Evaluating the Inclusion of LLM Training Emissions in SCI for API Consumers

Discussion Draft for Review and Consensus

This document is intended to drive discussions and develop shared understanding across GSF members. It is not a final approved position.

1. Introduction

Large Language Models (LLMs) like GPT-4, Gemini, and Claude are widely consumed via APIs. Users interact with these systems by sending prompts and receiving outputs—without any visibility or control over how the model was trained, what infrastructure was used, or the carbon impact of the development lifecycle.

This raises a critical question within the context of the Software Carbon Intensity (SCI) standard:

Should the emissions generated during the training phase of an LLM be partially allocated to end-users who only access the model via API-based inference?

This document explores both sides of the argument: the case for excluding training emissions in standard SCI measurement for API consumers, and the case for including them via amortization. It presents conceptual approaches, identifies challenges, and ends with a recommendation framework for practical implementation.

2. Scenario: LLM API Consumer Use Case

A common usage pattern for large language models (LLMs) involves consuming them via an API offered by a third-party provider. These models—such as GPT-4 (OpenAI), Gemini (Google), or Claude (Anthropic)—are pre-trained and exposed as inference endpoints. Users interact with the model by sending prompts and receiving generated responses, typically through HTTP-based API calls. They do not participate in the training process, nor do they manage the underlying infrastructure or model lifecycle.

Example:

An enterprise builds a customer service chatbot using the ChatGPT API or Vertex AI's Managed API. In this case:

- The enterprise designs and sends prompts for predefined tasks.
- It connects to the LLM via secured API calls.
- Billing is based on the number of tokens processed or the number of API calls made.

- The enterprise does not control, initiate, or have visibility into the training, hardware provisioning, or infrastructure decisions made by the provider.

Use Case 1: LLM API Consumers (Prompt-based Usage)

The model is consumed purely for inference purposes, with the user engaging only in runtime interactions, similar to querying a managed database or accessing a search engine via API.

In **analogous** use cases, such as with **managed database services**, the SCI (Software Carbon Intensity) calculation for the consuming application typically includes only the **emissions** associated with **runtime** usage, such as executing queries. The development or training emissions of the database system itself are not included in the consumer's SCI boundary.

2.1 Key Question for Consideration

This leads to a key question in the context of SCI standardization:

Should the emissions generated during the training of an LLM be partially allocated to API-based consumers who only use the model during inference and have no control over its development lifecycle?

This question introduces a boundary consideration for SCI:

- Whether to treat LLMs as managed services, where only inference emissions are included, or
- Whether to include a portion of upstream training emissions, distributed across usage, within the SCI calculation for downstream consumers.

This document presents both approaches, outlines associated challenges, and provides recommendations intended to inform consensus-building within the working group.

3. Challenges in Including Training Emissions for LLM API Consumers

In the context of LLMs consumed via APIs, including training emissions in the SCI score introduces a number of practical and methodological considerations. These arise particularly in cases where users do not own, operate, or manage the training infrastructure or model development process.

This section presents these challenges to support discussion and further exploration of possible inclusion methods.

3.1 Lack of Visibility into Training Processes

API consumers typically do not have access to key information required to estimate training emissions, such as:

- Total energy consumed during training
- Number and type of hardware resources used
- Training duration and number of iterations
- Energy source (renewable vs. fossil fuel-based)
- Efficiency strategies (e.g., model pruning, distributed training optimizations)

Without access to this data, any attempt to allocate training emissions to downstream users would rely on assumptions or external estimates, which may vary across providers and models. Without this visibility, consumers are unable to calculate or verify their share of training-related emissions. This creates uncertainty around the accuracy and consistency of any per-user allocation.

3.2 Multi-Tenancy and Shared Model Usage

In SCI, the amortization of hardware emissions is supported by a defined methodology: the embodied emissions of a physical device (e.g., GPU) are allocated over its useful life and actual compute utilization. This works because:

- The hardware is often owned or dedicated to a specific workload.
- The operator can track its lifespan, usage hours, and workload attribution.

However, applying this same principle to LLM training in a multi-tenant, API-accessible environment presents fundamental challenges.

Pre-trained LLMs are typically:

- Hosted on shared infrastructure (e.g., large cloud clusters)
- Accessed by thousands or millions of downstream users
- Dynamically routed across models and environments based on provider-level optimization

Attempting to amortize training emissions across this setup raises several unresolved questions:

3.2.1 Key Questions to Address

- **What is the total expected usage volume of the model (`N_total`)?**
  - Is this measured in prompts, tokens, API calls, or another metric?
  - Is this an estimate made at training time, or updated dynamically?
- **What is the lifespan of a given LLM model version?**

    - Is the intended lifecycle defined by time (e.g., 1 year), usage volume, or business decision?
    - How is amortization handled if the model is deprecated earlier than planned?
- **How are emissions allocated fairly across different users?**
    - Should usage be weighted equally across all prompt types and token lengths?
    - Do heavier users receive proportionally higher emissions allocations?
- **How is shared infrastructure accounted for?**
    - When the same hardware is used to train multiple models or perform unrelated tasks, how are emissions divided?
    - Can emissions from shared compute be isolated per model or training job?
- **What happens when a model is fine-tuned or re-trained?**
    - Are emissions from incremental updates folded into the original amortization schedule?
    - Is a new baseline set for each major update?

In the absence of clear answers to these questions, applying hardware-style amortization to LLM training across a multi-tenant architecture remains complex. Unlike physical devices with known depreciation schedules and usage tracking, LLMs hosted in shared environments operate with abstracted infrastructure and opaque access patterns.

Any effort to include training emissions for API users would require standardized assumptions or disclosures from model providers, which are not currently available in practice.

3.3 Versioning and Continual Updates

Large Language Models (LLMs) are not static assets. Once deployed, they typically undergo various forms of updates throughout their operational life. These updates include improvements to architecture, retraining on additional data, fine-tuning for safety or domain performance, and optimizations for latency and cost. In some cases, entirely new versions of the model may be released under the same service name (e.g., GPT-4 vs. GPT-4 Turbo).

This evolving nature introduces several complexities when considering how to amortize training emissions for inference use cases.

3.3.1 Key Considerations

- **What defines the version boundary?**
  Is every updated version (e.g., fine-tuned or optimized model) considered a new model from an emissions accounting standpoint? Or is it part of the same training lifecycle?
- **How is retraining accounted for?**
  If a model is retrained or fine-tuned periodically, should each iteration restart the amortization cycle, or be added incrementally to the original training emissions?
- **How do version changes affect allocation?**
  A user calling an API may unknowingly be routed to different model versions over time. How can emissions be accurately attributed if the underlying model varies between requests?
- **What if models are short-lived?**
  In some cases, a model may be retired earlier than anticipated due to the release of a

more efficient version. If the original model served fewer prompts than projected, how should unallocated training emissions be treated?
- **Should model upgrades retroactively impact user SCI?**
  If a provider replaces the backend model with a more efficient version, does the amortized training allocation for past calls remain valid, or should it be adjusted retroactively?

These questions highlight a key distinction between traditional hardware amortization and model-level amortization:

- **Hardware amortization** is typically linear and based on predictable wear/use cycles.
- **LLM training amortization**, especially in rapidly evolving environments, may be **non-linear, dynamic, and subject to version fragmentation**.

Addressing these uncertainties would require transparent version tracking, usage metrics, and emission disclosures from model providers. Without these, consistent and verifiable amortization of training emissions remains difficult to implement at the API consumer level.

3.4 SCI Boundary Alignment

The **Software Carbon Intensity (SCI)** specification defines emissions within the context of a system boundary. This boundary is based on what the application developer, organization, or user can **control**, **measure**, or **influence**. It ensures that SCI values are actionable and reflect operational responsibility.

In the case of LLM API consumers, including training emissions may introduce potential inconsistencies with these boundary principles.

3.4.1 Key Considerations

- **Control and Influence**
  LLM API users typically do not control the training infrastructure, model architecture, energy sources, or training iterations.
  Including upstream training emissions in their SCI score may assign responsibility for activities that are outside their operational scope.
- **Precedent with Managed Services**
  In current SCI applications, emissions from the development and maintenance of **managed cloud services** (e.g., databases, storage APIs, compute runtimes) are not typically included in the SCI calculation for consumers of those services.
  Only **runtime usage emissions**—such as executing a query or storing data—are considered within the consumer's boundary.
- **Consistency Across Use Cases**
  Including training emissions for LLM usage, while excluding development emissions for other managed services, may introduce inconsistencies in how SCI is applied across different software systems and deployment models.
  This could affect comparability between applications and challenge the interpretation of SCI scores.
- **Implications for Measurability and Actionability**
  The SCI framework emphasizes the importance of using **measurable and verifiable data**.

Without direct access to emissions data from training or a standardized disclosure mechanism, most API consumers would not be able to accurately report or validate training-related emissions.
This may reduce the trustworthiness and utility of the SCI value for both internal and external reporting.

Aligning the SCI boundary to include only emissions resulting from user-controlled inference activities may help maintain consistency with existing practices and ensure SCI remains practical for broader implementation. Any expansion to include upstream emissions—such as model training—would require a clearly defined boundary extension, transparent data availability, and appropriate documentation.

3.5 Lack of Standardization in Allocation Methods

If training emissions are to be included in the SCI calculation for LLM API consumers, a consistent and standardized method would be required to allocate those emissions fairly and transparently across different users. At present, such a methodology does not exist. This creates a number of unresolved challenges that affect the feasibility and comparability of SCI calculations.

3.5.1 Unclear Denominator for Allocation

For training emissions to be amortized across users, a denominator is needed to determine what portion of the total emissions each user should be responsible for. This denominator represents the **total expected usage of the model over its lifecycle**, but there is currently no consensus on what metric should be used.

Several possible options exist, each with limitations:

1. **Number of Prompts**
    - Simple to count and user-accessible.
    - However, prompt length and complexity can vary significantly. A brief prompt and a complex multi-sentence query would be treated equally despite requiring different amounts of compute.
2. **Number of Tokens**
    - More closely aligned with actual compute load, especially when considering both input and output tokens.
    - Already used for billing by several providers (e.g., OpenAI, Google Cloud).
    - However, tokenization schemes differ across providers, and not all providers offer transparent token-level usage data.
3. **Inference Time or Compute Time**
    - The most accurate reflection of energy consumption and carbon emissions.
    - Yet this information is generally not available to API consumers, as it is managed internally by the service provider.
4. **API Calls or Sessions**
    - Captures usage at a higher level of abstraction.
    - May be useful at the application level, but not granular enough for accurate emissions modeling. Sessions can vary in length and computational cost.

Without a common denominator, different users or providers may choose different methods of allocation, leading to inconsistencies in how SCI scores are calculated and interpreted.

In addition to the denominator issue, several other challenges complicate allocation:

3.5.3 Estimating Total Usage (N_total)

Accurate amortization requires an estimate of the total number of inferences, tokens, or sessions the model is expected to serve during its operational life. This is not typically disclosed and may change over time based on provider-side decisions.

3.5.4 Inconsistent Provider Disclosures

There is currently no standard for model providers to disclose training emissions (`E_train`) or usage volume (`N_total`). Without these, SCI estimates that include training emissions cannot be independently verified.

3.5.5 Scope of Responsibility Across Multiple APIs

Applications may integrate with multiple LLMs across providers. If each provider uses a different allocation method, aggregating these values introduces complexity and potential double-counting or misalignment.

3.5.6 Version-Specific Allocation

Training emissions and performance characteristics may vary across model versions (e.g., GPT-4 vs. GPT-4 Turbo). Without transparent versioning and traceability, users cannot align their usage with the correct emissions profile.

In the absence of standardized allocation methods and reliable provider disclosures, including training emissions in SCI calculations for LLM API consumers remains difficult to apply consistently. Further methodological development and stakeholder collaboration would be needed before this could become a viable option in standard SCI implementations.

4. Inclusion of Training Emissions Through Amortization

Previous sections outlined the approach where training emissions are excluded from the SCI calculation for LLM API consumers, based on boundary control and current feasibility.

This section presents an alternative approach**:** to include training emissions by amortizing the total emissions generated during model training over the expected lifetime usage of the model. The purpose is to explore how this inclusion could be methodologically structured and what types of assumptions, data inputs, and disclosures would be required to enable its application in practice.

Both the proposed approaches is intended for discussion and does not reflect a finalized standard.

4.1 Conceptual Approach to Amortization

To include training emissions in the SCI calculation for LLM API consumption, one proposed method is to amortize the total training emissions across the expected usage of the model. This allows a portion of the training emissions to be allocated to each **consuming entity** (e.g., user, application, or system) based on their proportional usage.

The allocation can be calculated as follows:

Let:

- `E_train` = Total emissions ($kgCO_2e$) associated with training the model
- `N_total` = Estimated total number of units of usage the model will serve over its operational life (e.g., prompts, tokens)
- `N_entity` = Number of units of usage attributable to the specific consuming entity (user or application)

Then the entity's share of training emissions:

`E_train_entity = E_train × (N_entity / N_total)`

The full SCI score for the consuming entity can be calculated by adding inference-related emissions:

`SCI_total_entity = E_infer_entity + E_train_entity`

This mirrors the SCI practice used for hardware amortization, where the embodied emissions of devices are distributed over their expected operational lifespan and usage profile. In the context of LLMs, this approach would require standardization of usage units, version tracking, and provider disclosures to ensure consistency and transparency.

4.2 Assumptions and Required Inputs

For the amortization-based inclusion of training emissions to be feasible and consistent, several key inputs, disclosures, and assumptions would be required from model providers and consuming entities:

- **Training Emissions Disclosure (`E_train`)**
  Model providers would need to disclose the total carbon emissions associated with the model's training process. This includes energy consumption, carbon intensity of the power source, hardware configuration, and supporting infrastructure such as cooling systems. Standardized methodologies for estimating and reporting `E_train` would help ensure comparability.
- **Total Usage Estimate (`N_total`)**
  Providers would need to estimate the total number of usage units (e.g., tokens, prompts) that the model is expected to serve during its operational lifetime. This forms the denominator for amortization and would ideally be revised over time based on actual observed usage.
- **Usage Attribution to Consuming Entities (`N_entity`)**
  Each consuming entity (user, application, or system) would need a way to track or be informed of its own usage of the model over time. This could be based on tokens processed, number of API calls, or another standardized usage metric.
- **Unit of Allocation**
  A consistent metric—such as token count, prompt count, or another agreed unit—would need to be defined for allocating emissions. This would ensure that all consuming entities are evaluated using the same basis for comparison and fairness.

- **Model Version Transparency and Routing Information**
  As models evolve, consuming entities may interact with different versions (e.g., GPT-4 vs. GPT-4o). To ensure alignment between usage and the corresponding training emissions, providers would need to offer version traceability and routing metadata.
- **Aggregation and Reporting Across Multiple Providers**
  In cases where a consuming entity uses multiple LLM APIs, consistent aggregation rules would be required to avoid double-counting or underrepresentation of training emissions. These rules would need to be defined to support multi-provider and multi-model usage scenarios (like in agentic scenario)

4.3 Opportunities for Standardization

While the challenges of including training emissions are significant, they also present several opportunities for advancing SCI practices in AI:

- **Methodological Development**
  There is an opportunity to define a robust, shared methodology for amortizing emissions from model training in a way that is consistent, transparent, and aligned with SCI principles.
- **Disclosure Frameworks**
  Establishing common formats and practices for disclosing model training emissions (`E_train`) and expected usage volumes (`N_total`) could create a foundation for comparable SCI reporting across providers.
- **Incentives for Efficiency**
  If training emissions are partially attributed to consumers, providers may be incentivized to train models more efficiently, use cleaner energy sources, or disclose sustainability metrics as a competitive differentiator.
- **Improved Version Traceability**
  Requirements for tracking which model version was used for a given interaction could improve not only emissions accounting but also transparency, reproducibility, and compliance in broader AI governance contexts.

These areas represent opportunities for collaboration between model providers, SCI working groups, and consuming entities to evolve the SCI framework in a direction that accommodates AI-specific complexities while maintaining practical applicability.

5. Recommendation

The preceding sections outlined two distinct approaches to handling training emissions in the SCI calculation for LLM API consumption:

1. **Exclusion of Training Emissions**
   – Based on the principle of system boundary control
   – Focuses solely on emissions from user-driven inference
   – Aligned with how SCI currently treats managed services
   – Emphasizes actionability, measurability, and consistency
2. **Inclusion of Training Emissions via Amortization**
   – Seeks to account for upstream lifecycle emissions
   – Requires model providers to disclose training emissions and expected usage

– Relies on standardization of allocation units, usage attribution, and version traceability
– Supports lifecycle awareness and broader sustainability objectives

5.1 Recommendation Framework

To maintain both practical applicability **and** methodological flexibility, the following dual-path approach is proposed for discussion:

Default SCI Boundary (Inference-Only Approach)

- For typical LLM API consumers, SCI calculations should include only emissions associated with inference.
- This reflects what the consuming entity can control or influence directly.
- Maintains consistency with current SCI methodology for managed services.

Extended SCI Boundary (Optional Inclusion of Training)

- An extended boundary may optionally include amortized training emissions when:
  - The model provider discloses total training emissions (`E_train`) and total expected usage (`N_total`)
  - A consistent allocation method is applied across consuming entities
  - The usage can be attributed to a specific model version with traceability
- Extended boundary calculations should be clearly labeled to avoid confusion with default SCI values.

Summary Table

Aspect	Default SCI (Inference-Only)	Extended SCI (Includes Training)
Training emissions included	No	Yes (if provider data available)
Inference emissions included	Yes	Yes
Provider disclosures required	No	Yes
Standardized usage unit required	Optional	Required
Traceability of model version	Not required	Required
Consistency with SCI boundaries	Aligned	Requires explicit extension
Appropriate for	Most API consumers	Advanced reporting scenarios

Conclusion

Both approaches have merit depending on the context of use and the maturity of the ecosystem. Maintaining the default SCI boundary enables broad adoption and practical

measurement, while offering an extended boundary provides a path for more comprehensive accounting as transparency and standards evolve.

This recommendation is intended to support further discussion and consensus within the SCI and AI workgroup. It does not represent a finalized standard but a structured direction for refinement and alignment.

Revision History

Version	Date	Author	Description
v0.1	3 Apr 2025	Navveen Balani	Initial draft for discussion