ADORA FOUNDATION

INSPIRING VISION • CREATIVE CONNECTION • TRANSFORMATIONAL ACTION

Token Optimization & The Future of Sustainable AI

Why token-aware design, agentic governance and energy-weighted metrics must sit at the heart of any responsible AI strategy.

EXECUTIVE OVERVIEW

Artificial intelligence has crossed an invisible threshold. AI is no longer a set of isolated models; it is an **always-on, agentic infrastructure** that silently reshapes internet traffic, data-centre demand, and environmental impact.

Recent analysis finds that bots now account for **over half of all global internet traffic**, with automated and AI-powered bots reaching roughly 51% of web activity in 2024. At the same time, Google reports that its AI systems process around **1.3 quadrillion tokens per month**, signalling unprecedented compute scale.

This report outlines how organisations can respond by:

- Adopting **TokenOps**: treating tokens and energy as first-class design constraints.
- Re-architecting systems around **inspectable scaffolds** (prompts, tools, agents, RAG flows).
- Deploying **energy-weighted tokens** (“e-tokens”) as the basis for AI sustainability accounting.

1. From Models to Systems: Where the Real Impact Lives

Public debate still treats “the model” as the primary object of concern. In practice, large language models are wrapped in a dense software scaffold:

- **System prompts** of thousands of tokens injected into every request.
- **Conversation memory** that replays previous interactions.
- **Retrieval layers** that fetch documents, web pages and snippets.
- **Tooling** (e.g. search, code execution, databases, MCP servers).
- **Agent frameworks** orchestrating multiple specialised actors.

These layers are more tractable, auditable and corrigible than model internals—yet they are rarely measured or governed. The result is a growing gap between how organisations *perceive* their AI footprint and how it behaves in production.

2. Agentic AI and the Coming Compute Cliff

Traditional models behave like sprinters: trained once at huge cost, then queried occasionally. Agentic systems behave more like **always-on teams**, continuously planning, acting and coordinating.

> *“Plan and book my three-city work trip plus a low-carbon family holiday add-on.”*

A single instruction like this can trigger hundreds of invisible invocations. Even modest estimates suggest that agentic workflows create 10–100× more token usage than single-shot interactions.

3. The Inference Iceberg: Why Tokens Mislead

THE NON-EQUIVALENCE PROBLEM

Most reporting today cites “tokens in, tokens out” as proxies for cost and carbon. This hides a fundamental asymmetry:

- Input tokens (prefill) are often cheaper per token than output tokens (decode).
- But long contexts make prefill dominate energy use due to quadratic attention costs.
- Additional invisible layers (tool calls, RAG, agent messages, logs) add compute that is never reflected in the user-visible token count.

We call this gap between visible input/output tokens **the Non-Equivalence Problem**: the disconnect between the tokens we count and the compute we actually consume.

The Hidden Input Token Configs and Injections

Even before a single “visible” user token is processed, infrastructure choices can silently inflate the true input. Serving stacks are tuned for large context windows and rich chat templates, so every request carries configuration overhead and automatic prompt injections.

- **Config:** the serving configuration that governs how much context the model can see.
  - **Maximum context length:** the upper bound on total tokens (system + tools + user + history) that the model will accept. Infrastructure allocates KV cache and memory for this full capacity, even when a given user prompt appears short.
  - **Chat template design:** the wrapper that turns a conversation into model input (roles, separators, metadata). A verbose template can consume hundreds of tokens before any user text is added, reducing room for business context and increasing baseline energy per request.

  Together, these limits drive KV cache sizing, batching strategy and memory allocation, which is why “small” prompts can still be expensive when contexts are configured to be huge.

- **Injections:** additional text automatically added to every request before it reaches the model.

- **System prompts:** long instruction blocks that set behaviour, safety posture and tone for the assistant.
- **Chat history:** previous user and assistant turns replayed to maintain continuity, often far exceeding the size of the current input box.
- **Stored sub-prompts:** conditional rules, safety instructions, style guides and routing hints that are spliced in based on product state.
- **Tool scaffolding:** RAG passages, search snippets, tool schemas and MCP metadata that describe capabilities and results for downstream tools.

None of these injections are visible to the end user, but every token is processed by the model and therefore counted in energy and latency terms.

The Hidden Output Volume Configs

How much text the model chooses to emit is also governed by configuration rather than user intent. The same 30-token question can generate a terse 80-token answer or a sprawling 800-token essay, depending on how "helpful" and "detailed" the system has been instructed to be.

- **Thinking steps:** how much intermediate reasoning the system encourages the model to perform.
  - **Explicit chain-of-thought:** prompts that ask the model to "think step by step" or "show your reasoning" can multiply output length compared with a direct answer.
  - **Hidden traces:** some setups generate reasoning internally (for verification or tool selection) but only show a polished summary to the user. The hidden trace still incurs full decode cost.

- **Sampling and verbosity:** settings that bias the model towards longer or shorter responses.
  - **Sampling parameters:** higher temperature or broader top-p sampling tend to produce wordier, more exploratory responses, increasing tokens per answer.
  - **Verbosity instructions:** system-level guidance like "be comprehensive", "explain in detail" or "provide multiple options" pushes the model to emit more text for the same user question.

The Hidden Token Compute Multipliers

Agentic systems multiply these effects. Instead of one model call, a user task may trigger an orchestrator, several planner and worker agents, and a set of helper models for routing, filtering and tool selection. Each of these components carries its own prompts, history and tool metadata.

- **Agentic architectures:** multi-step workflows composed of specialised agents.
  - **Orchestrators:** top-level controllers that decompose tasks, call tools and hand work to specialist agents.
  - **Planner / worker agents:** “researcher”, “writer”, “reviewer” roles that each generate their own prompts, drafts and summaries.
  - **RAG and context passing:** every hand-off may include RAG snippets, partial plans and previous answers, all of which must be re-tokenised as input for the next step.

- **Helper models:** smaller models or classifiers that support the main LLM.
  - **Intent and routing models:** decide whether to call an LLM at all, which tool to use, or which agent should handle the request.
  - **Ranking and filtering models:** score search results, RAG passages or candidate answers, often running on the same tokenised text as the main model.
  - **Tool discovery (e.g.**

    ```
    find_tool
    ```

    ): choose among many MCP tools or APIs based on textual descriptions and past usage.

  These helper calls are rarely surfaced in top-level dashboards, but from an energy perspective they behave like miniature LLM requests attached to every user interaction.

The Hidden Non-Inference Compute

Finally, not all impacts are token-linked at all. Each request can trigger a cascade of conventional compute that never passes through the LLM context window but still consumes energy and water.

- **Data and logs:** the observability layer that records what happened.

    - **Raw events:** prompts, responses and tool calls written to log stores for debugging and audits.
    - **Derived analytics:** aggregation jobs that compute metrics, traces and anomaly signals on top of those logs.
    - **Storage and retention:** long-lived archives kept for compliance or research, with associated storage and access costs.
- **Downstream systems:** everything the LLM triggers outside the model boundary.
    - **APIs, databases and caches:** application logic that executes queries, updates records or fetches data in response to model decisions.
    - **Knowledge maintenance:** batch jobs that re-index RAG corpora or refresh embeddings as content changes.
    - **Monitoring and alerting:** systems that inspect traces and metrics to enforce SLAs, safety and governance rules.

All of this “shadow compute” is ultimately triggered by prompts and tokens, but it is not visible if we only look at input/output token counters.

VISIBLE TOKENS VS HIDDEN LAYERS

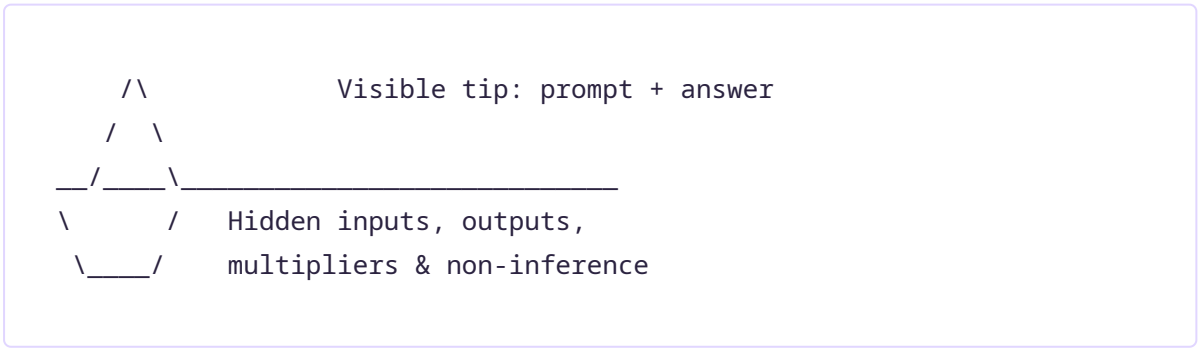
Visible tokens – what the user sees: prompt + final answer.

Hidden inputs – configs & injections (system prompts, history, tools).

Hidden outputs – verbosity, chain-of-thought, reasoning traces.

Hidden multipliers – agents, helper models, routing and tool brokers.

Hidden non-inference – logs, storage, downstream systems and batch jobs.



The metrics and academic focus on output/decode over inputs/prefill will result in almost every case, in a massive underestimate of true environmental

impact.

4. Metrics: From “Per Query” to Energy per N Tokens

Metrics

We need a better metric than "Wh per query" or "CO2 per prompt". We must resist "per query" becoming the accepted standard for disclosure. It is opaque, unverifiable, and fails to capture the agentic reality.

Traditional "per query" metrics are so coarse as to be practically useless. A short, multi-step request that fans out across tools, searches and planning can consume more energy than a long, single-shot analysis of a single document.

Because tokens are the common currency across text, images, audio and video, Adora Foundation treats **energy per N tokens** (for example, energy per 1,000 tokens) as the base unit. Input-normalised and output-normalised e-tokens make measurements comparable while reflecting full inference energy.

Energy-Weighted Tokens (e-Tokens)

A more useful basis for comparison is to measure the **real energy per request from input to output** and normalise it to tokens. An energy-weighted token, or **e-token**, is defined as:

> *“Total inference energy (Wh), amortised per 1,000 tokens using either input or output tokens as the denominator.”*

- **e-token (input)** – total energy per 1,000 input tokens.
- **e-token (output)** – total energy per 1,000 output tokens.

Both refer to the same physical energy but give complementary perspectives:

- Input-normalised for **budgeting and user-facing APIs**.
- Output-normalised for **model and hardware efficiency comparisons**.

Budgeting with e-Tokens

At system level, each application should have an explicit emissions budget derived from the volume of tasks it is expected to handle, the energy per N tokens of its architectures, and the carbon intensity of the regions where it runs. Instead of counting "queries per month", TokenOps teams work backwards from an acceptable emissions envelope and ask how many e-tokens they are willing to spend.

In practice, this means treating e-tokens as a design budget. Product and platform teams allocate that budget across features, decide which ones must be energy-frugal, and reserve headroom for genuinely high-impact use cases.

Rebound and Net Impact

Efficiency gains do not automatically translate into lower emissions. Cheaper and faster systems can increase usage and drive total energy up – the classic rebound effect. For any new AI feature, teams should explicitly ask:

- Does this feature replace a higher-energy process (for example, travel, manual review or physical prototyping)?
- Does it create new demand that would not exist without the AI capability?
- How might improved convenience change user behaviour and traffic volume over time?
- What is the net change in end-to-end system energy once external processes are included?

TokenOps governance connects these questions back to e-token budgets and emissions targets rather than celebrating local efficiency improvements in isolation.

5. TokenOps Maturity Model

TokenOps is the practice of treating tokens, energy and latency as first-class design variables alongside accuracy and UX. It asks teams to architect systems so that every major decision – model size, context window, memory pattern, agent depth and hardware – has an explicit cost in e-tokens.

The maturity model below describes how organisations evolve from ad-hoc token tracking to fully autonomous optimisation, where systems automatically choose the lowest-energy pathway (rules, SLMs or LLMs) that still meets user and safety requirements.

Level	Profile	Typical Behaviours
LEVEL 1	Unaware	No systematic tracking. “Per-N-token cost” used as sole budgeting metric.
LEVEL 2	Token-Aware	Basic counting. Teams watch dashboards but don't connect to architecture.
LEVEL 3	Token-Managed	Policies and guardrails. Curated tool lists, agent caps, smaller model experiments.
LEVEL 4	Token-Optimised	e-tokens in design. Automatic pruning of prompts/tools. Portfolio of SLMs/LLMs.
LEVEL 5	Token-Autonomous	Systems dynamically choose lowest-energy pathway (rules vs SLM vs LLM).

6. Principles of Token Optimisation

PRINCIPLE 1 — RIGHTSIZE YOUR MODELS

Match model complexity to the task. Larger is not always better: many workloads can be served by compact or specialised models that deliver most of the quality at a fraction of the energy.

Treat model capacity as a scarce resource. First define what "good enough" looks like for accuracy, latency and safety, then choose the smallest model family that can reliably achieve it.

- **Task analysis first:** define accuracy, latency, and cost thresholds.
- **Benchmark energy per 1,000 tokens (e-tokens)** alongside accuracy.
- **Orchestration:** use multiple models, not a single monolith.

PRINCIPLE 2 — MAKE LLM CALLS A LAST RESORT

Every token has a carbon cost. Exhaust lightweight, non-LLM solutions before invoking large, general-purpose models.

1. Rules, database queries and other deterministic code paths.
2. Traditional machine learning models.
3. Keyword or semantic search over well-structured content.
4. Compact or specialist language models.
5. High-capacity, general-purpose language models.
6. Multi-step agentic chains that coordinate many calls and tools.

Architect your stack so that most traffic is handled in the upper steps of this ladder, with only a small fraction reaching the most energy-intensive paths.

PRINCIPLE 3 — OPTIMISE CONTEXT WINDOWS AGGRESSIVELY

Context size grows energy use quadratically. Only send what is needed.

- **Context budget:** treat the available context window as a scarce budget; avoid filling it unless full-document reasoning is clearly required.
- **Lazy Loading:** Load tool descriptions/schemas lazily (e.g., MCP tools).
- **New chat per topic:** prefer starting fresh conversations over endlessly appending to giant threads.

- **Dynamic pruning:** drop low-value history based on relevance scoring rather than replaying everything.
- **Chunk with modest overlap:** split long documents into smaller pieces with just enough overlap to maintain coherence without duplicating most tokens.

In MCP-based stacks, much bloat comes from tool metadata. Use a tool broker pattern to

```
find_tool
```

then

```
call_tool
```

, significantly reducing metadata tokens in flight.

PRINCIPLE 4 — TIER YOUR MEMORY ARCHITECTURE

Choose the lightest viable memory pattern:

- **Tier 1:** Direct DB/API (Very low energy)
- **Tier 2:** Simple Indexing / Keyword Search
- **Tier 3:** Semantic Search (Medium)
- **Tier 4:** RAG (Medium-High)
- **Tier 5:** Full Context / Infinite Scroll (Very High)

PRINCIPLE 5 — MAINTAIN GREEN SOFTWARE HYGIENE

LLM efficiency multiplies when it sits on top of disciplined, low-waste software engineering.

- **Energy-proportional compute:** use auto-scaling and aggressive idle turn-down so that unused capacity does not burn power.
- **Effective caching:** design multi-level caches for prompts, results and embeddings to avoid recomputing identical work.

- **Data minimalism:** strip redundant boilerplate and whitespace from prompts and retrieved context; send only the fields the model truly needs.
- **Algorithmic efficiency:** regularly profile and optimise non-LLM code paths; gains there reduce total system energy for every user interaction.

PRINCIPLE 7 — MEASURE, BUDGET AND REVIEW

Token counts and energy per 1,000 tokens (e-tokens) are first-class metrics. Every LLM feature must have a defined energy budget.

- **Instrument everything:** capture input and output tokens, e-tokens and latency for each major feature and pathway, not just aggregate usage.
- **Assign budgets:** allocate explicit e-token and emissions budgets per application or feature tier, aligned with organisational climate goals.
- **Review cadence:** compare actual vs budgeted emissions on a regular schedule and treat large deviations as prompts for architectural change, not just cost tuning.
- **Close the loop:** feed monitoring insights back into model selection, context limits, memory tiering and prompt design.

7. Conclusion: TokenOps as Environmental Practice

Large-scale language model inference is no longer a niche technical concern. It is becoming one of the defining loads on digital infrastructure, with real consequences for energy systems, emissions and environmental justice. Treating tokens, prompts, tools and agents as designable infrastructure – rather than magical black boxes – is the only credible path to sustainable AI.

TokenOps reframes everyday engineering decisions as levers on the Non-Equivalence Problem. By measuring energy per N tokens, setting budgets, rightsizing models,

constraining context and governing agentic architectures, organisations can deliver powerful AI capabilities while keeping their environmental footprint in view.

This report and its toolkit describe an emerging discipline: token optimisation as the next environmental frontier for AI. The work now is to turn these patterns into practice, across products, teams and sectors.