Group 26: Towards LLM Energy Labels: Accuracy and Energy Efficiency

Levi Ari Pronk
EEMCS
Delft University of Technology

Ocean Wang
EEMCS
Delft University of Technology

Nicholas Wu
EEMCS
Delft University of Technology

Madhav Chawla
EEMCS
Delft University of Technology

Yasar Saltuk Bugra Kocdas
EEMCS
Delft University of Technology

***Abstract*—Large Language Models (LLMs) are deployed at increasing scale, yet their energy consumption remains opaque to end users. Inspired by the EU energy labelling framework for appliances, we propose Energy Per Correct Answer (EPCA) as a composite metric that jointly captures inference energy cost and task accuracy. We present an open-source benchmark pipeline that executes domain specific tasks on locally hosted LLMs, measures GPU energy consumption via NVIDIA power telemetry, evaluates correctness, and assigns an A–G energy label. We demonstrate the approach on coding (LeetCode-style pass@1) and logical reasoning (exact-match accuracy) domains. Our tool, including a web-based leaderboard, is publicly available to encourage transparent reporting of LLM energy efficiency.**

I. Introduction

The rising popularity of LLMs and generative AI in general has contributed to the expansion of data centers and an increase in their energy demand [1]. Training a single large model can use hundreds of megawatt-hours [2], and inference adds up quickly when models are called millions of times per day [3]. Despite this increase in total energy usage, there is no standard way for users to compare models in terms of energy efficiency relative to their accuracy.

The EU energy label for appliances (Regulation EU 2017/1369) shows that simple letter grades (A to G) can help consumers make better choices [4]. We apply this idea to LLMs by introducing **Energy Per Correct Answer (EPCA)**: the total inference energy in Joules divided by the number of tasks a model gets right.

Existing work shows that larger models use more energy [5], but does not yet connect energy to accuracy. Our contribution is to close that gap. We build a benchmark that measures both, and we deliver a web-based tool that recommends the most energy-efficient model for a given task domain.

The contributions of this paper are:

1) **EPCA as a metric** that penalises both high energy use and low accuracy.
2) An **open-source benchmark pipeline** with real GPU energy measurement.
3) **A web UI** where users pick a domain (e.g. coding, logical reasoning) and see which model offers the best trade-off between accuracy and energy.

II. Related Work

Current work into measuring LLM energy usage focuses mostly on energy consumption and carbon emissions, leaving out energy efficiency [6]. Other attempts to create LLM efficiency benchmarks have related factors such as model size and architecture to inference energy efficiency, but have identified task accuracy as an area that can still be explored [7].

Strubell et al. [2] were among the first to quantify the energy cost of training large NLP models, reporting that training a single Transformer can emit as much CO_2 as five cars over their lifetimes. Patterson et al. [3] extended this analysis to large-scale training runs and highlighted that inference energy is increasingly dominant as models are deployed at scale. Luccioni et al. [5] measured the inference energy of several deployed models and showed that larger models consume more energy per query, but did not relate this to task accuracy.

The EU energy labelling framework (Regulation EU 2017/1369) [4] provides a proven model for communicating efficiency to consumers. Research has shown that energy labels shifted 85% of refrigerator sales to A-rated models within a decade, demonstrating that simple letter grades can drive adoption of efficient alternatives. Wiesner et al. [1] have explored carbon-aware quality adaptation for energy-intensive services, showing that sustainability-aware scheduling is feasible in practice.

Our work differs from all of the above by combining real energy measurement during inference with task-level accuracy into a single composite metric (EPCA) and mapping it to an A–G label. This bridges the gap between energy measurement research and practical model selection.

For energy measurement, we build on hardware-level approaches rather than utilisation-based estimation. Specifically, we poll NVIDIA GPU power draw via `nvidia-smi` and

integrate over time using the trapezoidal rule, and we read Intel RAPL counters for CPU energy on Linux. For accuracy evaluation, we adopt pass@1 from HumanEval [8] and MBPP [9] for coding tasks, and exact-match accuracy for reasoning and mathematics tasks.

III. Methodology

A. Pipeline Overview

Our pipeline has five steps:

1) Load domain-specific tasks from a dataset.
2) Send each task to a locally hosted LLM (via Ollama) and record the output.
3) Measure GPU power draw during inference using `nvidia-smi` polling.
4) Check correctness by running the output in a sandboxed environment.
5) Calculate EPCA and assign an A to G label.

B. Accuracy Metrics

The accuracy metric must be binary (pass or fail), deterministic, and appropriate for the domain. We pick one metric per domain:

1) Coding: pass@1: Each model gets one attempt per problem at temperature 0. The generated code runs in a sandboxed subprocess against test assertions. A task counts as correct only if all assertions pass. We use pass@1 because it ties a single inference call directly to an outcome, ensuring that the energy measured corresponds exactly to one attempt; it checks functional correctness rather than textual similarity to a reference solution; and it is the standard metric in HumanEval [8] and MBPP [9], making results directly comparable to published benchmarks.

2) Logical Reasoning: Exact-Match Accuracy: Each model answers a multiple-choice question. The answer letter is compared to the correct answer. This is standard in reasoning benchmarks and fully automatable.

3) Secondary Metrics: For coding tasks we also report runtime (ms) and peak memory (KB) of the generated solution. These are shown alongside the label but do not affect EPCA.

C. Energy Measurement

We use our `EnergyMonitor` module for hardware-level measurement instead of utilisation-based estimation.

GPU: A background thread polls `nvidia-smi` every 250 ms, recording power in watts and temperature. Energy in Joules is calculated by integrating the power readings over time using the trapezoidal rule.

CPU: On Linux with Intel CPUs we read the RAPL energy counter before and after inference. On systems without RAPL we fall back to `psutil` CPU usage as a proxy.

Controls: Models run in round-robin order to avoid GPU temperature bias. There is a 5-second rest between measurements for thermal cooldown. Each model gets a warm-up prompt before timing starts. All models run on the same machine.

D. EPCA and Labels

EPCA is simply the total energy (Joules) divided by the number of correct answers. A model that uses little energy but gets few answers right will score poorly. A model that gets everything right but uses a lot of energy will also score poorly. Only models that are both efficient and accurate get a good EPCA.

If a model solves zero tasks, EPCA is undefined and the model gets label G.

We map EPCA to labels using the thresholds in Table I. The scale follows a geometric progression, with each band representing an approximate doubling used in the EU energy labelling framework where each grade represents an approximate doubling in consumption. The absolute values (5, 10, 20, …, 160 J) were empirically derived from initial benchmark runs runs on 7B-parameter models, placing typical efficient models in the A–C range and inefficient or large models in the D–G range. In practice, absolute thresholds do not generalise well across domains with varying energy scales, Our final results therefore adopt a relative thresholding approach (described in Section V) that adapts to the actual distribution of EPCA values.

TABLE I
Energy Label Thresholds

Label	EPCA (J per correct answer)
A	≤ 5
B	$\leqslant 10$
C	$\leqslant 20$
D	$\leqslant 40$
E	$\leqslant 80$
F	$\leqslant 160$
G	> 160

E. Statistical Validation

We use bootstrap confidence intervals (1,000 resamples, 95% CI) to test whether EPCA differences between models are significant. We also compare our pass@1 scores against published leaderboard entries to check that our task suite gives consistent results.

IV. The Tool

A. Benchmark Pipeline

The benchmark is a Python CLI with three commands: `evaluate` (run models on tasks, measure energy), `score` (compute EPCA and labels from results), and `plot` (generate charts from results). It uses Ollama for local model inference and our `EnergyMonitor` for energy measurement.

B. Web UI

The main deliverable is a web-based leaderboard where users can:

1) Pick a task domain (e.g. Coding, Logical Reasoning).
2) See how accuracy is measured for that domain and why that metric was chosen.
3) View a ranked table of models with their EPCA, pass rate, energy, and A to G label.
4) Choose the best model for their use case.

The UI displays pre-computed benchmark results. Benchmarks run on a dedicated GPU server; the web app just serves the results.

A mockup of the web UI is shown in Fig. 1.

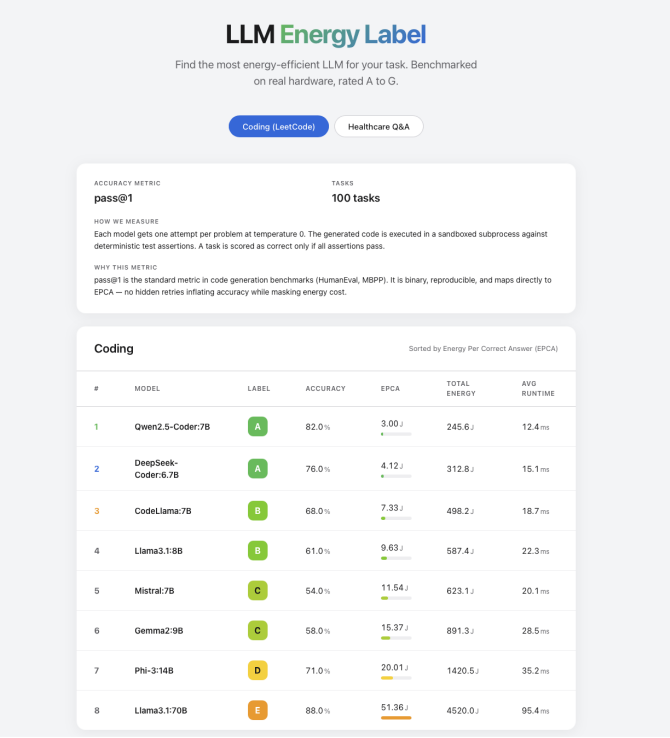

Fig. 1. Mockup of the web-based leaderboard. Users select a domain and see model rankings by EPCA with energy labels.

C. Adding New Domains

Adding a new domain takes three steps: (1) create a task dataset, (2) run the benchmark, (3) register the domain with its accuracy metric description. This makes the tool extensible to any domain where correctness can be checked automatically.

V. Experimental Setup

Eight models were evaluated via Ollama on a Linux GPU machine: `qwen2.5-coder:3b`, `qwen2.5-coder:7b`, `mistral:7b`, `llama3.1:8b`, `qwen3:4b`, `codellama:7b`, `qwen3:14b`, and `phi3:3.8b`. GPU energy was measured by polling `nvidia-smi` every 250 ms and integrating power readings with the trapezoidal rule; CPU energy was obtained from Intel RAPL counters. GPU energy accounted for 97–99% of total inference energy. Each domain was run for 30 iterations per task in round-robin model order to prevent thermal bias, with a 5-second rest between measurements and a warm-up prompt before each model's first timed call. Labels were assigned using a relative threshold system, in which the median EPCA across all models anchors the scale and thresholds are set at multiples of that value (0.40×, 0.65×, 0.90×, 1.30×, 2.00×, 3.50× for labels A through F respectively). This spreads labels across the actual distribution of results rather than relying on the fixed thresholds in Table I; all labels in the results below use the relative system. Where the relative system produces a skewed distribution (e.g. when one outlier model anchors the scale), fixed-threshold labels are also reported.

VI. Results and Discussion

A. Coding Domain

The coding benchmark used five tasks based on well-known LeetCode problems (Two Sum, Reverse Integer, Valid Parentheses, Merge Sorted Lists, Maximum Subarray). The prompts and test assertions were written by the authors; each task includes 3–6 deterministic assertions covering standard and edge cases (1,200 measurements: 30 iterations × 5 tasks × 8 models). Each model generated a Python function named `solve()`, which was executed pass@1 in a sandboxed subprocess; correctness was determined by majority vote across iterations. *reverse_integer* was failed by every model on every attempt (0/240) due to sign-character reversal errors, non-standard function names, and incorrect overflow handling. Its energy cost is included in EPCA but it contributes zero correct answers, raising the EPCA for all models.

The results are shown in Table II. `qwen2.5-coder:3b` achieved the best EPCA (415.7 J) and shares label B with `qwen2.5-coder:7b`. `mistral:7b` received label C and `llama3.1:8b` label D. `qwen3:4b` received label F; the remaining three models received label G.

Three observations follow from the results. First, three models (`qwen2.5-coder:3b`, `llama3.1:8b`, `qwen3:4b`) share a 60% pass rate but have EPCA values of 415.7 J, 909.7 J, and 3,121.4 J respectively, a 7.5× range. An accuracy-only ranking treats them as equal. Scaling from `qwen2.5-coder:3b` to `qwen2.5-coder:7b` improves accuracy by 20 percentage points at 1.86× the energy cost; both receive label B, but the EPCA increases from 415.7 J to 580.2 J.

Second, the two chain-of-thought models consume more energy than their parameter count suggests. `qwen3:4b` (4B

TABLE II
CODING DOMAIN RESULTS (30 ITERATIONS, 5 TASKS, PASS@1 MAJORITY VOTE)

Model	Pass Rate (%)	Energy/Task (J)	EPCA (J)	Label
qwen2.5-coder:3b	60	249.4 ± 45.7	415.7	B
qwen2.5-coder:7b	80	464.2 ± 69.8	580.2	B
mistral:7b	40	246.2 ± 104.5	615.5	C
llama3.1:8b	60	545.8 ± 72.0	909.7	D
qwen3:4b	60	1872.8 ± 895.3	3,121.4	F
codellama:7b	20	763.2 ± 1278.6	3,815.9	G
qwen3:14b	20	7489.6 ± 1004.1	37,447.9	G
phi3:3.8b	0	210.4 ± 39.4	—*	G

* EPCA undefined (zero correct answers); label G assigned automatically.

TABLE III
LOGIC DOMAIN RESULTS (30 ITERATIONS, 10 TASKS, EXACT-MATCH MAJORITY VOTE)

Model	Pass (%)	Energy/Task (J)	EPCA (J)	Label	Fixed
mistral:7b	60	0.2 †	0.33 †	A	A
qwen2.5-coder:3b	70	21.6	30.9	A	D
codellama:7b	30	22.1	73.6	B	E
qwen2.5-coder:7b	90	105.5	117.2	C	F
llama3.1:8b	50	86.5	173.1	D	G
phi3:3.8b	80	186.9	233.6	E	G
qwen3:4b	70	1,564.2	2,234.6	G	G
qwen3:14b	80	3,170.3	3,962.8	G	G

† GPU records 0 J (inference <250 ms); energy from CPU proxy only. Fixed label uses absolute thresholds from Table I.

TABLE IV
MATH DOMAIN RESULTS (30 ITERATIONS, 10 TASKS, EXACT-MATCH MAJORITY VOTE)

Model	Pass (%)	Energy/Task (J)	EPCA (J)	Label	Fixed
mistral:7b	30	0.2 †	0.77 †	A	A
qwen2.5-coder:3b	60	21.1	35.2	A	D
codellama:7b	50	26.9	53.8	B	E
phi3:3.8b	80	86.6	108.3	C	F
llama3.1:8b	60	85.4	142.4	D	G
qwen2.5-coder:7b	60	103.6	172.6	E	G
qwen3:4b	100	605.4	605.4	G	G
qwen3:14b	100	2881.6	2,881.6	G	G

† GPU records 0 J (inference <250 ms); energy from CPU proxy only. Fixed label uses absolute thresholds from Table I.

parameters) used 1,872.8 J per task, compared to 464.2 J for `qwen2.5-coder:7b` (7B parameters). The high variance of `qwen3:4b` (std = 895.3 J) reflects task-dependent chain length. `qwen3:14b` consumed 7,489.6 J per task and 1.12 MJ in total while passing only one task, giving an EPCA of 37,447.9 J.

Third, `phi3:3.8b` had the lowest per-task energy of all models (210.4 J) but passed zero tasks due to `SyntaxError` on every attempt. EPCA is undefined for it and the model receives label G automatically. This case shows why energy alone is not a useful metric.

B. Logic Domain

The logic benchmark used ten multiple-choice questions covering syllogisms, propositional logic, mathematical logic, and number-pattern problems (2,400 measurements: 30 iterations × 10 tasks × 8 models). Correctness was evaluated by matching the model's response against the expected answer letter. To address the possibility of random guessing on multiple-choice questions, each task was repeated 30 times and correctness determined by majority vote. A model guessing uniformly at random on a four-option question would pass at 25%, and the probability of a majority of 30 independent random guesses being correct is negligible. In practice, per-task pass rates were either 0/30 or 30/30 for all but one model–task pair, confirming that model responses are deterministic rather than random.

Table III shows the results. The relative threshold system spreads labels from A to G across models; fixed-threshold labels are also reported for comparison. `mistral:7b`'s EPCA (0.33 J) is notably low, but this is an artefact of a measurement limitation: `mistral:7b`'s responses completed in under 250 ms, below the `nvidia-smi` polling interval, so the GPU recorded 0 J and the reported energy (58.7 J total) comes from the CPU proxy only. The EPCA of 0.33 J should not be interpreted as a true energy advantage. A reduced polling interval is planned for future runs.

Two results are worth noting. `phi3:3.8b` scored 0% on coding but 80% on logic. This reversal shows that EPCA labels are domain-specific and that a model's performance on one task type does not generalise to another. `qwen2.5-coder:7b` achieved the highest pass rate across all models on logic (90%), including both qwen3 models despite being a standard-architecture model. The two chain-of-thought models used 1,564 J and 3,170 J per task, compared to 0.2–187 J for the other six models, a spread of approximately 16,000×, versus 30× in the coding domain.

C. Math Domain

The math benchmark used ten multiple-choice questions covering algebra, fractions, functions, geometry, and statistics (2,400 measurements: 30 iterations × 10 tasks × 8 models). Correctness was evaluated by the same majority-vote procedure as the logic domain. Two tasks (*algebra_1*, *algebra_2*) were solved by every model on every attempt; two (*fractions_2*, *statistics_2*) had a combined pass rate of 37.5% across all models.

Table IV shows the results. As in the logic domain, `mistral:7b` anchors the relative scale with an EPCA of 0.77 J, again due to the fast-inference measurement limitation described above: all responses completed in under 100 ms and GPU energy was not captured. Fixed-threshold labels are therefore also reported.

The math domain produces two results not seen in coding or logic. `qwen3:4b` and `qwen3:14b` are the only models

to achieve 100% pass rate in any domain. Because all ten tasks pass majority vote, their EPCA equals their mean energy per task (605.4 J and 2,881.6 J respectively); they still receive label G due to energy cost. `mistral:7b`'s pass rate drops to 30%, its lowest across all three domains, as it fails fractions, functions, and geometry tasks entirely. `phi3:3.8b` again scores 80%, consistent with its logic result and confirming that its 0% coding failure is specific to code generation rather than a general capability limitation.

VII. Conclusion

We introduced Energy Per Correct Answer (EPCA) as a composite metric that jointly captures inference energy cost and task accuracy, and mapped it to A–G energy labels inspired by the EU energy labelling framework. We implemented an open-source benchmark pipeline that measures real GPU energy consumption via `nvidia-smi` power polling and evaluates correctness through sandboxed code execution (pass@1) and exact-match accuracy.

Our experiments across three domains (coding, logical reasoning, and mathematics) with eight locally hosted models yielded three key findings. First, EPCA reveals efficiency differences that accuracy-only rankings hide: three models sharing a 60% pass rate on coding had EPCA values spanning a 7.5× range. Second, chain-of-thought models consume disproportionately more energy than their parameter count suggests, with energy spreads of up to 16,000× between the most and least efficient models. Third, energy labels are domain-specific: `phi3:3.8b` scored 0% on coding but 80% on logic and mathematics, confirming that a single label cannot represent a model across all tasks.

Our work has several limitations. The `nvidia-smi` polling interval of 250 ms is too coarse for models that respond in under 250 ms, leading to 0 J GPU readings for fast models on simple tasks. The task suites are small (5–10 tasks per domain) and may not fully represent domain difficulty. Energy measurements are hardware-specific and not directly comparable across machines.

Future work should address these limitations by adopting finer-grained energy measurement (e.g. NVIDIA NVML with sub-100 ms polling), expanding task suites to 100+ tasks per domain, and validating EPCA rankings across different GPU hardware. We also plan to extend the benchmark to additional domains such as summarisation and translation, and to explore how quantisation and other optimisation techniques affect the energy–accuracy trade-off.

The benchmark tool, including the web-based leaderboard, is publicly available at https://github.com/NCHWU/Energy_Label to encourage transparent reporting of LLM energy efficiency.

Acknowledgment

The authors would like to thank Trusted ID (https://www.trusted-id.eu) for providing access to GPU compute resources used in the experiments.

References

[1] P. Wiesner, D. Grinwald, P. Weiß, P. Wilhelm, R. Khalili, and O. Kao, "Carbon-aware quality adaptation for energy-intensive services," in *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*, ser. E-Energy '25. ACM, Jun. 2025, pp. 415–422. [Online]. Available: http://dx.doi.org/10.1145/3679240.3734614

[2] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.

[3] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," *arXiv preprint arXiv:2104.10350*, 2021.

[4] "Regulation (EU) 2017/1369 — energy labelling framework," 2017. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2017:198:TOC

[5] A. S. Luccioni, S. Viguier, and A.-L. Ligozat, "Power hungry processing: Watts driving the cost of AI deployment?" in *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, 2023, pp. 85–95.

[6] M. Özcan, P. Wiesner, P. Weiß, and O. Kao, "Quantifying the energy consumption and carbon emissions of LLM inference via simulations," 2025. [Online]. Available: https://arxiv.org/abs/2507.11417

[7] K. Pronk and Q. Zhao, "Benchmarking energy efficiency of large language models using vLLM," 2025. [Online]. Available: https://arxiv.org/abs/2509.08867

[8] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[9] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le *et al.*, "Program synthesis with large language models," *arXiv preprint arXiv:2108.07732*, 2021.