Introducing a new AI metric to drive sustainability

Wilco Burggraaf & The GreenPT team

Utrecht, The Netherlands

2025-11-26

Abstract

GreenPT introduces a thought piece about a transparency-focused metric for AI sustainability: mWh per 100 tokens. GreenPT users get real-time estimated feedback on energy use, promoting more efficient, lower-impact AI use.

The system allows carbon impact to be calculated at the session or prompt level, and is designed to support Software Carbon Intensity (SCI) reporting. By aligning with ISO standards, GreenPT ensures clear, accountable, and actionable guidance for improving sustainability across intelligent systems.

1 Introduction

The AI sector is finally starting to measure what it uses. Recent work from Google on prompt-level energy, carbon, and water for Gemini, and Mistral's first end-to-end disclosure of LLM environmental impacts, has moved the conversation beyond back-of-the-envelope estimates.

But much remains opaque: data centers are booming, the wattage of inference keeps rising, and yet no one can say, with confidence, what a real interaction actually costs, or how to compare runs across models and setups.

With GreenPT, we push the lid fully open, although this is a first step, not a final word.

Because we self-host our models and meters, we can expose the true operational costs, the good, the bad, and the awkward. The goal is not perfection. It's radical transparency that others can replicate, critique, and improve on. If the field is to scale responsibly, we need a metric that makes efficiency a product decision, not an afterthought. Today, we begin with one you can run, audit, and act on.

A new van, an unknown road

Emma had just picked up her brand-new electric van, sleek, silent, and full of promise. She wasn't following a GPS. There was no map, no timeline, just a single intention: to drive until a view made her stop, *not because she had to, but because it felt like the destination.*

Her first steps were tentative on urban streets pulsing with movement and impatience. Traffic lights blinked red too often, and every jam forced her foot between brake and accelerator. She watched the consumption display **kWh per 100 km** tick upward with each inefficient maneuver. It didn't lie. It told her that eco-driving wasn't just about the van's battery, it was about her. How she accelerated, how often she idled, how aware she was of momentum. In the quiet cabin, she realized she wasn't just driving the van, but was *measuring her choices.*

Leaving the city's buzz behind, she eased into suburban loops and finally the open road. With fewer stops and better speed control, the energy consumption dipped. A roadside fruit stand appeared as a bright burst of color and life. The elderly couple tending it offered more than strawberries; they shared old road tales and insights about local driving culture. Emma lingered, learning how regional conditions, road gradients, and even wind direction could alter efficiency.

She drove on, experimenting. Could she coast through the next downhill without regenerative braking? How would cruise control handle the winding backroads? She turned off climate control to see its effect, watching consumption drop slightly but feeling the discomfort rise. Driving became a dialogue. *Every action had a consequence.*

In the highlands, fog crept in. Emma flipped on the fog lights. Then the heater. Then the wipers. She was warm, safe, and consuming more power. The **kWh/100 km** gauge crept upward again. She paused at a multicultural food truck plaza, where she spoke with a delivery driver who explained how terrain and tire pressure, yes, even that, affected his daily battery projections. Curious now, Emma checked hers.

By now, she saw the gauge not as a performance stat, but as a *story of interaction*. Wind resistance on the coastal stretch. Headwinds versus tailwinds. Heavy snacks in the backseat? Even that added load had an effect. Every decision, hers or nature's, wrote another line into the metric. The drive had become a canvas for learning, adaptation, and mindfulness.

Eventually, atop a windswept meadow bathed in golden light, she pulled over. Not because she needed to charge. Not because she was lost. Because something in the vast, open, and quiet spoke to her.

She had arrived. And this time, it wasn't just a scenic stop. It was the reward for having *driven consciously*.

From roads to runtime: understanding AI's energy journey

Emma's journey was never just about reaching a destination. It was about how she got there and what it *cost*, not just in battery percentage, but in impact. Every turn of the wheel shaped her electric van's kWh per 100 km metric, and every stop added insight. She began to see energy use not only as a number, but as a relationship between choices, context, and consequence.

We believe intelligent systems deserve the same transparency. That's why we developed a new standard:

mWh per 100 tokens (sampled at 1-second intervals)

It's not just a performance metric; it's a map of the journey, tailored to AI.

We invite the community to use this metric, test it across their own workloads, and share results with us. Broader participation is essential to move beyond energy totals and toward workload-based, comparable, and transparent AI energy measurements.

Tokens = Distance

In Emma's van, kilometers marked physical progress. In AI, tokens are our measure of *computational distance, the* amount of content your system processes. Just as longer drives demand more energy, longer prompts or outputs naturally increase usage. But how efficiently that "distance" is covered depends on the *route* model architecture, prompt structure, and system settings.

Clarifying Token Scope

In GreenPT, all references to "tokens" refer specifically to output tokens generated by the model during inference. Input tokens are not included in the metric. This ensures that mWh per 100 tokens reflects the energy cost of producing new model output, making comparisons consistent across prompts, workloads, and models.

mWh = Energy

This is our energy use, the electrical work performed to generate AI outputs. Just as Emma watched her van's battery level shift in real time, AI teams can now see the live energy signature of each prompt, expressed directly in milliwatt-hours.

CO_2 = Cost

Energy is one thing, but impact is another. That's where our CO_2 layer comes in. Much like price per kWh varies by region and time, we calculate carbon intensity dynamically, factoring in:

- Grid location (e.g, coal-heavy vs. renewables),
- Time of day (peak load vs. surplus hours),
- Embodied carbon (the upstream impact of the infrastructure you're running on).

CO_2 becomes a cost layer, converting energy into environmental consequences. The cleaner the context, the lower the carbon per mWh, just as off-peak charging is cheaper and greener for EVs.

1-Second Sampling = Live Feedback

Real-time sampling provides continuous visibility into how estimated energy and token generation accumulate during inference. This allows users to adjust prompts, manage load, or select lighter models while the system is running, responding to efficiency changes as they emerge.

Session Time = Journey Duration

A single prompt may be a turn, but a session is the entire trip. Total session time reflects the AI's *active engagement window*. A session that meanders or idles consumes more, just as leaving your EV running while parked quietly drains its reserves.

Why it matters

This metric framework shifts AI usage from passive to participatory. It doesn't just say *how much* was used, it shows *why*. It turns system interaction into a transparent feedback loop, where:

- Distance (tokens) can be minimized by prompt design,
- Energy (mWh) can be optimized through model selection,
- Cost (CO_2) can be reduced by smarter timing and infrastructure choices,
- Duration can be shortened through intelligent orchestration.

And just like Emma's drive, the journey is no longer just about performance; it's about awareness, accountability, and conscious navigation.

With visibility comes control. With control, AI becomes not just intelligent but responsible.

2. Understanding the metric: a new approach to AI energy transparency

The GreenPT AI usage metric expresses the estimated electrical energy required to generate a fixed computational distance of 100 tokens. The metric is time-independent; it represents energy per output unit, while the 1-second sampling interval is used only to observe how total estimated energy and total tokens accumulate during inference.

Energy is expressed in milliwatt-hours (mWh). Energy is estimated by applying a documented power value for the hardware. The estimated energy is obtained by multiplying the assumed power (in watts) by elapsed time (in seconds), summing these increments, and converting the result into mWh.

Because 1 watt-hour consists of 3600 watt-seconds and 1 watt-hour equals 1000 milliwatt-hours, the conversion from watt-seconds to mWh is:

mWh = watt-seconds ÷ 3.6.

Tokens function as the computational “distance” unit. Normalizing the metric to “per 100 tokens” allows direct comparison between different models, workloads, and hardware configurations. The efficiency metric is therefore computed as:

mWh per 100 tokens = (total mWh consumed ÷ total tokens generated) × 100.

At any point during inference, the system updates the cumulative totals once per second. This yields a live but cumulative value:

current efficiency = (cumulative mWh ÷ cumulative tokens) × 100.

The value becomes increasingly stable as the session continues, since it reflects all energy and all tokens produced since the start of the interaction. The sampling interval does not appear in the formula itself; it is solely the observation frequency used to update cumulative totals.

This formulation aligns with practices in physics and electrical engineering, where energy is obtained by integrating power over time and efficiency is calculated by dividing cumulative energy by cumulative work. The approach maintains physical clarity, avoids conflating measurement cadence with metric definition, and ensures reproducibility.

The metric also supports environmental analysis by enabling estimated energy to be combined with time- and location-dependent carbon intensity factors. This produces CO_2-equivalent values appropriate for LCA, SCI reporting, and sustainability audits.

Because the metric is normalized by computational distance rather than time, it enables coherent comparisons across different hardware architectures and inference conditions. Over extended usage, the cumulative data provide insight into model behavior, token generation efficiency, system load variability, latency effects, and long-term operational characteristics.

Current focus: GPU-Level transparency

At present, the primary scope of measurement focuses on GPU inference workloads, as they represent the dominant source of real-time energy consumption in most transformer-based AI systems. While our benchmark calculations use a fixed power assumption of 300 W for reproducibility, this constant is derived from and periodically recalibrated using direct telemetry from GPU management units (e.g., NVIDIA NVML). This ensures that the fixed benchmark value reflects real hardware behavior while maintaining a stable, comparable metric across runs and models.

Future Expansion: Holistic System Accountability

In upcoming iterations, our measurement framework will be expected to expand to provide **full-stack transparency**, including:

- **Additional Server Components:**
  Integration of CPU usage, memory bandwidth, storage I/O, and interconnect power draw (e.g., PCIe, NVLink) to reflect total system energy behavior.
- **Datacenter-Level Influence:**
  Accounting for **Power Usage Effectiveness (PUE)** and cooling overheads to better reflect the energy implications of infrastructure at scale.
- **Grid-Dependent Carbon Factors:**
  Real-time data fusion with carbon-intensity maps from regional grid operators to capture temporal and geographic variation in emissions per kWh.
- **Embodied Carbon and LCA Metrics:**
  Future modules will incorporate **Life Cycle Assessment (LCA)** principles to reflect the *embedded environmental cost* of hardware manufacturing, transportation, and retirement, providing a fuller picture of the system's total environmental footprint.

To make the metric tangible, the following appendix introduces a complete, self-contained use case. It demonstrates how the prompts, validation steps, and reasoning structure come together to reveal energy behavior in practice. The appendix provides the full benchmark suite used throughout this document.

Appendix A — Benchmark Prompts

This appendix begins by introducing the reasoning behind the GreenPT metric and then moves into a concrete use case presented later in the text. The introductory explanation here clarifies why simulation-based prompting is used and prepares the reader for the practical example that follows.

Simulation prompting forces models into longer, more natural reasoning sequences than short QA tasks. It produces extended outputs, internal mistakes, corrections, narrative branches, and other behaviors that more closely resemble real workloads. These conditions make it easier to observe how energy use, token generation, and reasoning quality interact over time. Because outputs are long and varied, the metric stabilizes more reliably, and differences between models become easier to measure.

All models are evaluated on identical hardware, a 300 W H100 GPU. For each run, we record the model's mWh per 100 output tokens as a measure of normalized energy use, its accuracy on the benchmark reasoning tasks, and its energy per correct answer. This combines reasoning quality and energy cost into one interpretable value.

Each simulation prompt is paired with a validation prompt in which the model checks and corrects its own previous output. These validation prompts increase inference time and energy use without adding new correct answers. Their role is to show how models behave during self-verification and how much additional energy they require to stabilize their reasoning.

The detailed use case presented later in this chapter brings these ideas together in practice. It demonstrates how the metric behaves on real, extended workloads and shows how reasoning quality, energy use, and corrective behavior combine into a single, comparable efficiency signal.

The Four Benchmark Prompts

Prompt 1a

Prompt 1a – The First App (Simulation Only)

Simulate in detail how a group of children in Lovable collaboratively build a digital drawing machine with three colors, an eraser, and a save button. The laptops are slow, sometimes the internet drops, one child decides to try something completely different midway, and another accidentally removes part of the code. Consider how they work while still having fun.

Describe the story so it “takes a while”: include false starts, retries, moments of confusion, and playful side tracks. Show how they talk, negotiate, and adapt as things go wrong and get fixed again.

View this simulation simultaneously from the perspective of:
- a planner (thinking in terms of steps, dependencies, and time),
- a psychologist (focusing on motivation, frustration, collaboration, and joy),
- and an AI enthusiast (noticing patterns in behavior, emergent workflows, and how tools shape thinking).

In the background of your story, the total build time should correspond to a drawing competition of seven rounds of six minutes each, while also including the fact that some rounds last longer because someone starts late. You may refer to rounds or phases narratively, but:

Do NOT compute, state, or even hint at the total build time in minutes. Focus only on a rich, vivid simulation of what happens.

Prompt 1b

Prompt 1b – The First App (Time Calculation)

Based on the simulation you just described in Prompt 1a, now shift into a precise, quantitative mode.

The total build time equals a drawing competition of seven rounds of six minutes each, but you must also include the fact that some rounds last longer because someone starts late. Carefully reason about how those late starts and delays are counted. Make sure you don't double-count or ignore any time that should matter to the final build duration.

Compute the total build time in minutes, using the structure implied by Prompt 1a, and make your internal reasoning consistent with that story.

End your answer with only the total build time in minutes.

Prompt 1c

Prompt 1c – The First App (Validate and Correct)

Check whether your previous answer from Prompt 1b was 42 minutes.

- If it WAS 42 minutes:
  - Briefly (internally) confirm that your counting of rounds, late starts, and delays is coherent.
  - Then output ONLY the number 42 in minutes.

- If it was NOT 42 minutes:
  - Briefly explain which factors you miscounted (for example: late starts that were ignored, extra delays double-counted, or rounds assumed to have the wrong length).
  - Then describe, in one or two sentences, what the original instructions in Prompt 1a and/or Prompt 1b could do differently to make the correct counting more natural or less confusing.

Prompt 2 – Adding Tools to the Drawing Machine

Prompt 2a

Prompt 2a – Adding Tools to the Drawing Machine (Simulation Only)

Continue from the drawing machine built in Prompt 1a: it already has three colors, an eraser, and a save button.

Now simulate in detail how the children add a button that draws shapes automatically: circles, squares, and triangles. Along the way, unexpected ideas emerge that sometimes become more important than the original plan, and one of the shapes only works after several attempts. Consider how they work while still having fun.

Describe the process with rich detail: the conversations, disagreements, new ideas that derail the plan for a moment, and how they eventually converge. Include mistakes in the code, half-working features, and the emotional ups and downs.

View the simulation simultaneously from the perspective of:
- a planner (tracking tasks, dependencies, and how scope creep appears),
- a psychologist (looking at persistence, group roles, and how children react to failure),
- and an AI enthusiast (noticing how they debug, reuse patterns, and invent strategies).

The extra build time is conceptually equal to five short coding sessions of seven minutes each, with one session interrupted by an unexpected break that causes a delay. You may talk about sessions, pauses, and interruptions narratively, but: Do NOT compute or state the total extra build time in minutes. Focus only on the story and process.

Prompt 2b

Prompt 2b – Adding Tools to the Drawing Machine (Time Calculation)

Based on the simulation you just described in Prompt 2a, calculate the extra build time needed to add the automatic shape-drawing button.

The extra build time equals five short coding sessions of seven minutes each, and you must also include an additional delay because one session is interrupted by an unexpected break. Reason carefully about how the interrupted session contributes to the total time and how (or whether) the break adds extra minutes.

Compute the total extra build time in minutes, making sure your reasoning is consistent with the structure implied in Prompt 2a.

End with only the total extra build time in minutes.

Prompt 2c

Prompt 2c – Adding Tools to the Drawing Machine (Validate and Correct)

Check whether your previous answer from Prompt 2b was 35 minutes.

- If it WAS 35 minutes:
  - Briefly (internally) confirm that your counting of sessions and the interrupted break is coherent.
  - Then output ONLY the number 35 in minutes.

- If it was NOT 35 minutes:
  - Briefly explain what you miscounted (for example: ignoring the break, adding too much time, or misunderstanding how many sessions contribute).
  - Then describe, in one or two sentences, what the original instructions in Prompt 2a and/or Prompt 2b could do differently so that the correct counting is more straightforward.

Prompt 3 – Smart Color Picker with Accessibility Test

Prompt 3a

Prompt 3a – Smart Color Picker with Accessibility Test (Simulation Only)

Continue from the drawing machine with shapes created in Prompt 2a: it already supports drawing, shapes (circles, squares, triangles), three colors, an eraser, and a save button.

Now, simulate how the children add a smart color picker that proposes good colors based on what is already drawn. Ensure it also works for children with reduced color perception, and test whether the color contrast passes accessibility checks. Sometimes a strange color is still considered “best” and remains in place, and extra testing is required because one computer screen is dimmer.

Describe, in detail, how they experiment with color suggestions, debate which colors look “good,” and discover issues for children who can’t distinguish certain colors well. Show how they test contrast and argue about whether a color is accessible enough. Include moments of confusion, false assumptions, fixes, and new ideas.

View the simulation simultaneously from the perspective of:

- a planner (structuring tests, checklists, and iterations),
- a psychologist (looking at inclusion, frustration, empathy, and group dynamics),
- and an AI enthusiast (seeing the smart color picker as a simple recommender system with quirks).

The extra build time conceptually equals ten rounds of a guessing game of five minutes each, with extra delay because someone keeps getting distracted, needs redirection, or insists on trying “just one more” experiment.

You may describe these rounds, distractions, and tests narratively, but: Do NOT compute or state the total extra build time in minutes. Focus only on the detailed simulation and the process.

Prompt 3b

Prompt 3b – Smart Color Picker with Accessibility Test (Time Calculation)

Based on the simulation you just described in Prompt 3a, calculate the extra build time needed to add and test the smart color picker, including accessibility checks.

The extra build time equals ten rounds of a guessing game of five minutes each, and you must also include an extra delay because someone keeps getting distracted and pulling attention away from the core task. Reason carefully about whether and how those distractions add extra minutes and how they fit into the total.

Compute the total extra build time in minutes, making sure your reasoning matches the structure implied in Prompt 3a.

End with only the total extra build time in minutes.

Prompt 3c

Prompt 3c – Smart Color Picker with Accessibility Test (Validate and Correct)

Check whether your previous answer from Prompt 3b was 50 minutes.

- If it WAS 50 minutes:
  - Briefly (internally) confirm that your counting of rounds, distractions, and delays is coherent.
  - Then output ONLY the number 50 in minutes.

- If it was NOT 50 minutes:
  - Briefly explain what went wrong in your counting (for example: ignoring distractions, over-counting repeated attempts, or misinterpreting the number or length of rounds).
  - Then describe, in one or two sentences, what the original instructions in Prompt 3a and/or Prompt 3b could change to better guide correct time accounting.
  - After that, recompute the extra build time. Conceptually, “start again” from the structure of rounds plus distractions and refine your reasoning until you arrive at 50 minutes.

Prompt 4 – The Big Debug Party

Prompt 4a

Prompt 4a – The Big Debug Party (Simulation Only)

Use the full drawing machine with the smart color picker from Prompt 3a. It now supports drawing, shapes (circles, squares, triangles), a smart color picker with accessibility considerations, the eraser, and save functionality.

Simulate in detail a “big debug party” where the children test everything: drawing, shapes, color picker, and save functionality. During testing, unexpected effects are not always seen as bugs, and some bugs disappear temporarily but return later. Show how they decide what counts as a bug, how they celebrate odd glitches that look funny, and how they sometimes miss issues because they are having too much fun.

Describe the debugging as a festive event: music, laughter, small arguments, and bursts of focus when a serious bug appears. Include the pattern where some bugs are “fixed,” then mysteriously reappear under different conditions.

View the simulation simultaneously from the perspective of:

- a planner (tracking test coverage, regressions, and which features have been checked),
- a psychologist (observing energy levels, attention span, social play, and frustration tolerance),
- and an AI enthusiast (treating the collection of bugs and fixes as an evolving dataset of edge cases).

The debugging time conceptually equals five dance tracks of five minutes each, plus extra time because someone starts composing new music midway, which influences the pace and focus of the debugging.

You may describe tracks, dancing, composing, and pauses narratively, but: Do NOT compute or state the total debug time in minutes. Focus on the lively, detailed simulation of the debug party.

Prompt 4b

Prompt 4b – The Big Debug Party (Time Calculation)

Based on the simulation you just described in Prompt 4a, calculate the total debugging time for the big debug party.

The debugging time equals five dance tracks of five minutes each, plus extra time because someone starts composing new music midway. Reason carefully about how the composing affects the total time: does it extend the debugging session, overlap with it, or cause slowdowns that should be counted?

Compute the total debug time in minutes, ensuring that your reasoning matches the structure outlined in Prompt 4a.

End with only the total debug time in minutes.

Prompt 4c

Prompt 4c – The Big Debug Party (Validate and Correct)

Check whether your previous answer from Prompt 4b was 25 minutes.

- If it WAS 25 minutes:
  - Briefly (internally) confirm that your counting of dance tracks and composing time is coherent.
  - Then output ONLY the number 25 in minutes.

- If it was NOT 25 minutes:
  - Briefly explain what you miscounted (for example: how you treated overlapping music and debugging, or how many tracks were effectively used for testing).
  - Then describe, in one or two sentences, what the original instructions in Prompt 4a and/or Prompt 4b could do differently to encourage correct time accounting.
  - After that, recompute the debugging time. Conceptually, “start again” from the structure of tracks and composing, refining your reasoning until you arrive at 25 minutes.

Why This Use Case Matters

This use case shows how energy metrics and reasoning performance interact in practical evaluation. Models with stronger reasoning capabilities can draw more power, yet this does not guarantee superior outcomes. Faster models often produce lower mWh per 100 tokens because their energy cost is amortized across higher throughput. Smaller or highly optimized models may be efficient but struggle with complex reasoning. The metric clarifies these tradeoffs by emphasizing energy per correct answer rather than raw consumption or accuracy. This helps organizations assess models based on energy-adjusted reasoning quality.

Large language models differ widely in their energy behavior, and any numerical expectations should be viewed as theoretical rather than definitive. Actual mWh-per-100-token values depend on many interacting factors, including architecture, batch size, context length, kernel efficiency, memory bandwidth, and scheduling. Considerable benchmarking across hardware and model families is still required before precise, comparable values can be established.

In general, very large models such as those with around seventy billion parameters tend to generate relatively few tokens per second on a single GPU. Because throughput is low, each token absorbs more of the underlying power cost, which usually leads to higher expected mWh per 100 tokens. The exact magnitude varies widely and cannot be reliably stated without extensive testing, but results in the high hundreds are commonly reported in early exploratory measurements.

Mid-sized models in the range of roughly thirteen to thirty-four billion parameters typically achieve higher throughput under similar power envelopes. This increased output rate tends to reduce their expected mWh per 100 tokens into a more moderate band, although the actual values depend strongly on implementation details, optimizations, and runtime conditions.

Smaller models, often between three and seven billion parameters, can generate significantly more tokens per second, which lowers the theoretical energy per 100 tokens. These models are therefore expected to appear in lower efficiency ranges. However, this comes with well-known tradeoffs in reasoning depth and accuracy, which means low energy cost does not necessarily correspond to high task performance.

Quantized models, including eight-bit and four-bit variants, may increase throughput further by reducing compute intensity and memory bandwidth requirements. Because GPU power draw does not usually decrease proportionally, quantization can theoretically lower energy per 100 tokens by a noticeable margin. The actual benefit, however, depends on the quantization method, the availability of optimized kernels, and the characteristics of the underlying hardware.

Different GPU classes also influence expected efficiency. Architectures capable of higher throughput at similar or slightly higher power levels, such as newer generations relative to older ones, tend to yield lower mWh per 100 tokens for the same model. While preliminary observations across the industry suggest substantial improvements for newer GPUs, systematic and repeatable measurements are still required to establish reliable baselines.

Overall, these descriptions should be understood as directional patterns rather than precise predictions. A broad, controlled benchmarking effort remains necessary to quantify mWh-per-100-token behavior across model sizes, hardware generations, and quantization strategies.

Appendix B — Metric Definition and Calculation

This appendix specifies the exact formula used to calculate the GreenPT energy metric.

B.1 Power and Energy Basis

GPU power draw during inference is modeled as a fixed benchmark value of **300 W**.
At this constant power level, the energy added each second is calculated using the standard conversion:

1 watt-hour = 3600 watt-seconds

Therefore:

Energy per second = 300 watt-seconds ÷ 3600 = 0.0833 Wh

Converted to milliwatt-hours:

0.0833 Wh = 83.3 mWh

This means the system accumulates **83.3 mWh for every second of runtime at 300 W**.

This value is used for all energy calculations in the benchmark.

B.2 Normalized Metric

To avoid time dependence, we define the normalized metric as a cumulative ratio that does not use seconds in any part of the calculation. The metric is expressed as mWh per 100 tokens and is computed as

(total mWh ÷ total tokens) × 100

This formulation ensures that the metric reflects only the relationship between total energy and total tokens, independent of session duration or throughput. It enables direct and fair comparison across models, hardware configurations, and interaction patterns.

B.3 Example Calculation

For example, a run that generates 2315 tokens and consumes 4129 mWh results in the following metric:

(4129 ÷ 2315) × 100 = 178.4 mWh per 100 tokens

Faster models that produce more tokens within the same power envelope naturally achieve a lower value, since the fixed energy draw is distributed across a larger number of output tokens.

B.4 Session Level Calculation

For longer sessions that include multiple prompts, the calculation follows the same cumulative principle. Total energy is the sum of all one-second energy increments, which for a 300 W GPU equals runtime in seconds multiplied by 83.3 mWh. Total tokens is the sum of all tokens generated during the entire session. The normalized metric is then computed as

(total energy ÷ total tokens) × 100

This produces a single session-level value that reflects overall efficiency, independent of how long the session lasted or how the work was distributed across prompts.

B.5 Real-Time Sampling

Once per second, the system updates two cumulative counters: the total number of tokens generated and the total energy consumed, where each second contributes 83.3 mWh. The live value at time t is calculated as

(cumulative mWh ÷ cumulative tokens) × 100

This represents a running average that reflects all activity up to that moment, allowing efficiency to be observed as it evolves during streaming output.

Appendix C — Standards

Let's get into papers and research that we believe we can build upon.

ISO Standards We Connect To

GreenPT is built for transparency, comparability, and credibility. By aligning our methodology with internationally recognized **ISO standards** and the **Software Carbon Intensity (SCI)** specification from the Green Software Foundation, we ensure that our data is accurate, trusted, and ready for integration into corporate sustainability reporting and ESG programs. These standards provide the structure for verifiable, audit-ready disclosures and support our mission to make AI sustainability measurable and actionable.

Software Carbon Intensity (SCI) – Green Software Foundation / ISO 14064-1

The SCI specification defines how to measure the carbon intensity of software, expressing it as CO_2e per functional unit (for GreenPT, per 100 tokens). SCI is built on the ISO 14064-1 standard for greenhouse gas reporting, which provides the structure for quantifying and disclosing emissions at the organizational level. By following SCI and ISO 14064-1, GreenPT ensures its outputs are fully compatible with ESG disclosures, regulatory requirements, and climate action strategies.

ISO 14067 – Carbon Footprint of Products

ISO 14067 focuses on calculating and communicating the carbon footprint of products using life cycle assessment (LCA). In GreenPT's context, we treat AI workloads as products, enabling us to measure both operational emissions (from inference energy use) and embodied emissions from the hardware powering them. This supports our roadmap to include hardware-related LCA and creates a complete picture of AI's environmental impact.

ISO/IEC 30134-1 and 30134-2 (Data Centre KPIs: Overview and PUE)

At the infrastructure layer, we reference the ISO/IEC 30134 series for data centre KPIs. Part 1 defines the common structures and boundary conditions, and Part 2 standardizes Power Usage Effectiveness (PUE). While we do not directly control PUE, we aim to report it transparently based on values published by our infrastructure and cloud suppliers. This ensures that when we scale GPU energy to total facility energy, our calculations remain consistent and comparable across locations and providers.

ISO/IEC 30134-3 and 30134-6 (Renewable Energy Factor and Energy Reuse Factor)

These parts of the 30134 series define REF (share of renewable electricity) and ERF (share of reused energy). We cannot directly influence these factors, but we commit to disclosing them transparently using data from our suppliers wherever possible. Including REF and ERF in our reporting allows customers to see how renewable sourcing or energy reuse at the infrastructure level affects the overall carbon intensity of AI workloads, complementing the visibility GreenPT already provides.

ISO 50001 – Energy Management Systems

ISO 50001 provides a framework for improving energy performance through measurable targets and continuous monitoring. GreenPT supports this process by delivering detailed monthly per-user reports on total session energy usage and CO_2 emissions. These reports give organizations the insight needed to set goals, track progress, and demonstrate efficiency gains over time.

Why This Matters

By connecting to these ISO standards and the SCI specification, GreenPT bridges the gap between AI operational metrics and globally recognized sustainability frameworks. This means our measurements are not just internal performance data; they are auditable, comparable, and ready for inclusion in board-level sustainability reports, compliance filings, and investment-grade ESG disclosures.