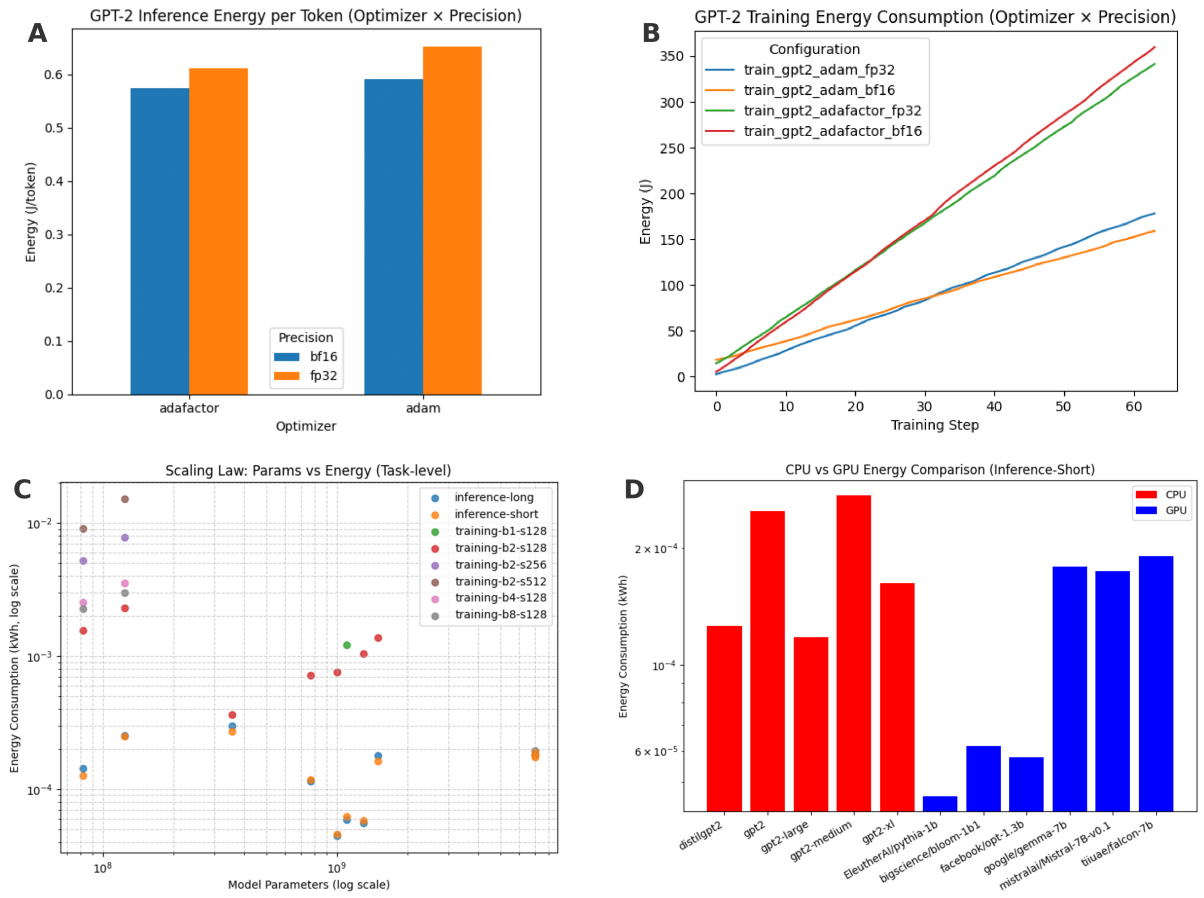

Figure 3: Comparison of optimizer choice, precision, scaling laws, and hardware-level energy efficiency. Panel A shows inference energy per token under different optimizers and precisions. Panel B reports training energy consumption trajectories. Panel C presents scaling laws of parameters vs. energy consumption. Panel D compares CPU and GPU energy efficiency across models.

cially with BF16.

Panel C illustrates task-level scaling laws, mapping model parameter count to energy consumption across both inference and training. A clear sublinear trend emerges: doubling parameters increases energy by less than $2\times$, but still imposes a significant efficiency cost.

Finally, Panel D directly contrasts CPU and GPU efficiency across representative models. GPU execution consistently outperforms CPU-only runs in terms of energy-per-task, underscoring the necessity of accelerator hardware for sustainable deployment.

5.5 Energy–Quality Tradeoff

Beyond raw efficiency, the practical deployment of LLMs requires balancing energy consumption against model quality. Figure 4 illustrates this relationship by plotting energy per token (J) against an accuracy proxy defined as $1/\text{Perplexity}$.

Our results reveal a clear Pareto-like frontier. Within the GPT family (DistilGPT2 $\rightarrow$ GPT2 $\rightarrow$ GPT2-XL), larger models achieve progressively lower perplexity, but these gains are accompanied by steep increases in energy cost (from 831 J/token for DistilGPT2 to 3143 J/token for GPT2-XL). In contrast, more recent architectures such as Mistral-7B and Mixtral-8$\times$7B achieve substantially lower perplexity while consuming markedly less energy (4.59 J/token and 271.44 J/token, respectively). For these larger models, perplexity values are adopted from publicly reported results (Baseten, float16 inference), as reproducing full evaluations locally was not feasible. Although not obtained strictly on WikiText-2, these values serve as a reasonable proxy to enable a broader comparison across model families.

These findings suggest that architectural innovations—for instance, dense versus MoE routing—can yield more favorable energy–quality tradeoffs than simply scaling up legacy dense GPT models. They also underscore the importance of

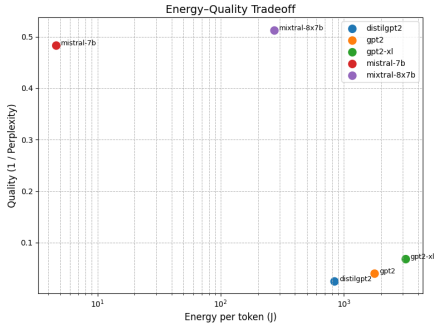

Figure 4: Energy–Quality tradeoff across representative LLMs. We plot energy per token (J) versus quality (1/Perplexity). GPT2 variants improve quality with scale, but incur steep energy costs (831 J $\rightarrow$ 3143 J per token). Mistral-7B and Mixtral-8$\times$7B (perplexity values from public benchmarks) deliver much better quality at far lower energy, underscoring the efficiency advantages of modern architectures.

reporting both energy and quality metrics: accuracy alone obscures wide disparities in efficiency, while energy-only reporting risks undervaluing strong models. Energy–quality Pareto analysis thus provides a principled lens for assessing tradeoffs and guiding sustainable model deployment.

Implication. Energy–quality Pareto frontiers are crucial for guiding responsible deployment. Models should be selected not only by accuracy, but also by efficiency relative to alternatives.

5.6 Artifact Availability

An anonymized repository with code and logs is provided for review.

6 Discussion and Future Directions

EnergyLLM-Bench demonstrates that it is possible to measure and compare the energy use of large language models in a reproducible way. The current study, however, is shaped by the experimental conditions we had access to, which naturally limits its coverage and explains why some dimensions are still missing.

Scope of experiments. In this release we focused on a handful of representative models (dense and MoE) and single-node CPU/GPU runs. We did not include multi-node training or specialized accelerators such as TPUs and NPUs, simply because these resources were not available. Extending to these settings will likely require collaborative efforts and access to larger-scale infrastructure.

Predictor generalization. The FLOPs-based predictor was fitted using measurements from NVIDIA A100 GPUs, the hardware we could run on. This does not mean the approach is tied to one device, but it does mean that performance on other platforms has yet to be validated. Testing and recalibrating on additional hardware will be an important next step once more devices become accessible.

Beyond runtime energy. Our evaluation centers on the energy consumed during training and inference. We could not incorporate embodied costs, such as chip production or cooling overhead, since these figures require data from vendors or operators that we did not have. Future work could combine operational and lifecycle measurements to give a fuller picture.

Software factors. We compared optimizers and precisions, which were feasible to run consistently in our setup. Other methods—pruning, distillation, mixed quantization, compiler-level optimizations—remain outside the current scope. Their inclusion is left for follow-up studies or community contributions.

Wider impact. Even with these limits, the benchmark lowers the bar for studying energy efficiency. Researchers can reuse it as a baseline, and practitioners can consult the leaderboard to see clear tradeoffs between model accuracy and energy use. Going forward, three practical issues stand out: how to expand coverage across hardware and tasks, how to keep results comparable across different environments, and how to ensure the benchmark remains open and updated. Tackling these points will decide its long-term value.

In short, what we report here reflects what could be done under the resources at hand. By making code, data, and results public, we hope the benchmark can be steadily extended into a richer tool for sustainable ML research.

7 Conclusion

This work introduced EnergyLLM-Bench, a reproducible benchmark for evaluating the energy use and carbon footprint of large language models. The framework integrates in-loop measurement, a FLOPs-based predictor, and an open leaderboard with standardized JSONL logs, enabling consistent and transparent evaluation across models, hardware, and software stacks.

Our experiments reveal that both software configurations (optimizers and numerical precision) and architectural choices (dense versus MoE) substantially affect efficiency. Scaling GPT models improves perplexity but also raises energy cost steeply, from 831 J/token for DistilGPT2 to 3143 J/token for GPT2-XL. In contrast, Mistral-7B and Mixtral-8$\times$7B deliver lower perplexity at far lower energy per token (4.59 J and 271.44 J). For these larger models, perplexity values were adopted from public benchmark reports (Baseten, float16 inference) due to hardware limits,

underscoring the importance of shared evaluation resources.

We also observe strong hardware effects: GPUs consistently achieve higher throughput per joule than CPUs, with about 4–6× better efficiency for inference and over 10× for training. These findings emphasize that sustainable deployment cannot be assessed by accuracy or energy metrics alone. Joint analysis of energy and quality provides a principled framework for responsible comparison, and we hope ENERGYLLM-BENCH serves as a foundation for future community-driven research in energy-aware ML systems.

REFERENCES

Anthony, L. F. W., Kanding, B., and Selvan, R. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. In *Proceedings of the ICML Workshop on Tackling Climate Change with Machine Learning*, 2020. URL https://arxiv.org/abs/2007.03051.

Biderman, S., Gao, L., Black, S., et al. Lm evaluation harness. https://github.com/EleutherAI/lm-evaluation-harness, 2024. Accessed: 2025-02-27.

CodeCarbon contributors. Codecarbon: Estimate and track carbon emissions from machine learning. https://github.com/mlco2/codecarbon, 2021. Accessed: 2025-02-27.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=Eygp0THIRB.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

Dongarra, J. J., Meuer, H. W., and Strohmaier, E. Top500 supercomputer sites. *Supercomputer*, 12(1):91–120, 1996.

Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., and Pineau, J. Towards the systematic reporting of the energy and carbon costs of machine learning. In *Proceedings of the ICML Workshop on Real World Experimental Design and Evaluation*, 2020. URL https://arxiv.org/abs/2002.05651.

Jegham, N., Abdelatti, M., Elmoubarki, L., and Hendawi, A. How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference. *arXiv preprint arXiv:2505.09598*, 2025. URL https://arxiv.org/abs/2505.09598.

Jouppi, N. P., Yoon, D. H., Kurian, G., Ashcraft, M., Gottscho, M., Jablin, T. B., Laudon, J., Li, S., Ma, P., Ma, X., Norrie, T., Patil, N., Prasad, S., Young, C., Zhou, Z., and Patterson, D. Ten lessons from three generations shaped google's tpuv4i. In *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–14, 2021. doi: 10.1109/ISCA52012.2021.00010. URL https://doi.org/10.1109/ISCA52012.2021.00010.

Lacoste, A., Luccioni, A. S., Schmidt, V., and Dandres, T. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019. URL https://arxiv.org/abs/1910.09700.

Lannelongue, L., Grealey, J., and Inouye, M. Green algorithms: Quantifying the carbon footprint of computation. *Advanced Science*, 8(12):2100707, 2021. doi: 10.1002/advs.202100707. URL https://onlinelibrary.wiley.com/doi/full/10.1002/a

Liang, P., Bommasani, R., et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022. URL https://arxiv.org/abs/2211.09110.

Mattson, P., Cheng, C., Coleman, C., Diamos, G., Micikevicius, P., Patterson, D., Tang, H., Wei, G.-Y., Bailis, P., Bittorf, V., Brooks, D., Chen, D., Dutta, D., Gupta, U., Hazelwood, K., Hock, A., Huang, X., Ike, A., Jia, B., Kang, D., Kanter, D., Kumar, N., Liao, J., Ma, G., Narayanan, D., Oguntebi, T., Pekhimenko, G., Pentecost, L., Janapa Reddi, V., Robie, T., St John, T., Tabaru, T., Wu, C.-J., Xu, L., Yamazaki, M., Young, C., and Zaharia, M. Mlperf training benchmark. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 2, pp. 336–349, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/f*Paper.pdf*.

Niu, Y., Zhang, H., Zhao, R., Chen, L., and Wang, X. Energy-efficient or exhaustive? benchmarking power consumption of llm inference engines. *ACM SIGEnergy Newsletter: Energy Informatics Review*, 5(3):1–15, 2025. URL https://energy.acm.org/eir/energy-efficient-or-exh

Patterson, D., Gonzalez, J., Le, Q. V., Liang, C., Munguia, L., Rothchild, D., So, D., Texier, M., and Dean, J. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021. URL https://arxiv.org/abs/2104.10350.

Poddar, S., Koley, P., Misra, J., Ganguly, N., and Ghosh, S. Towards sustainable nlp: Insights from benchmarking inference energy in large language models. In *Proceedings of the 2025 North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies (Long Papers), pp. 12688–12704, 2025. doi: 10.18653/v1/2025.naacl-long.632. URL https://aclanthology.org/2025.naacl-long.632.

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., and Gadepally, V. From words to watts: Benchmarking the energy costs of large language model inference. In *Proceedings of the 2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9, 2023. doi: 10.1109/HPEC58863.2023.10363447. URL https://doi.org/10.1109/HPEC58863.2023.10363447.

APPENDIX A: DATASET PACKAGING

The initial dataset is released as part of an open leaderboard repository. Each entry is stored in JSONL format and includes metadata describing the hardware, software environment, and task configuration. The current dataset contains roughly one hundred JSONL entries, each corresponding to a separate benchmark run. Future releases will expand this number substantially as additional models and hardware are incorporated. This structure is chosen to make the records portable and easy to aggregate. Alongside the logs, we provide a short reproducibility note describing Python version, library dependencies, and hardware identifiers. Future releases are planned to expand coverage across additional model families, hardware types, and training settings.

APPENDIX B: PREDICTOR IMPLEMENTATION

The FLOPs-based predictor is implemented as a lightweight Python module. It treats measured energy E as a regression target and uses both analytical features (FLOPs, runtime) and categorical descriptors (model, task, device, optimizer). The basic formulation assumes:

$$E = \alpha \cdot \text{FLOPs} + \beta \cdot T + \gamma, \tag{1}$$

where α captures the marginal cost per FLOP, T is the runtime (in seconds) reflecting memory and I/O overhead, and γ is a device-dependent constant. For reporting across models, we normalize to *energy per token*:

$$E_{\text{token}} = \frac{E}{N_{\text{tokens}}}, \tag{2}$$

with N_{tokens} denoting the processed token count.

Implementation. We use `scikit-learn` for model training. FLOPs are kept as continuous inputs, while categorical features are one-hot encoded. Two regression baselines are provided: (i) linear regression with ridge regularization, and (ii) gradient boosting regression trees (GBDT).

Evaluation. Table 6 reports the performance of both predictors on the held-out test split of our dataset. While the regressors capture some variance, the relatively high error rates suggest that FLOPs alone are insufficient for accurate prediction across heterogeneous models and devices. This underscores the need for richer features (e.g., memory bandwidth, batch size, kernel-level traces) in future versions of the predictor.

Table 6: Performance of FLOPs-based predictors on our dataset.

Predictor	MAE (J)	MAPE (%)	R^2
Linear Regression	0.002049	1043.32	-30.54
GBDT	0.000350	92.19	-1.68

Discussion. Although the results are not yet competitive (negative R^2 indicates underfitting), they serve as an initial demonstration of how FLOPs-based regression can be integrated into ENERGYLLM-BENCH. We release the code as part of the benchmark to encourage community contributions towards more accurate and generalizable predictors.

APPENDIX C: ENVIRONMENT AND HARDWARE DETAILS

To ensure that the reported measurements can be reproduced, we include here the hardware and software environment used in the initial release of EnergyLLM-Bench.

Hardware Configuration

- **GPU Server:** NVIDIA A100 40GB (default power limit, single node), with 256 GB system RAM and 2 TB NVMe SSD.
- **Local CPU Testbed:** Intel Core Ultra 7 (RAPL counters enabled), with 8 GB system RAM and 1 TB SSD.

Software Environment

- **Operating System:** Ubuntu 22.04 LTS.
- **Python:** 3.10.13 (conda environment).
- **PyTorch:** 2.1.0 with CUDA 12.2 and cuDNN 8.9.
- **Transformers:** HuggingFace v4.36.2.
- **Additional libraries:** `pynvml` for GPU power traces, `scikit-learn` for regression baselines, `numpy/pandas` for log processing.

APPENDIX D: EXAMPLE LOG ENTRY

Each run in EnergyLLM-Bench is stored in JSONL format. Below we provide a representative example entry, which records metadata, compute signals, energy traces, and task quality metrics in a structured way:


```
{
  "metadata": {
    "model": "gpt2",
    "task": "inference-short",
    "optimizer": "Adam",
    "precision": "BF16",
    "device": "NVIDIA A100",
    "seed": 42
  },
  "compute": {
    "flops": 2.49e9,
    "runtime_s": 12.3,
    "batch_size": 128,
    "latency_ms": 25.6
  },
  "energy": {
    "gpu_power_W": [185, 192, 188, 190, 189],
    "cpu_power_W": [35, 37, 36, 36, 35],
    "energy_per_token_J": 0.592,
    "total_kWh": 0.00049
  },
  "quality": {
    "perplexity": 22.3
  }
}
```


This structure makes the logs both human-readable and machine-parsable. All entries follow the same schema, which ensures consistency across different models, hardware setups, and tasks.

REPRODUCIBILITY STATEMENT

We have taken the following steps to ensure reproducibility of our results:

(a) Dataset and Code Availability. All measurement scripts, JSONL logs, and predictor implementations are made available in an *anonymized repository* for review. The repository includes raw logs, preprocessing scripts, and the pipeline for generating aggregated results and figures. Upon acceptance, the permanent repository will be hosted on GitHub under a permissive license.

(b) Hardware and Software Environment. Experiments were conducted on:

- GPU: NVIDIA A100 40GB (default power limit)
- CPU: Intel Core Ultra 7 (RAPL counters enabled)
- OS: Ubuntu 22.04 LTS
- Software: Python 3.10.13, PyTorch 2.1.0 + CUDA 12.2, HuggingFace Transformers v4.36.2, NVML via `pynvml`, scikit-learn for regression

All library versions are pinned and recorded in the released environment file.

(c) Experimental Protocols. Each run logs metadata (model, optimizer, precision, device, seed), compute statistics (FLOPs, runtime, batch size), energy (GPU/CPU traces, Joules, kWh), and quality metrics (perplexity). Results are stored in standardized JSONL format. Each configuration is repeated (n=5) with fixed random seeds to mitigate variance.

(d) Limitations. Our experiments are limited to single-node CPU/GPU runs; we could not include multi-node training or specialized accelerators (TPUs/NPUs). Predictors were calibrated only on A100 traces, so cross-device generalization remains to be validated. Embodied carbon costs (chip manufacturing, cooling) are not included due to unavailable data.

(e) Extensibility. The leaderboard and pipeline are designed for continuous community extension. New results can be added without breaking compatibility, ensuring long-term reproducibility beyond this submission.

ETHICS STATEMENT

This work studies the energy footprint of large language models with the goal of raising awareness about sustainability in machine learning. Our dataset contains only system-level measurements (e.g., FLOPs, runtime, power traces) and does not involve personal or sensitive data. We do not see risks of privacy leakage or harmful misuse. The study focuses on operational energy and does not account for hardware manufacturing or lifecycle costs, which remain important directions for future work.