SPROUT: Green Generative AI with Carbon-Efficient LLM Inference

Anonymous submission

Abstract

The rapid advancement of generative AI raises significant environmental concerns, notably carbon emissions. Our framework, SPROUT, demonstrates the carbon footprint reduction of generative LLM inference by over 40% in real-world evaluations, marking a critical effort to align NLP with eco-friendly initiatives.

1 Introduction

The AI boom, driven by the demand for generative artificial intelligence (GenAI) (Nijkamp et al., 2023; Jumper et al., 2021; Pierce and Goutos, 2023; Chen et al., 2023a), has prompted concerns over its environmental impact, particularly in terms of carbon emissions associated with the datacenters hosting these technologies. OpenAI's reported pursuit of trillions in investment for AI chips (Fortune, 2024), destined for their datacenter infrastructure, underscores the scale of resource expansion required to support GenAI's growth.

Generative large language models (LLMs) have gained a substantial user base across various scientific fields (Singhal et al., 2023; Lin et al., 2023; Liu et al., 2024, 2023b; Christofidellis et al., 2023). This underscores a critical need for research focused on minimizing LLMs' environmental impact. Although training these models requires extensive compute cycles and carbon footprint, it is the inference processes of these LLMs that are poised to become the predominant source of emissions, according to various prior studies (Chien et al., 2023; Wu et al., 2022; de Vries, 2023). Unlike traditional natural language understanding models that predict a single masked word or sentiment, generative LLMs are even more carbon-demanding as they perform iterative predictions for each request until reaching a predefined token or iteration limit. Despite the urgency of this issue, there lacks a solution for reducing carbon emissions specifically from the LLM inference operations – which is natural given the field is in early stages, but rapidly evolving.

In this paper, we design SPROUT as the first work to address the sustainability challenges in running a generative LLM inference service. Various previous works have attempted to reduce the carbon footprint of machine learning (ML) applications (Wu et al., 2022; Acun et al., 2023a; Li et al., 2023a), but none has designed optimizations tailored to LLM inference which is becoming a dominant workload and requires intervention to reduce its carbon footprint. The following summarizes SPROUT's insights and contributions.

Introduction of generation directives to LLM inference for carbon saving. Previous works have identified the opportunity to manipulate the number of parameters in the model to save energy and cost (Wan et al., 2020; Romero et al., 2021), while SPROUT is the first work to identify that in generative language model inference, its autoregressive generation pattern presents a unique opportunity beyond previous works. SPROUT introduces the concept of "generation directives", a strategy to indirectly manipulate the number of autoregressive inference iterations while providing high-quality content generation. For example, a directive can guide the model to provide a concise response, saving significant carbon from generating a long sequence while still being accurate. Identifying the variability in the carbon intensity of the electricity generation and the diverse requirements of different tasks, SPROUT can leverage different generation directives to minimize the carbon footprint of LLM inference with a guarantee of generation quality.

Design, implementation and evaluation of carbon-friendly generation directive configuration for LLM inference. We present SPROUT, a novel carbon-aware generative language model inference framework designed to reduce carbon footprint through the strategic use of token generation directives while maintaining high-quality outputs. From the selection of directive levels based on electricity grid carbon intensity and user behavior variability, SPROUT introduces a linear programming approach for system-level optimization, balancing carbon savings with generation quality. SPROUT identifies the difficulty in retrieving generation quality feedback, and implements an automatic offline and opportunistic quality assessment mechanism to ensure the framework's decisions are informed by up-to-date generation quality.

We evaluate SPROUT using production software setup, state-of-the-art LLM, representative corpus to synthesize user prompts, and real carbon intensity traces from global electricity grid operator regions. Our evaluation confirms SPROUT's effectiveness in reducing carbon emissions by more than 40% while still achieving high generation quality.

2 Background and Motivation

Carbon footprint of an inference request. The carbon footprint is a metric for quantifying the amount of greenhouse gas emissions (gCO_2) generated. When requesting a service from a datacenter server (e.g., HTTP requests), its carbon footprint comprises the **operational carbon** and **embodied carbon**. The operational carbon comes from electricity generation to power the datacenter, which powers the hardware (e.g., GPUs) that serves the request (carbon intensity $\times$ energy). The carbon intensity (denoted as $CO_2^{\text{Intensity}}$) of the grid, representing the amount of CO_2 emission per unit of energy usage (gCO_2/kWh), reflects the "greenness" of the energy source. For example, wind turbines have lower carbon intensity than coal power plants. Due to the temporal difference in availability of renewable energy, *carbon intensity varies over time.*

Embodied carbon (denoted as CO_2^{Embed}) represents the carbon emissions associated with the manufacturing and packaging of computer components, effectively "embodied" within the device itself. We follow the methodology in (Gupta et al., 2021, 2022) to model the embodied carbon. For an inference request processed by a computing device, its share of embodied carbon is proportional to the execution time relative to the device's overall lifespan. The total carbon footprint of serving an inference request, C_{req}, can be formally expressed as:

$$C_{\text{req}} = CO_2^{\text{Intensity}} \cdot E_{\text{req}} + \frac{CO_2^{\text{Embed}}}{T_{\text{life}}} \cdot T_{\text{req}} \quad (1)$$

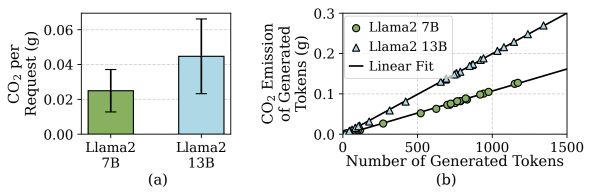


Figure 1: Quantifying the impact of factors on an inference request's carbon footprint: (a) the number of model parameters and (b) the number of generated tokens.

Here, E_{req} and T_{req} represent the energy consumption and execution time for the request, respectively, with T_{life} indicating the assumed device lifespan, set to five years for this analysis. Given that the lifespan of the device significantly exceeds any single request's execution time, operational carbon dictates the total carbon footprint, except in scenarios where $CO_2^{\text{Intensity}}$ approaches zero.

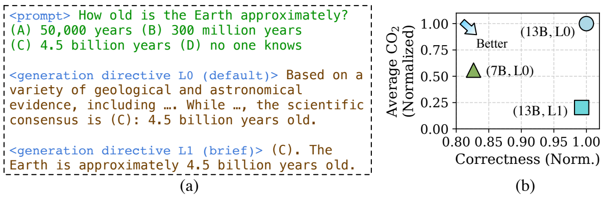
Motivational Empirical Study and Opportunities. We make three major observations.

Takeaway 1. The LLM inference carbon footprint depends on not only the model size but also the number of tokens generated, presenting a new opportunity to reduce carbon without being forced to choose a smaller model size.

In Fig. 1 (a), we demonstrate how the carbon footprint of LLM inference changes with model size, showcasing examples with the Llama2 model at 7 billion (smaller model) and 13 billion parameters (larger model). In Fig. 1 (b), we execute a series of input prompts on the Llama2 7B and 13B model and observe that there is a strong linear correlation between the total carbon emission and the volume of tokens generated from request.

The autoregressive token generation iteratively predicts the subsequent token until an end-of-sequence (EOS) token emerges or a predefined limit is reached. Despite initial computations to pre-fill the KV cache with key and value vectors from the input prompt, we show that *the overall carbon emission of a request is largely dictated by the quantity of generated tokens*. Our experimental results show that rather than naively relying on smaller models and potentially compromising the contextual understanding capabilities, we can potentially infer from a larger size model but focus on generating fewer tokens (Fig. 1 (b)).

Takeaway 2. Incorporating generation directives into prompts can significantly reduce the carbon footprint by enabling concise yet accurate responses. To control the LLM token generation length, we introduce "generation directive".


```
<prompt> How old is the Earth approximately?
(A) 50,000 years (B) 300 million years
(C) 4.5 billion years (D) no one knows

<generation directive L0 (default)> Based on a
variety of geological and astronomical
evidence, including …. While …, the scientific
consensus is (C): 4.5 billion years old.

<generation directive L1 (brief)> (C). The
Earth is approximately 4.5 billion years old.
```

Figure 2: (a) Example of applying generation directive. (b) Hosting larger models (e.g., Llama2 13B) with generation directives can outperform smaller models (e.g., Llama2 7B) in both carbon emission and correctness.

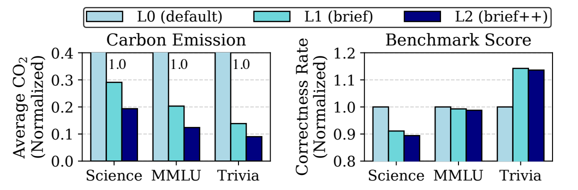


Figure 3: Applying generation directives across different tasks reveals varied sensitivity to these directives.

Definition 1 *A **generation directive** is an instruction (e.g., "respond concisely") that guides the model to generate tokens. Each **generation directive level** specifies a pre-defined text sequence that acts as this guiding instruction.*

In Fig. 2 (a), we show a prompt from the MMLU task (Hendrycks et al., 2020). Without using any specific directives (level L0), the Llama2 13B model defaults to generating an extensive number of tokens. However, such detailed background information may not always align with user preferences. Applying a generation directive (level L1) ensures both brevity and correctness. This practice demonstrates the potential to reduce carbon emissions from token generation. Fig. 2 (b) demonstrates such potential quantitatively by measuring the CO_2 emission and MMLU correctness rate. It shows that employing generation directives with a larger model (13B, L1) significantly outperforms smaller models (7B, L0) in both carbon and the accuracy of generated content. This is attributed to the larger model's superior contextual understanding, which, when combined with concise generation directives, retains its comprehensive knowledge base without unnecessary verbosity, highlighting the advantage of optimizing response generation instead of model sizes.

Takeaway 3: The impact of employing generation directives on carbon emissions and accuracy differs across user tasks, presenting an interesting challenge in optimally utilizing these directives, particularly in the context of fluctuating carbon intensity. In Fig. 3, we show the impacts of different generation directives (L0, L1, L2) on different tasks including science knowledge (Lu et al., 2022) and trivia knowledge (Joshi et al., 2017). We observe that both the amount of carbon emission and the generation's correctness rate vary with the task. Responding to these challenges, we design SPROUT, a generative LLM inference framework that takes advantage of generation directives to dynamically optimize the carbon footprint while guaranteeing high-quality generations.

3 SPROUT Design

3.1 System Overview and Key Ideas

SPROUT is designed as the first carbon-aware generative language model inference framework, utilizing token generation directives to minimize carbon footprint while ensuring high-quality content generation. Fig. 4 shows a brief design overview of SPROUT. Once the user prompts are assigned to an inference server by the load balancer, they are tokenized into numerical representations. In this phase, a generation directive selector ❶ assigns a directive level to each prompt, integrating it into the tokenized input. The policy for assigning directive levels is established by SPROUT's token generation directive optimizer ❷, as detailed in Sec. 3.2. This optimizer systematically considers the carbon intensity of the local grid and the feedback on both the quality and carbon footprint of token generation.

To retrieve the local carbon intensity, we access third-party API endpoints such as Electricity Maps (Maps, 2024). To enable inference carbon feedback, SPROUT monitors the datacenter PUE and device energy with tools such as `nvidia-smi` to record the GPU power and processing time of requests and save the logs to the database. However, obtaining the token generation quality feedback is a different process from the above metrics. After autoregressive inference concludes on the inference server ❸, the generated tokens are detokenized and sent back to the user clients, while simultaneously, the request and node monitoring logs are archived in the database. A generation quality evaluator ❹ then extracts a sample of prompts from the database, generates responses for each at all generation directive levels, and identifies the directive level that yields the best response for each request.

However, determining the directive level that yields the best response presents a challenge due to the subjective nature of preference and the absence of a definitive best response. Since manual

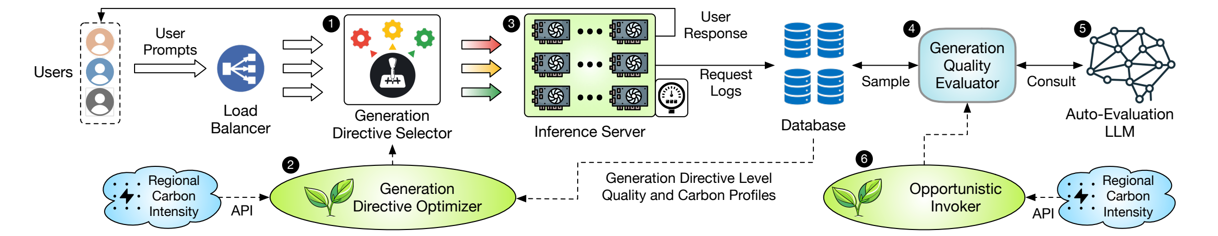

Users
User Prompts
Load Balancer
1
Generation Directive Selector
3
Inference Server
User Response
Request Logs
Database
Sample
4
Generation Quality Evaluator
Consult
5
Auto-Evaluation LLM
Regional Carbon Intensity
API
2
Generation Directive Optimizer
Generation Directive Level Quality and Carbon Profiles
6
Opportunistic Invoker
API
Regional Carbon Intensity

Figure 4: Overview of SPROUT's Carbon-Friendly Inference System.

evaluation by humans is impractical, following a methodology from recent research (Dubois et al., 2024), SPROUT employs an LLM-based automatic evaluator, rather than human evaluators, to provide generation quality feedback, aligning with common academic and industry practices (Liu et al., 2023a; Bai et al., 2024; MistralAI, 2024).

SPROUT's evaluator consults an auto-evaluation LLM ❺ to gauge its preference for the responses, logging them back into the database. The whole process happens offline, and since the evaluation process also incurs carbon emission, SPROUT's opportunistic evaluation invoker ❻ (Sec. 3.3) ensures the evaluations are carried out only as necessary and during low carbon intensity periods.

3.2 Generation Directive Optimizer

While employing generation directives to reduce token output in the autoregressive process is beneficial for lowering carbon emissions, it poses a risk to content quality. Two key external factors further complicate this balance: the regional carbon intensity powering the datacenter, which directly affects the efficacy of carbon savings, and the nature of user prompts, which influences the impact of generation directives on both emissions and content quality. To address these challenges, SPROUT's optimizer is designed to dynamically adjust to fluctuations in carbon intensity and the variability of user prompt tasks. In scenarios of low carbon intensity, SPROUT prioritizes directives that enhance content quality, leveraging the carbon discount in generating new tokens. Under high carbon intensity, it opts for directives that may slightly compromise quality but significantly reduce emissions. This strategy underpins the mathematical formulation of the SPROUT optimizer, ensuring that it targets both carbon footprint and content quality.

Optimization variable. Optimizing directive levels for each request introduces several practical complications: (i) Dimensionality challenge: the number of dimensions equals the number of received requests (user prompts) at each optimization step. (ii) Computational overhead: the optimization is in the critical path before the autoregressive inference starts, delaying the time to first token (TTFT). (iii) Predictability issues: anticipating the impact of each directive level on carbon emissions and content quality for individual requests is challenging. We can only infer general trends from historical data, which do not apply to specific future prompts.

Considering these challenges, SPROUT adopts a system-level optimization strategy for generation directive levels, rather than an impractical per-request optimization. It achieves this by determining the probability of selecting each directive level for all user requests (prompts). Let n denote the number of available generation directive levels. The optimization variable, represented as $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]^T$, defines $x_i \in [0, 1]$ as the probability of applying the i-th directive level to any request, with x_0 representing the baseline directive L0 (indicating no directive). To ensure every request receives a directive level, the condition $\sum_{i=0}^{n-1} x_i = 1$ must be satisfied. This system-wide probabilistic approach to directive selection, while not optimizing for individual prompts, is shown to achieve carbon savings close to those of an impractical per-request Oracle optimizer (Sec. 5).

Objective function. Following Eq.1, we design the objective function $f(\mathbf{x})$ to encapsulate the expected carbon footprint of an inference request.

$$f(\mathbf{x}) = k_0 \cdot \mathbf{e}^T \mathbf{x} + k_1 \cdot \mathbf{p}^T \mathbf{x} \tag{2}$$

where $\mathbf{x}$ denotes the probabilities of selecting each directive level across all user prompts. It incorporates (i) the current regional carbon intensity (k_0 in gCO_2/kWh), obtained via API; (ii) the prorated per-second embodied carbon of the inference hardware through its device lifetime (k_1 in gCO_2/s); and (iii) the profiles of energy consumption ($\mathbf{e}$) and processing time ($\mathbf{p}$) for requests employing various generation directive levels. The vectors $\mathbf{e} = [e_0, e_1, \ldots, e_{n-1}]^T$ and $\mathbf{p} = [p_0, p_1, \ldots, p_{n-1}]^T$ represent the average energy (in kWh) and processing time (in seconds), respectively, for recent requests guided by each directive level.

Generation quality constraints. The optimizer also requires feedback from the generation quality evaluator, which reports the auto-evaluation LLM's preference on which directive level is the best for all sampled requests. Let $\mathbf{q} = [q_0, q_1, \ldots, q_{n-1}]^T$ where $q_i \in [0, 1]$ denote the preference rate of each directive level reported by the evaluator. For example, if $\mathbf{q} = [0.5, 0.3, 0.2]^T$, it means 50% of the time, the auto-evaluator prefers the response generated using directive L0, 30% of the time by L1 and 20% of the time by L2. We can denote the expected generation quality as $\mathbf{q}^T\mathbf{x}$. During the optimization, we need to make sure the preference rate does not deviate beyond a threshold of $\xi \in [0, 1]$ away from the q_0 generation baseline using directive $L0$. In addition, SPROUT designs the actual quality deviation from q_0 to vary based on the current carbon intensity – when the carbon intensity is low, the constraint should be more strictly enforced (deviation closer to 0) since renewable energy is abundant in the grid to support high-quality generation, and vice versa, during high carbon intensity periods, the deviation should be closer to ξ. This can be formulated as an inequality constraint:

$$\mathbf{q}^T\mathbf{x} \geq \left(1 - \frac{k_0 - k_0^{\min}}{k_0^{\max} - k_0^{\min}} \cdot \xi\right) \cdot q_0 \tag{3}$$

where $k_0^{\min}$ and $k_0^{\max}$ are the known historical minimum and maximum carbon intensities, respectively, and are used for min-max normalization of k_0. The parameter ξ, adjustable according to system requirements, facilitates a balance between carbon footprint and content quality. For SPROUT's evaluation (detailed in Sec. 5), we set ξ to 0.1.

Problem formulation. The overall optimization is

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \tag{4}$$

$$\text{s.t.} \quad \mathbf{q}^T\mathbf{x} \geq q_{lb}, \tag{5}$$

$$\forall i,\ 0 \leq x_i \leq 1, \tag{6}$$

$$\sum_{i=0}^{n-1} x_i = 1 \tag{7}$$

For simplicity, we replace the right-hand side of Eq. 3 with scalar q_{lb} to represent the quality lower bound. Eq. 6 indicates that the probability of each level is within the range of 0 to 1, and Eq. 7 indicates that all probabilities sum to 1. Note that $f(\mathbf{x})$ is linear because both $\mathbf{e}^T$ and $\mathbf{p}^T$ are constants to the optimization variable $\boldsymbol{x}$ (Eq. 2), and all the constraints in Eq. 5, 6 and 7 are linear to $\mathbf{x}$. Therefore, we have mapped the optimal generation directive level configuration problem to a linear programming problem and we can use the HiGHS dual simplex solver (Huangfu and Hall, 2018) to find the optimal solution for $\mathbf{x}$.

3.3 Opportunistic Offline Quality Assessment

In Eq. 5, SPROUT relies on the $\mathbf{q}^T$ vector to impose the quality constraint. As a carbon-friendly generative LLM inference framework, SPROUT not only cares about the carbon footprint of the inference server but also the quality evaluation process, especially when the auto-evaluation LLM can have $> 10\times$ number of parameters than the inference model (e.g., GPT-4 compared with Llama model). Note that the quality evaluation is not in the critical path of online inference serving and thus can be done offline opportunistically in a different server.

SPROUT triggers the offline quality evaluation based on specific carbon intensity thresholds of the evaluation server. When deciding on whether to evaluate at the current time t, it is critical to weigh the carbon intensity of the evaluator LLM at the current moment, denoted as $k_2^{(t)}$, against the time elapsed since the last evaluation at t_0. Direct and frequent evaluations can lead to unnecessary carbon emissions without significant benefit, whereas delayed evaluations can undermine the optimizer's reliability, as the $\mathbf{q}^T$ vector becomes outdated (Sec. 3.2). To mitigate these issues, we first enforce a grace period to ensure the evaluation does not occur too frequently, then introduce an urgency multiplier to the carbon intensity to capture the increasing need for re-evaluation as time progresses. The urgency-adjusted carbon intensity $k_2'^{(t)}$ is expressed as

$$k_2'^{(t)} = e^{-\beta(t-t_0)} \cdot k_2^{(t)} \tag{8}$$

The urgency parameter, β, determines the rate at which the evaluation interval incurs penalties over time, ensuring that the value of immediate evaluation – offering a timely update to the $\mathbf{q}^T$ vector in Sec. 3.2 – is weighed against waiting for potentially lower future carbon intensities. By default, we set β so that the urgency-adjusted carbon intensity $k_2'^{(t)}$ becomes $1/2$ of the actual carbon intensity after 24 hours without evaluation. An offline evaluation starts under conditions of: (i) t_s represents a local minimum for $k_2'^{(t)}$, indicating a positive second-order derivative at that point; (ii)

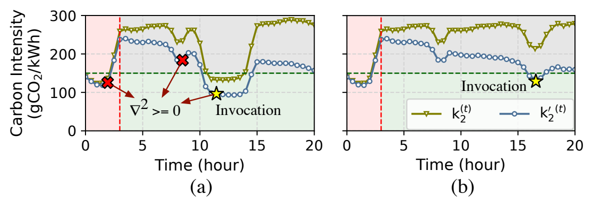


Figure 5: Process to select the opportunity to invoke quality evaluation (golden star). (a) The urgency-adjusted $k_2'^{(t)}$ must fall within the green zone after the grace period (red area) and below the carbon intensity threshold (green line). The red crosses, despite showing a positive second-order derivative, do not qualify for evaluation. (b) Even if carbon intensity stays high all the time, the increasing evaluation urgency ensures that offline evaluation always occurs.

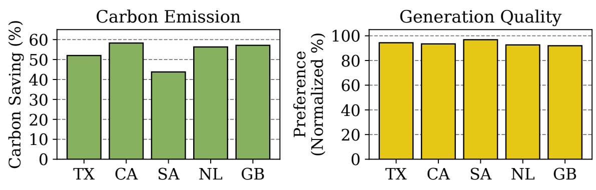


Figure 6: SPROUT significantly saves carbon while preserving quality across all geographical regions.

a grace period has elapsed since the last evaluation; (iii) the urgency-adjusted carbon intensity at t_s, $k_2'^{(t_s)}$, falls below a predefined threshold, such as 50% of the historical maximum carbon intensity. This evaluative mechanism, illustrated in Fig. 5, highlights moments of evaluation marked by stars in two different cases, underlining SPROUT's consideration for both carbon intensity and the need for timely quality feedback.

We have implemented SPROUT's generation directives as system prompts, implemented the inference server and monitoring framework following industry standards, and developed an automatic quality evaluation mechanism for Sec. 3.3. More details are provided in Appendix A.2.

4 Methodology

Experiment setup. We conduct experiments on a testbed comprising two nodes, each equipped with two NVIDIA A100 40GB GPUs and two AMD EPYC 7542 CPUs. We use Meta Llama2 13B (Touvron et al., 2023) to establish the inference server, with each GPU hosting a model instance within its 40GB HBM memory. To assess SPROUT's efficiency, three levels of generation directives are implemented: L0 as the default baseline with no directives, L1 for "brief" generation, and L2 for "very brief" generation. GPT-4, accessed via the OpenAI API, serves as the auto-evaluation LLM for offline quality assessments. Each of our quality evaluation requests to OpenAI's `gpt-4-0613` API costs about \$0.01 on average. SPROUT is evaluated using a diverse set of NLP tasks across five real-world electricity grid operation regions of US Texas (TX), US California (CA), South Australia (SA), Netherlands (NL), and Great Britain (GB) in February 2023, detailed in Appendix A.3.

Competing schemes. SPROUT is evaluated alongside five distinct strategies, detailed as follows: **BASE** is the baseline strategy that represents a vanilla LLM inference system, it does not explore the opportunity of generation directives discussed in Sec. 2. **CO_2_OPT** represents a scheme that aggressively minimizes CO_2 emissions without considering the generation quality. It will always use the generation directive level that yields the lowest carbon footprint. **MODEL_OPT** is an implementation of the idea to adjust the underlying model parameters to achieve optimization goals from previous works (Romero et al., 2021; Wan et al., 2020). Unaware of the generation directives, this scheme uses inference model variants (i.e., Llama2 7B and 13B) as optimization variables since model variants also introduce the trade-offs between carbon and generation quality. It represents the optimal model variant selection for the user prompts. **SPROUT_STA** is a static version of SPROUT, applying a single, month-long optimal generation directive configuration identified through offline configuration sweeping, without dynamic adjustments based on real-time carbon intensity and generation feedback. **ORACLE** is an impractical scheme based on oracle information. It assumes the inference carbon emission on every generation directive level is known ahead of time for all user prompts, and knows the exact generation quality feedback for future prompts instead of relying on sampling.

Metrics. The two primary metrics are the inference carbon footprint and the text generation quality. The carbon footprint metric accounts for the CO_2 emissions associated with each inference, averaged for comparison against the default operation represented by BASE. The generation quality is measured from the auto-evaluation LLM's preference, normalized against BASE as a percentage.

5 Evaluation

Effectiveness of SPROUT. SPROUT *consistently achieves substantial carbon savings while maintaining high generation quality in diverse geo-*

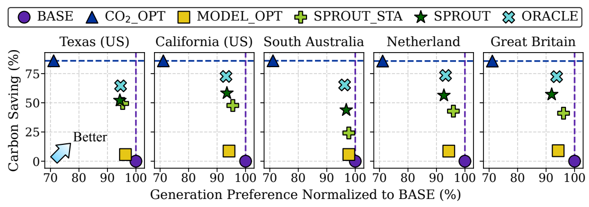


Figure 7: Sprout excels when competing against competitive strategies and is closest to Oracle

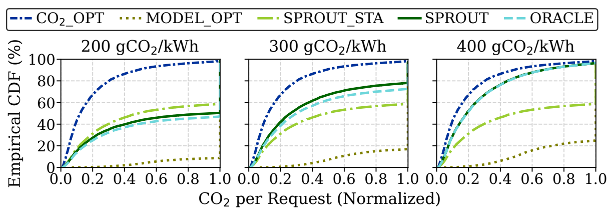


Figure 8: Cumulative distribution function (CDF) of per-request CO$_2$ emission normalized to Base as carbon intensity varies.

graphical regions in Table 2. As shown in Fig. 6, Sprout's application of optimized generation directives can reduce carbon emissions by up to 60%. The normalized generation preferences across all regions remain above the 90% mark, notably reaching over 95% in South Australia (SA) alongside a carbon saving exceeding 40%.

Below, we contextualize the magnitude of potential savings for easier interpretation, but do not claim that Sprout directly achieves them. For example, from an inference service provider perspective, according to a recent survey (de Vries, 2023), deploying OpenAI's ChatGPT service necessitates around 29K NVIDIA A100 GPUs, equating to an energy consumption of 564 MWh daily. In the Azure West US region of California (Microsoft, 2024), this translates to monthly CO$_2$ emissions of 3,266 tonnes. Adopting Sprout-like solution could result in a monthly carbon reduction of 1,903 tonnes – equivalent to offsetting the carbon footprint of flying 6,358 passengers from New York City to London (ICAO, 2024).

Sprout *outperforms competing methods, closely aligning with the* Oracle *standard.* Fig. 7 illustrates Sprout's performance against competing strategies outlined in Sec. 4, showcasing its proximity to the ideal Oracle in both carbon savings and normalized generation preference across all regions. Here, vertical lines denote the upper bound of generation preference in our evaluation, while horizontal lines indicate the upper bound of carbon savings. Unlike CO$_2$_Opt, which prioritizes carbon reduction at the expense of generation quality, Sprout maintains a balance closer to Base quality. While Model_Opt, Sprout_Sta, and Sprout exhibit similar preferences, Model_Opt falls short in carbon savings, highlighting the limitations of optimizing solely based on inference model variants (Romero et al., 2021; Wan et al., 2020). In contrast to its static version Sprout_Sta, Sprout demonstrates that its dynamic approach to generation directives yields results nearer to the Oracle benchmark, underscoring the effectiveness of adaptive configurations.

Analysis of the Sources of Sprout's Effectiveness and Evaluation Overhead. First, we show that Sprout dynamically adapts when carbon intensity varies. Fig. 8 presents the empirical cumulative distribution function (CDF) for 10K inference requests across three environmental carbon intensities: 200, 300, and 400 gCO$_2$/kWh. The x-axis scales the CO$_2$ emissions of each request relative to its execution on the Base system. Since we only show CO$_2$ per request, as expected, CO$_2$_Opt is the best among all the schemes – 80% of requests have used less than 30% of the Base carbon emission. When carbon intensity increases, Sprout's CDF moves closer and closer to CO$_2$_Opt, indicating that Sprout's optimizer is adapting to the regional carbon intensity since the gain from using more concise directives gets amplified at higher carbon intensities. Specifically, when carbon intensity is 200 gCO$_2$/kWh, 40% of Sprout's requests have used less than 40% of the carbon footprint than Base; when it increases to 400 gCO$_2$/kWh, about 75% of Sprout's requests have less than 40% of Sprout's carbon footprint. Unlike CO$_2$_Opt and Sprout_Sta, which do not adjust based on carbon intensity and thus maintain constant CDF curves, Sprout exhibits a dynamic adaptation that aligns closely with Oracle in a request-level analysis.

The offline quality evaluator is key to Sprout's effectiveness. In Fig. 9, we select Sprout-friendly prompts which are prompts whose shorter responses are on average more preferred by the auto-evaluator than their default responses, and mix them with unfriendly prompts (shorter responses are less preferred by auto-evaluator than default responses). Over time, we vary the proportion of these two types of prompts, and observe that when the portion of friendly is high, Sprout without the evaluator will miss out on the opportunity to save more carbon while achieving higher evaluator preference at the same time. As we can see

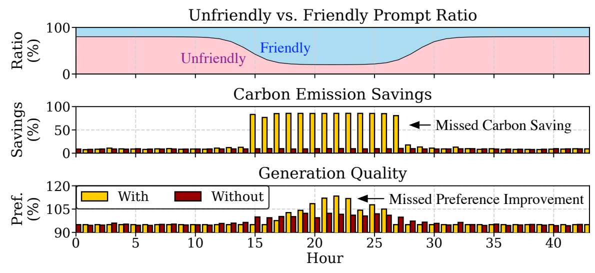


Figure 9: Without its offline evaluator, SPROUT misses the opportunity to leverage requests friendly to concise directive levels, thus forfeiting potential benefits in carbon savings and generation quality simultaneously.

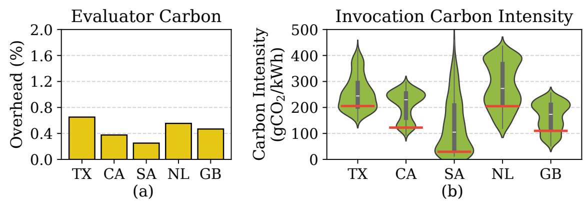


Figure 10: (a) Carbon overhead of SPROUT's offline evaluator. (b) Violin plot of evaluated region's carbon intensity distribution, and the carbon intensity where SPROUT invokes offline evaluation (marked as red line).

around hour 22, the normalized preference is above 100%, meaning the auto-evaluation LLM prefers SPROUT's generation over the default generation more than 50% of the time.

The offline evaluator's low carbon overhead is also a key contributor to SPROUT's carbon savings. In Fig. 10 (a), we show the carbon overhead of SPROUT's offline evaluator. Since GPT-4 is only accessible from third-party API, we use the following numbers to estimate the offline evaluation carbon footprint. GPT-4 is speculated to use a mixture-of-experts (MoE) architecture, and during inference, only one expert is active. Thus, the model size is equivalent to one expert that has 220B parameters, which can be hosted on 16 A100 GPUs. With the measured average API accessing time of 500ms, we assume all 16 GPUs are running at max power (250W), under no network delay and no batched processing. Despite our conservative estimation where in reality the GPU generation time is much shorter than 500ms (network latency, pre- and post-processing) and multiple requests can be processed simultaneously in a batch, the overhead in Fig. 10 (a) serving 30 requests per second (RPS) (Kwon et al., 2023) is still well below 1% for all regions. The minimal carbon impact stems from (i) strategically timing evaluations to coincide with periods of low carbon intensity as shown in Fig. 10 (b), and (ii) designing the request to the auto-evaluation LLM such that it generates only a minimal number of assessment tokens, as detailed in Appendix A.2.

We further show that SPROUT is robust across different seasons, and show the Pareto front of the carbon and quality trade-off in Appendix A.4.

6 Related Work

Sustainable AI (Wu et al., 2022) and Sustainable HPC (Li et al., 2023a) have explored various carbon trade-offs in ML infrastructure. Various works have analyzed the AI development's impact on carbon emission (Patterson et al., 2021, 2022; Schwartz et al., 2020; Acun et al., 2023b; Strubell et al., 2019; Anderson et al., 2023). SPROUT is motivated by these works and takes the effort a step further to LLM inference application. While systems like Carbon Explorer (Acun et al., 2023a), Ecovisor (Souza et al., 2023), Clover (Li et al., 2023b), and Dodge et al.(Dodge et al., 2022) have been designed to adapt to varying carbon intensities, they have not been specifically optimized for generative LLM inference workloads.

Previous works have explored pre-training and fine-tuning algorithms for controllable text generation, steering the generation towards specific lexical choices or sentiments (Zhang et al., 2023; Zhou et al., 2023; Dinu et al., 2019; Keskar et al., 2019). SPROUT suggests a promising new direction – controlling LLMs generation toward carbon efficiency. Various works have focused on performance and memory optimization, exploring strategies like sparsity and pruning (Liu et al., 2023c; Frantar and Alistarh, 2023), speculative decoding (Leviathan et al., 2023; Chen et al., 2023b), GPU kernel tiling and fusion (Dao, 2023; Zheng et al., 2023). These advancements are crucial for facilitating the deployment of larger LLMs to a broader audience. However, the environmental implications of these technologies are equally important. Carburacy (Moro et al., 2023) and LLMCarbon (Faiz et al., 2023) offer carbon footprint evaluations to help researchers gauge the environmental impact of LLM training, while SPROUT is the first work to tackle the carbon footprint challenge of generative LLM inference.

7 Conclusion

This paper introduced SPROUT, a framework to enhance the sustainability of generative language models. SPROUT can reduce the carbon footprint of LLM inference by over 40%, indicating a greener future for natural language generation.

8 Limitation

SPROUT may not be useful for requests that generate very short responses. In this case, adding a generation directive to the prompt may incur more carbon than not using directives. However, note that the extra carbon to process a longer input sequence that includes a generation directive is very limited as modern LLM serving systems maintain a KV cache, which stores key and value vectors from previously processed tokens without recomputing their KV vectors. The generation directive will be maintained in the KV cache after the initial pre-filing phase during LLM inference.

SPROUT is not evaluated on serving commercial LLMs such as ChatGPT and Gemini because these models are close-sourced. Our evaluation requires local deployment to perform carbon measurements.

Expert LLM users may send API requests and specify the system prompt. SPROUT will conservatively not apply generation directives to such requests as the directive may conflict user's system prompt (e.g., if the user explicitly asks for detailed responses).

9 Ethical Considerations

This work does not raise ethical concerns as we aim to align natural language generation with environmental sustainability.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Kiwan Maeng, Udit Gupta, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. 2023a. Carbon explorer: A holistic framework for designing carbon aware datacenters. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 118–132.

Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Aditya Sundarrajan, Kiwan Maeng, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. 2023b. Carbon dependencies in datacenter design and management. *ACM SIGENERGY Energy Informatics Review*, 3(3):21–26.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Thomas Anderson, Adam Belay, Mosharaf Chowdhury, Asaf Cidon, and Irene Zhang. 2023. Treehouse: A case for carbon-aware datacenter software. *ACM SIGENERGY Energy Informatics Review*, 3(3):64–70.

Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051*.

Anthropic. 2024a. Introducing the next generation of claude.

Anthropic. 2024b. System prompts.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2024. Benchmarking foundation models with language-model-as-an-examiner. *Advances in Neural Information Processing Systems*, 36.

Jaykaran Charan and Tamoghna Biswas. 2013. How to calculate sample size for different study designs in medical research? *Indian journal of psychological medicine*, 35(2):121–126.

Boyang Chen, Zongxiao Wu, and Ruoran Zhao. 2023a. From fiction to fact: the growing role of generative ai in business and finance. *Journal of Chinese Economic and Business Studies*, 21(4):471–496.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023b. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Andrew A Chien, Liuzixuan Lin, Hai Nguyen, Varsha Rao, Tristan Sharma, and Rajini Wijayawardana. 2023. Reducing the carbon impact of generative ai inference (today and in 2035). In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, pages 1–7.

Dimitrios Christofidellis, Giorgio Giannone, Jannis Born, Ole Winther, Teodoro Laino, and Matteo Manica. 2023. Unifying molecular and textual representations via multi-task language modelling. In *International Conference on Machine Learning*, pages 6140–6157. PMLR.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Alex de Vries. 2023. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3063–3068, Florence, Italy. Association for Computational Linguistics.

Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the carbon intensity of ai in cloud instances. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pages 1877–1894.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. 2024. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36.

FacebookResearch. 2024. Inference code for llama models.

Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Parteek Sharma, Fan Chen, and Lei Jiang. 2023. Llmcarbon: Modeling the end-to-end carbon footprint of large language models. *arXiv preprint arXiv:2309.14393*.

Fortune. 2024. Sam altman seeks trillions of dollars to reshape business of chips and ai.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S Lee, David Brooks, and Carole-Jean Wu. 2022. Act: Designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 784–799.

Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2021. Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 854–867. IEEE.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Qi Huangfu and JA Julian Hall. 2018. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142.

Huggingface. 2024. Mistral-7b-instruct-v0.1.

ICAO. 2024. International civil aviation organization carbon emissions calculator.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. 2021. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Baolin Li, Rohan Basu Roy, Daniel Wang, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2023a. Toward sustainable hpc: Carbon footprint estimation and environmental implications of hpc systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15.

Baolin Li, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2023b. Clover: Toward sustainable ai with carbon-aware machine learning inference service. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2024. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. 2023. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130.

Pengfei Liu, Yiming Ren, Jun Tao, and Zhixiang Ren. 2024. Git-mol: A multi-modal large language model for molecular science with graph, image, and text. *Computers in Biology and Medicine*, 171:108073.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023a. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Zequn Liu, Wei Zhang, Yingce Xia, Lijun Wu, Shufang Xie, Tao Qin, Ming Zhang, and Tie-Yan Liu. 2023b. MolXPT: Wrapping molecules with text for generative pre-training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1606–1616, Toronto, Canada. Association for Computational Linguistics.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023c. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Electricity Maps. 2024. Electricity Maps Live 24/7.

Microsoft. 2024. Azure global infrastructure experience.

MistralAI. 2024. Employ another llm for evaluation.

Gianluca Moro, Luca Ragazzi, and Lorenzo Valgimigli. 2023. Carburacy: summarization models tuning and comparison in eco-sustainable regimes with a novel carbon-aware accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14417–14425.

Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. 2023. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978.

OpenAI. 2024. Chat markup language.

David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean. 2022. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Natalie Pierce and Stephanie Goutos. 2023. Why law firms must responsibly embrace generative ai. *Available at SSRN 4477704*.

Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021. {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*, 63(12):54–63.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. 2023. Ecovisor: A virtual energy system for carbon-efficient applications. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 252–265.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Chengcheng Wan, Muhammad Santriaji, Eri Rogers, Henry Hoffmann, Michael Maire, and Shan Lu. 2020. {ALERT}: Accurate learning for energy and timeliness. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 353–369.

Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. 2022. {MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 945–960.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813.

Tong Xie, Yuwei Wan, Wei Huang, Zhenyu Yin, Yixuan Liu, Shaozhou Wang, Qingyuan Linghu, Chunyu Kit, Clara Grazian, Wenjie Zhang, et al. 2023. Darwin series: Domain specific large language models for natural science. *arXiv preprint arXiv:2308.13565*.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Ningxin Zheng, Huiqiang Jiang, Quanlu Zhang, Zhenhua Han, Lingxiao Ma, Yuqing Yang, Fan Yang, Chengruidong Zhang, Lili Qiu, Mao Yang, et al. 2023. Pit: Optimization of dynamic sparse deep learning models via permutation invariant transformation. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 331–347.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. In *International Conference on Machine Learning*, pages 42602–42613. PMLR.

A Appendix

A.1 Miscellaneous Design Considerations

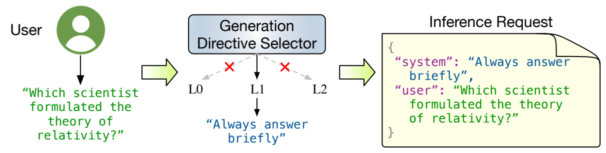
Role of auto-evaluation LLM. The auto-evaluation LLM, boasting orders of magnitude more parameters than the inference model, might seem like an ideal choice for processing user prompts. However, utilizing a giant model like GPT4, with its estimated 1.76 trillion parameters, entails considerable development, training, and deployment resources, making it impractical for most organizations due to high costs and environmental impact. Also, directly serving millions of user prompts on such a model incurs significantly more carbon emissions than a model with billions of parameters. Therefore, for most cases, it is better to fine-tune an open-sourced model like Llama to tailor to the user targets and use third-party LLMs like GPT4 for occasional quality feedback.

There may be instances where the auto-evaluator's preferences diverge from an individual user's expectations, as users might have varying inclinations toward the conciseness or detail of responses. In such cases, the inference service could proactively notify users when responses are condensed due to elevated carbon intensity levels, subsequently inquiring about their preference for more detailed answers. Should a user client express a preference for details, SPROUT can then specifically mark this preference by applying the baseline directive level, L0, to all their future prompts, ensuring tailored responses that align more closely with their expectations.

Number of evaluation samples. According to the sample size theory in (Charan and Biswas, 2013), 384 samples is an appropriate size for 95% confidence level and 5% margin of error. SPROUT uses a default 500 request samples to collect generation quality feedback, inference service providers can also adjust this number according to budget. This fixed sample size during offline evaluation has minimal impact relative to the total volume of prompts processed from the inference server. Consequently, the carbon emissions associated with these evaluations are deemed negligible and are not factored into the carbon footprint reduction strategy detailed in Sec. 3.2.

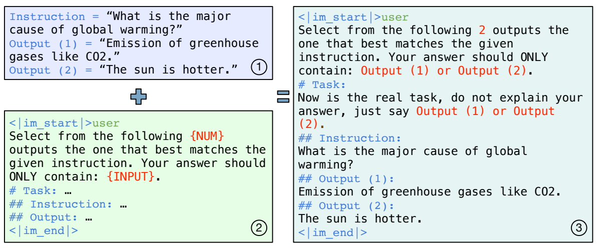
A.2 Implementation

Applying generation directive levels. The inference service provider specifies the number of directive levels and the actual directive sequence to apply for each level. SPROUT implements the generation directives as the system prompt alongside the user prompt, as the system prompt is widely accepted as a prompting format compatible with leading AI platforms like OpenAI ChatML (OpenAI, 2024), Llama (FacebookResearch, 2024), Anthropic Claude (Anthropic, 2024b), MistralAI (Huggingface, 2024),


User
"Which scientist formulated the theory of relativity?"
Generation Directive Selector
L0
L1
L2
"Always answer briefly"
Inference Request

```
{
 "system": "Always answer
   briefly",
 "user": "Which scientist
   formulated the theory
   of relativity?"
}
```

Figure 11: SPROUT implements generation directive level assignment as LLM system prompts.



```
Instruction = "What is the major
cause of global warming?"
Output (1) = "Emission of greenhouse
gases like CO2."
Output (2) = "The sun is hotter."
```

```
<|im_start|>user
Select from the following {NUM}
outputs the one that best matches the
given instruction. Your answer should
ONLY contain: {INPUT}.
# Task: …
## Instruction: …
## Output: …
<|im_end|>
```

```
<|im_start|>user
Select from the following 2 outputs the
one that best matches the given
instruction. Your answer should ONLY
contain: Output (1) or Output (2).
# Task:
Now is the real task, do not explain your
answer, just say Output (1) or Output
(2).
## Instruction:
What is the major cause of global
warming?
## Output (1):
Emission of greenhouse gases like CO2.
## Output (2):
The sun is hotter.
<|im_end|>
```

Figure 12: A simplified example of SPROUT's quality evaluation query. Box 1 represents the instructions and outputs generated using different directives, box 2 represents the template, and box 3 represents the ChatML (OpenAI, 2024) query to the evaluator LLM.

etc. Figure 11 illustrates SPROUT's method of incorporating a specific directive, such as the text from level L1, directly into the inference request as a system prompt. When a system prompt already exists within a user prompt, SPROUT conservatively discards the generation directive to avoid conflict with the user-specified system prompt.

Inference server and monitoring. SPROUT seamlessly integrates with existing inference server setups by processing system prompts together with user prompts, avoiding the need for infrastructure alterations. Mirroring industry-standard LLM inference practices, the server incorporates vLLM (Kwon et al., 2023) for its high-throughput and efficient KV cache management and utilizes FlashAttention (Dao, 2023) to streamline self-attention computations at the CUDA kernel level. To accurately log execution metrics as outlined in Eq. 2, the CarbonTracker (Anthony et al., 2020) package has been adapted to monitor each inference processing node, facilitating the calculation of $\mathbf{e}^T$ and $\mathbf{p}^T$ vectors essential for optimizing SPROUT's operation.

Automatic quality evaluation. We extend the AplacaEval (Li et al., 2024) project to build SPROUT's quality evaluator. Specifically, we generalized the auto-annotator to be able to query the auto-evaluation LLM to select the best one from an arbitrary number of generations, each corresponding to a specific generation directive level. We also implemented shuffling of the generations to remove position bias in the query. The evaluator is diligently implemented to prompt the auto-evaluation LLM to generate minimal tokens – just enough to identify the preferred output followed by the EOS token. This design is both carbon-efficient and cost-effective as commercial LLMs charge based on the number of tokens generated. A simplified example is shown in Fig. 12 where when we send the query to auto-evaluation LLM, it will generate "Output (1)" as the preferred output. We have manually examined the preference of several auto-evaluation LLMs (GPT-4, GPT-4 Turbo, GPT-3.5 Turbo) and confirm that the evaluator accurately identifies the correct response in over 97% of cases.

Table 1: Language modeling tasks to evaluate SPROUT.

Dataset	Description	Task
Alpaca (2023)	Instructions generated by OpenAI's `text-davinci-003`	Instruction tuning
GSM8K (2021)	Grade school math problems	Arithmetic and multi-step reasoning
MMLU (2020)	Massive multitask language understanding	Multiple-choice questions
Natural Questions (2019)	Real-user questions from Google	Question answering
ScienceQA (2022)	Science knowledge (e.g., Biology/Physics/Chemistry)	Multiple-choice science questions
TriviaQA (2017)	Trivia questions collected by trivia enthusiasts	Reading comprehension

Table 2: Geographical regions used to evaluate SPROUT.

Region	abbr.	Operator	Annual Min/Max
Texas (US)	TX	Electric Reliability Council of Texas (ERCOT)	124 / 494 (gCO_2/kWh)
California (US)	CA	California Independent System Operator (CISO)	55 / 331 (gCO_2/kWh)
South Australia	SA	Australian Energy Market Operator (AEMO)	10 / 526 (gCO_2/kWh)
Netherland	NL	TenneT	23 / 463 (gCO_2/kWh)
Great Britain	GB	National Grid Electricity System Operator (ESO)	24 / 282 (gCO_2/kWh)

A.3 Experimental Details

We randomly sample prompts from tasks in Table 1 to evaluate SPROUT. These tasks span various fields and applications, serving as critical benchmarks in performance evaluations for leading LLMs such as Llama (Touvron et al., 2023), Claude (Anthropic, 2024a), GPT (Achiam et al., 2023), Gemini (Team et al., 2023), as well as the ones used for scientific discovery (Singhal et al., 2023; Taylor et al., 2022; Xie et al., 2023; Almazrouei et al., 2023). To simulate realistic user prompts for the inference server, the composition of prompts from each task follows the request patterns from Alibaba's AI Platform trace (Weng et al., 2022), ensuring the evaluation comprehensively represents practical scenarios.

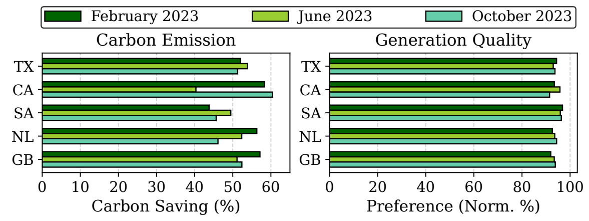


Figure 13: SPROUT remains effective during different seasons.

The evaluation of SPROUT extends across five grid operation regions in various countries, as described in Table 2. Given the variability in carbon intensity by region, this diversity enables a comprehensive assessment of SPROUT's performance in differing environmental contexts. The study uses carbon intensity data from February (default), June, and October of 2023, sourced from Electricity Maps (Maps, 2024) at hourly intervals, to gauge SPROUT's adaptability to fluctuating carbon intensity levels across these regions. Despite the offline evaluation LLM not being sensitive to latency and thus not requiring proximity to users – allowing it to be located in any global data center with the lowest carbon footprint. However, for a more cautious approach, we assume it resides in the same region as the inference server.

A.4 Robustness and Implications

We also assess the robustness of SPROUT and its broader implications. Fig. 13 presents an evaluation of SPROUT across various periods of 2023 (different carbon intensity variation patterns), demonstrating its consistent efficacy across different seasons. SPROUT consistently enables the inference server to achieve over 40% carbon emission savings while sustaining high levels of generation quality.

SPROUT offers inference service providers the ability to balance carbon savings against quality through the adjustable parameter ξ. Fig. 14 illustrates the Pareto front demonstrating the trade-off between carbon savings and generation quality as ξ is varied. Notably, even when tightening the generation preference criterion to 95% (indicating the evaluator prefers SPROUT's generation 48.7% and the default 51.3% of the time), SPROUT consistently secures over 40% carbon savings across all regions.

To the best of our knowledge, SPROUT is the first approach to utilizing generation directives for generative LLM inference, with a particular emphasis on advancing its environmental sustainability. This strategy opens up extensive possibilities beyond

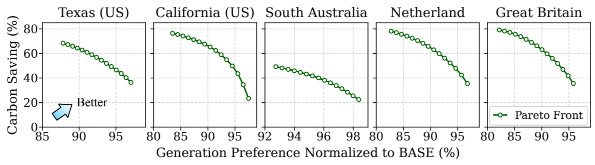


Figure 14: Pareto front of SPROUT across geographical regions.

its current focus. For instance, using generation directives can significantly enhance LLM inference throughput, thereby reducing the number of GPU servers needed to achieve specific rates of requests per second (RPS). This efficiency translates into reduced capital expenses for building LLM inference infrastructure and lowers the embodied carbon associated with manufacturing the GPU servers.