Evaluating LLM Knowledge Placement Techniques for Carbon-Efficiency

RANKYUNG HONG, University of Minnesota, Twin Cities, USA
ABHISHEK CHANDRA, University of Minnesota, Twin Cities, USA

Large language models increasingly answer questions over a fixed body of knowledge. Practitioners supply that knowledge in three common ways: retrieval-augmented generation (RAG) looks it up at query time, prefix caching reuses shared work, and LoRA fine-tuning trains it into the model. Prior comparisons measure quality and speed, but not carbon. We compare the trade-offs between all three metrics by varying model size and context length. Our results show that when the knowledge is stable and matches the questions asked, fine-tuning is the greenest, fastest, and most accurate, because it produces shorter answers. Bigger models cost far more carbon than smaller models. Above all, running inference on a clean electricity grid lowers total carbon the most.

CCS Concepts: • **Social and professional topics** → **Sustainability**; • **Computing methodologies** → **Natural language processing**; **Natural language generation**; **Machine learning**; • **Hardware** → *Impact on the environment*.

Additional Key Words and Phrases: Sustainable computing, LLM inference, retrieval-augmented generation, prefix caching, LoRA fine-tuning, parameter-efficient fine-tuning, carbon measurement, knowledge placement

1 INTRODUCTION

Large language models (LLMs) now power knowledge-intensive question answering: product support, policy lookup, scientific Q&A, and so on. Inference for these workloads is now a non-trivial slice of datacenter energy [14, 17]. Practitioners building such systems typically pick between three strategies that materialize the required knowledge at different points in the inference stack — in the prompt, in retained activations, or in the model's weights — each with a different per-query cost profile.

(a) Retrieval-augmented generation (RAG) [12] keeps the knowledge in an external index. At query time the system embeds the question, retrieves the most relevant passages from a vector store, and pastes them into the prompt that the LLM sees. The model itself is unchanged. **(b) Prefix caching**, exemplified by the PagedAttention mechanism in the vLLM serving system [10], keeps the knowledge in a frequently-reused part of the input. The model's intermediate state (the "KV cache") for a shared prompt prefix is computed once and reused across queries that share that prefix, avoiding redundant prefill compute. **(c) Lightweight fine-tuning** via Low-Rank Adaptation (LoRA) [6] pushes the knowledge into the model's weights. LoRA trains a small number of low-rank adapter matrices alongside the frozen base model (typically about 0.1% of the parameters) and merges them at inference time. The choice between (a), (b), and (c) is workload-dependent and interacts with both model scale and context length, but published head-to-head comparisons report quality and latency [1, 16, 22] without measuring carbon under a unified accounting.

Authors' addresses: Rankyung Hong, hongx293@umn.edu, University of Minnesota, Twin Cities, Minneapolis, Minnesota, USA; Abhishek Chandra, chandra@umn.edu, University of Minnesota, Twin Cities, Minneapolis, Minnesota, USA.

This gap matters because the three strategies have qualitatively different carbon profiles: RAG inflates input length, cache amortizes prefill across queries that share a prefix, and fine-tune trades a one-time training cost for cheaper per-query inference. Whether and when each strategy wins on *carbon*, not just quality, is unmeasured.

The three strategies are not always direct substitutes — RAG is the natural fit when knowledge is frequently updated or external, prefix caching exists primarily to improve serving throughput, and LoRA is typically used for to adapt format or specialize a model. They are, however, the three options a practitioner picks between when deciding *how to supply knowledge to an LLM* under joint quality, latency, and now carbon constraints. Our aim is to chart the carbon trade-offs across that decision so that sustainability becomes one of the axes a deployment owner can reason about, rather than a property they have to discover post-hoc.

Where this fits in prior LLM inference carbon work. Recent measurement and optimization studies have addressed LLM sustainability from three angles complementary to ours. Nguyen et al. [15] characterized LLaMA inference at 1B/3B/7B parameters across two GPU generations and three grid regions, modelling operational and embodied carbon analytically. Li et al. [13] refined CPU+GPU lifecycle carbon models and showed that combined batching, sharding, and parallelization can deliver a 17% carbon improvement at iso-throughput. Wilkins et al. [24] built workload-conditioned energy and runtime models for LLM inference and used them for energy-optimal scheduling on heterogeneous CPU-GPU systems. Each treats inference of a *single fixed model+context* configuration as the unit of measurement and optimizes the hardware-and-scheduling axis. The strategy axis (*which knowledge-placement configuration to deploy in the first place*) has not been measured under unified carbon accounting. Our paper supplies that axis: holding hardware and scheduler fixed, we vary how knowledge is supplied to the model.

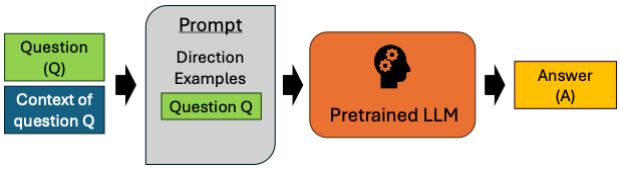
Why a practitioner-facing measurement is needed. Existing quality-only head-to-head comparisons disagree on the right choice [1, 16, 22]. Some favor RAG (for rarely-seen facts and frequently-changing data), others favor fine-tuning (style and format adaptation), and recent work argues for hybrid RAG plus fine-tuning pipelines [27]. None reports per-query carbon, none includes prefix caching as a co-equal third option, and none compares all three under a unified carbon-accounting model. A deployment owner trying to translate this literature into a carbon-informed choice is left interpolating between studies that share neither metric nor configuration. We close that gap with a controlled 2×2 where the same identical questions traverse all three strategies at two model scales, allowing paired comparisons (§5) and operational+embodied carbon attribution under a single accounting model.

This paper makes the following contributions:

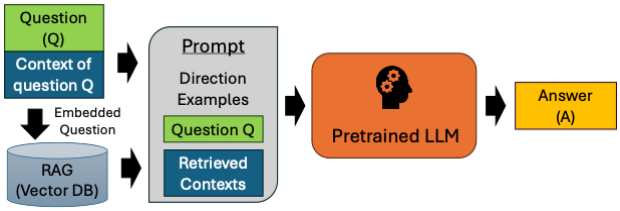
- **A unified, paired carbon comparison.** We compare three knowledge-placement strategies (RAG, prefix caching, LoRA fine-tuning) plus a no-augmentation baseline, across model scale and context length, with identical question sets traversing every method so that comparisons are paired, counting both operational and embodied carbon [2, 14, 20].
- **A mechanism for the carbon ordering.** Per-query carbon orders consistently as fine-tune < cache < RAG across every cell we measure; the mechanism is output length, since decoded tokens dominate per-query energy at fixed model size. The carbon and latency claims are mechanism-driven and dataset-agnostic; the accompanying quality lead is conditional on in-distribution evaluation.
- **Winning the break-even race does not mean winning on carbon.** Replaying the LoRA amortization under real US state-grid CIs (EPA eGRID 2022) shifts the fine-tune-vs-RAG break-even by ~ 6.5×. On a dirty inference grid, fine-tune's shorter answers save more carbon per query, so the training cost is repaid in fewer queries — but those same queries are all dirtier, so the fastest-amortizing scenario emits the *most* total carbon. What matters is long-run deployment carbon, set by where inference runs: the lowest-carbon scenario is the one served on a clean grid, regardless of break-even speed.
- **Actionable guidance for practitioners.** We distill the findings into three rules: fine-tune on stable, in-distribution knowledge; place inference on the cleanest grid available; and use the smallest model that meets the accuracy bar, since a larger model costs several times more carbon per query for accuracy gains that may not be needed.

2 BACKGROUND AND RELATED WORK

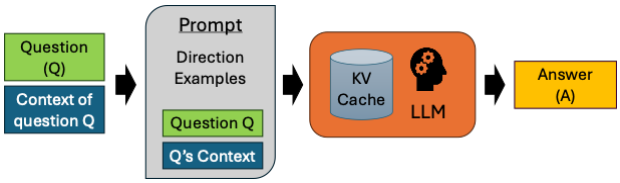
Knowledge-placement strategies. The three strategies sit on what we call a *persistence-of-materialization spectrum*: the relevant knowledge can live in the prompt, in retained activations, or in the model's weights, and each choice has a different per-query cost (Fig. 1). Retrieval-augmented generation (RAG) [3, 12] embeds the user's question, retrieves the most similar passages from a pre-built dense index of the corpus, and prepends those passages to the prompt the model sees (Fig. 1b). The knowledge is materialized *transiently* — in the prompt, recomputed on every query. The model itself is unchanged. Prefix caching reuses the model's per-token attention state (the "key-value cache") across queries that begin with the same context. The PagedAttention mechanism [10] in the vLLM serving system organizes this state in fixed-size pages so that shared prefixes can be referenced rather than recomputed (Fig. 1c). The knowledge is materialized *intermediately* — a precomputed "model's reading" of a knowledge-bearing prefix is kept alive across queries that hit it. Low-Rank Adaptation (LoRA) [6] keeps the base model's weights frozen and trains a pair of small matrices added to a subset of the attention weights (the query and value projections). Only ~0.1% of the parameter count is updated, which makes training cheap and the adapters quick to ship (Fig. 1d). The knowledge is materialized *persistently* — baked into the weights and reused at no extra prompt



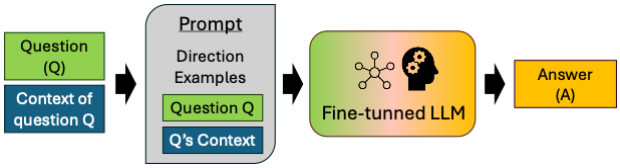
(a) Base — no augmentation.



(b) RAG — retrieved contexts from a vector DB are added to the prompt.



(c) Cache — vLLM reuses the prefilled KV state of a shared prefix.



(d) Fine-tune — LoRA-adapted weights replace the base pretrained LLM at inference.

Fig. 1. The four pipelines compared in this paper. Knowledge is placed differently in each: in the prompt via retrieval (RAG), in the model's KV cache (Cache), or in the model's weights via LoRA adapters (Fine-tune); Base is shown for contrast.

cost. For contrast, the *Base* pipeline (Fig. 1a) sends the question to the pretrained LLM with no augmentation.

We note that, strictly speaking, prefix caching was designed as a serving optimization rather than a knowledge-placement mechanism: the model's weights are unchanged and the prompt is unchanged. We include it on the spectrum because, once a knowledge-bearing prefix is retained, its key-value state functions as a precomputed, queryable representation of that prefix — an intermediate, least aggressive form of placement between transient prompt injection (RAG) and permanent weight modification (LoRA). This dual view is what motivates a unified carbon comparison: practitioners co-deploy the three for overlapping reasons even when their primary design intents differ.

ML carbon accounting. Operational accounting follows the energy × grid-CI × PUE (power usage effectiveness) chain established in [17]. Lacoste et al. [11] popularized per-region reporting. Embodied carbon (GPU manufacturing, amortized) is included following [14], which reports 0.003 kg CO_2 per GPU-hour for A100-class hardware, the constant we use. Faiz et al. [2] provides a predictive operational+embodied model for LLMs. Samsi et al. [20] measures

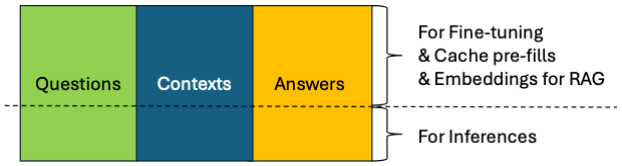


Fig. 2. In-distribution split. The same SQuAD/TriviaQA data — questions, contexts, and answers — is partitioned: the train half supplies LoRA fine-tuning targets, the RAG dense index, and Cache's prefilled prefix contexts; the eval half supplies the questions answered (and scored) at inference.

inference energy across LLaMA scales but does not compare placement strategies. Wu et al. [25] argues embodied carbon must be reported alongside operational. Kopczyk and Chandra [9] re-examines the storage carbon of machine-learning workloads, a component our compute-focused accounting does not capture.

RAG-vs-fine-tune comparisons. Several recent works compare RAG and fine-tuning on quality but not carbon. Ovadia et al. [16] reports RAG beats unsupervised continued-pretraining fine-tuning on knowledge injection. Balaguer et al. [1] treats them as additive on a domain-adaptation case study. Soudani et al. [22] shows RAG dominates fine-tuning on tail entities. None of these works measures per-query carbon.

3 METHODOLOGY

Models. TinyLlama-1.1B-Chat-v1.0 [26] is a small open instruction-tuned model representative of the "edge / single-GPU" deployment class. Llama-3.1-8B-Instruct [4] is a mid-sized instruction-tuned model that fits comfortably on a single 40 GB datacenter GPU. Two model sizes let us see whether the strategy ranking changes with scale.

Datasets. SQuAD [18] is a reading-comprehension benchmark over Wikipedia passages with short (~150-token) contexts and span-extractive answers. TriviaQA [8] pairs trivia questions with long Wikipedia / web evidence documents (often multi-thousand tokens) and accepts entity-style answers with multiple aliases. Each dataset ships question–context–answer triples; we use the train split to fine-tune the LoRA adapter, to build the FAISS index that RAG retrieves from, and as the source of the contexts that Cache pre-fills its KV state on, while the validation split provides the questions we measure at inference (Fig. 2).

Methods per cell. The *Base* pipeline sends each question to the unmodified pretrained model with no augmentation, included as a reference point. *(a) RAG*: FAISS [7] dense index over the training passages, top-$k=3$ retrieval, prepended to the prompt. Embeddings: `all-MiniLM-L6-v2` [19]. *(b) Prefix caching*: vLLM [10] with automatic PagedAttention prefix caching enabled. Questions are grouped by shared context to maximize hit-rate. *(c) LoRA fine-tuning*: small trainable adapter matrices (rank $r=8$, scaling $\alpha=16$) attached to each layer's query and value attention weights (`q_proj`, `v_proj`), trained on the dataset's training split.

Question-pairing protocol. Within a dataset, all three methods see identical question sets in identical order at both model scales (verified across all twelve (cell, method) result files). Method comparisons within a cell and scale comparisons within a dataset are therefore paired. Cross-dataset comparisons use disjoint question sets and we interpret them with caution.

Table 1. Lifecycle CIs (IPCC AR5 [21]) and example-state grid CIs (EPA eGRID 2022 [23]), in g CO_2eq/kWh.

Source	Lifecycle CI	Example state	Grid CI
Wind	11	Iowa (63% wind)	282
Solar PV	48	California (20% solar)	458
Natural gas	490	Florida (74% gas)	819
Coal	820	West Virginia (90% coal)	895

Metrics. *Token F1* (official SQuAD): let $\hat{y}$ and g be the bags of normalized tokens of the prediction and a gold answer, with overlap $c = \sum_t \min(\hat{y}_t, g_t)$; then

$$\mathrm{F1}(\hat{y}, g) = \frac{2c}{|\hat{y}| + |g|}, \qquad \mathrm{F1}^{\star} = \max_{g \in \mathcal{G}} \mathrm{F1}(\hat{y}, g),$$

reported as the maximum over the set $\mathcal{G}$ of gold answers. *Semantic similarity* (SemSim) is the maximum cosine similarity of MiniLM embeddings of the prediction and gold:

$$\mathrm{SemSim} = \max_{g \in \mathcal{G}} \frac{\mathbf{e}_{\hat{y}} \cdot \mathbf{e}_g}{\|\mathbf{e}_{\hat{y}}\| \, \|\mathbf{e}_g\|}.$$

SemSim is format-tolerant: it scores "Paris" and "The capital of France is Paris" similarly, so it disentangles the answer's information content from its surface form. *Per-query carbon* is the sum of operational and embodied components:

$$C_q = \underbrace{E_q \cdot \mathrm{CI} \cdot \mathrm{PUE}}_{\text{operational}} + \underbrace{T_q \cdot \epsilon_{\mathrm{GPU}}}_{\text{embodied}},$$

where E_q is per-query GPU energy (codecarbon), CI is grid carbon intensity, PUE = 1.2 follows [14], T_q is per-query GPU-hours, and $\epsilon_{\mathrm{GPU}} = 0.003$ kg CO_2 per A100-hour is the amortized manufacturing cost from the same source. Training carbon is computed identically and amortized over query volume in F2.

4 EXPERIMENTAL SETUP

Hardware and software. Single NVIDIA A100 40 GB on a commodity cloud GPU instance. Python 3.11, vLLM (automatic prefix caching enabled), HuggingFace `transformers` and `peft` for LoRA training, `sentence-transformers` for the MiniLM retriever and semantic-similarity metric, FAISS-CPU for the dense index, and `codecarbon` for operational energy logging.

Carbon intensity. Operational CO_2 is computed under codecarbon's grid carbon intensity for the United States (~369 g CO_2/kWh, codecarbon country-average fallback; US-average from EPA eGRID 2022 is the comparable 375 g/kWh [23]). Our cloud instance did not expose a sub-region that codecarbon could resolve. *Scope of CI in our results.* Findings F1–F3 hold the grid CI fixed at this US-average, so the cross-method differences they report are driven by measured energy and embodied-hour differences, not by CI variation; in that sense they are energy-grounded. F4 is where real-world CI variation enters: we replay the same energy traces against four US state-grid mixes (EPA eGRID 2022) to test how regional placement reshapes the break-even and absolute carbon. Real-time temporal CI variation is a separate axis we discuss as future work (§6). Table 1 summarizes the carbon-intensity values used elsewhere in this paper under two reporting conventions. *Lifecycle* CI counts the CO_2-equivalent emissions of manufacturing, construction, operation, and end-of-life

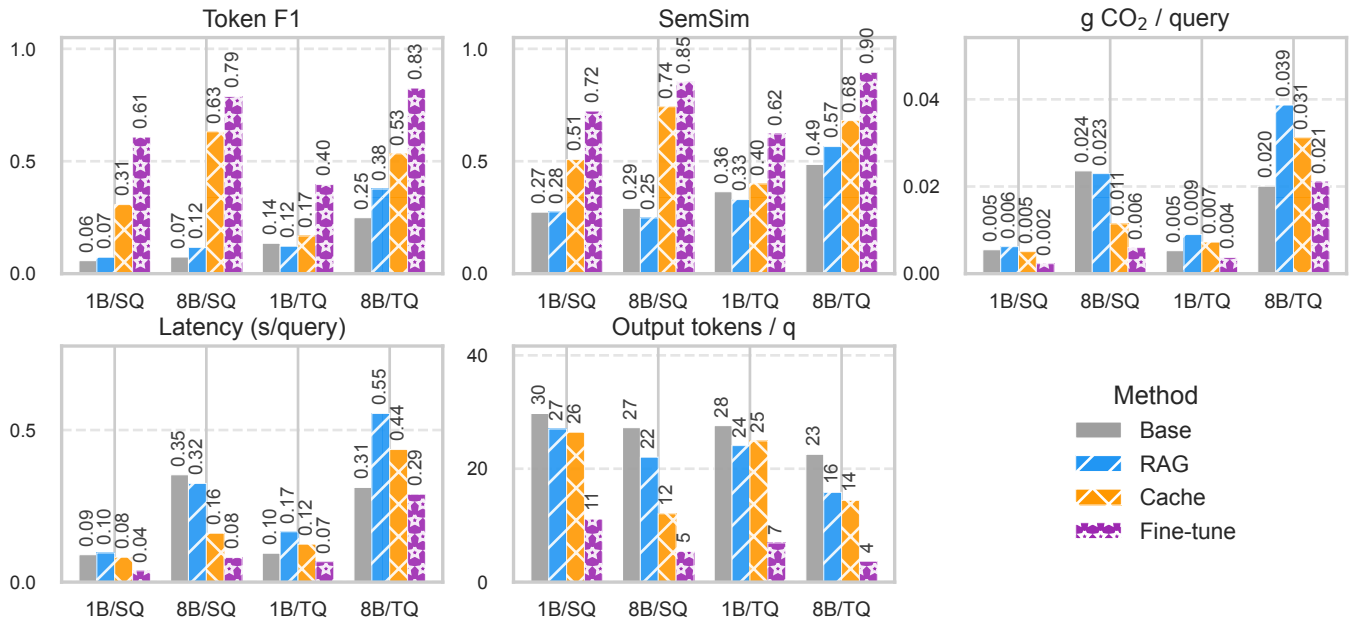


Fig. 3. Main 2×2 benchmark across {1B, 8B} × {SQuAD (SQ), TriviaQA (TQ)}. Four methods: *Base* (gray, no augmentation), *RAG* (blue), *Cache* (orange), *Fine-tune* (purple, LoRA on the same dataset's train split). Five panels: Token F1, SemSim, g CO_2/query, latency, output tokens/query. Fine-tune wins on every panel in every cell; the carbon advantage over the next-best method (Cache) is 1.5–2.2× and traces to a 2.4–3.6× reduction in output tokens.

for a generation technology amortized per kWh produced (CO_2-equivalent aggregates CO_2, methane, and N_2O by their 100-year warming effect); the *Intergovernmental Panel on Climate Change* (IPCC), the UN climate-science body, publishes the standard medians in its Fifth Assessment Report (AR5, 2014) [21]. *Grid* CI, in contrast, is the operational-only emission rate of a real grid mix (e.g., a US state), reported here from EPA eGRID 2022 [23]; it sits well above the dominant source's lifecycle CI because no real grid is 100% one technology.

Workload sizes. 300 SQuAD validation questions split equally across three prefix-reuse regimes: *low* (100 questions over 100 distinct passages, 1 question per passage), *medium* (100 questions over 20 passages, 5 per passage), and *high* (100 questions over 5 passages, 20 per passage). The three regimes share the same total query count but vary the number of distinct prefixes the cache can exploit: *low* approximates open-domain QA where each question hits a different passage. *High* approximates customer-support FAQ or tail-entity lookups where many users ask about the same document. We additionally evaluate 200 TriviaQA validation questions in a single "all" regime, since TriviaQA's longer evidence documents do not cluster like SQuAD's shared-passage structure.

Pipeline and reproducibility. LoRA: rank 8, α=16, applied to `q_proj/v_proj` only; bf16; effective batch 32; learning rate 2×10^{-4} (1B) / 1×10^{-4} (8B); 3 epochs on SQuAD, 2 on TriviaQA. Decoding is greedy (temperature 0, `max_new_tokens` = 32) using each model's native chat template (§3). Workloads use a fixed random seed that deterministically splits the 300 SQuAD questions across the three reuse regimes; the pretrained models are unmodified Hugging Face releases (TinyLlama-1.1B-Chat-v1.0, Llama-3.1-8B-Instruct, all-MiniLM-L6-v2).

5 RESULTS

We score predictions with Token F1 and semantic similarity (SemSim; defined in §3), and measure cost as per-query carbon (operational plus embodied) and per-query latency.

Figure 3 consolidates all 16 (cell, method) numeric results. We report four findings (F1–F4) below.

F1 — Fine-tune dominates for in-distribution dataset. The findings below evaluate train and test on splits of the same dataset. Under that scope, fine-tune simultaneously achieves the highest Token F1, highest SemSim, lowest per-query carbon, and lowest latency in all four cells (Fig. 3). Margin over the next-best method (Cache) ranges from +0.16 to +0.30 on F1 and +0.11 to +0.22 on SemSim. *Mechanism:* fine-tune emits 2.4–3.6× fewer output tokens than Cache and ~4× fewer than RAG. Decode is the largest per-query energy term at fixed model size [20] but prefill is not negligible: at 8B/SQuAD the 2.4× token ratio yields only a 1.9× carbon ratio (fine-tune 0.006 vs. Cache 0.011 g/q), implying prefill contributes roughly 20% of per-query energy on this workload. Output length is therefore the dominant lever, not the only one. Per-question pairing on identical inputs confirms decisive dominance vs. Cache (67%–79% of non-tied questions) and vs. RAG (67%–99%). The quality lead is conditional on in-distribution evaluation; the carbon and latency wins are mechanism-driven and dataset-agnostic.

F2 — Scale tradeoff: 8B sharpens quality and shortens output, but per-query carbon rises sharply. Going 1B→8B raises fine-tune Token F1 from 0.61 → 0.79 on SQuAD and 0.40 → 0.83 on TriviaQA, while output tokens drop (11 → 5 on SQuAD, 7 → 4 on TriviaQA). The cost: per-query carbon climbs 2.6× on SQuAD (0.0023 → 0.006 g/q) and 5.7× on TriviaQA (0.0037 → 0.021 g/q).

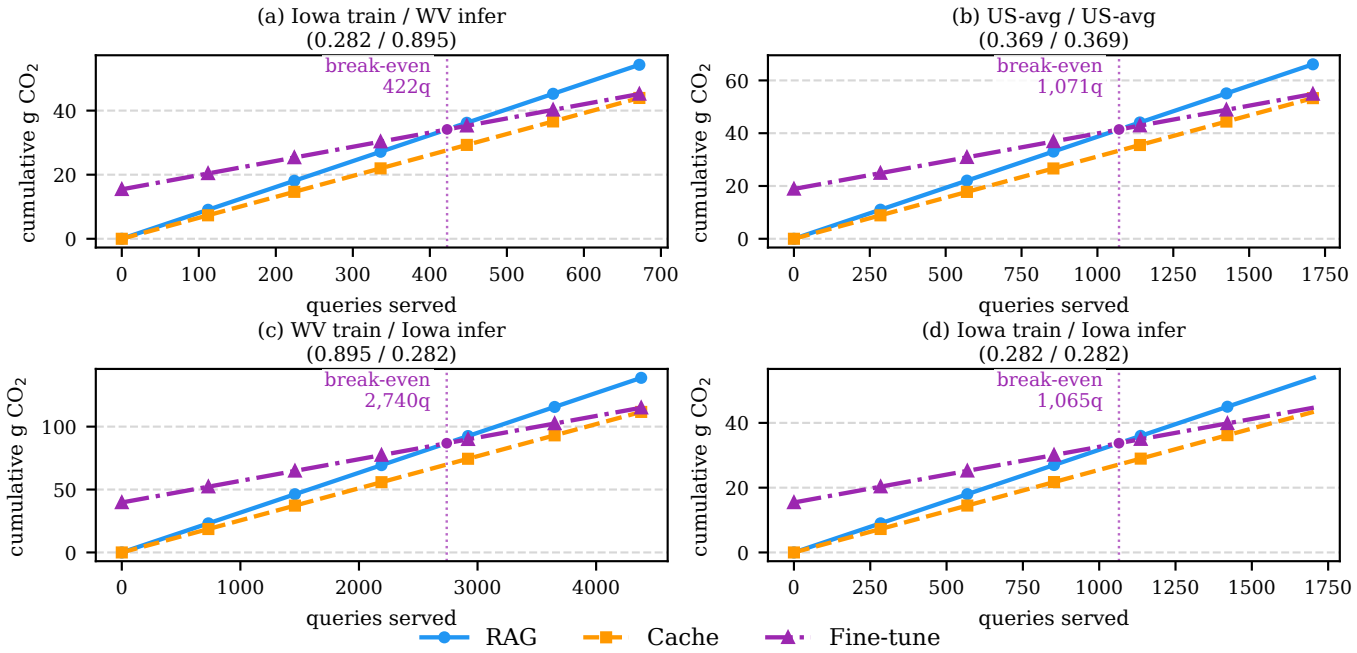


Fig. 4. Cumulative CO_2 vs. queries served, 8B+TriviaQA cell, under four (train CI, infer CI) scenarios spanning the codecarbon US-average (0.369 kg/kWh) and real US-state grid CIs from EPA eGRID 2022 [23] — Iowa (wind-heavy, 0.282) and West Virginia (coal-heavy, 0.895). (a) Iowa training, West Virginia inference: break-even at 422 queries (low training cost plus a large per-query gap pull break-even down). (b) US-average baseline: 1,071 queries. (c) West Virginia training, Iowa inference: 2,740 queries (high training cost plus a tiny per-query gap push break-even out). (d) Iowa for both: 1,065 queries — similar break-even to the US-average baseline, but the lowest absolute carbon of the four scenarios because clean inference multiplies across every query. Break-even varies by ~ 6.5× across scenarios; total per-cell carbon is set primarily by the inference grid, since inference is the multiplied cost.

The strategy ranking still holds (carbon ratio over Cache stays in 1.5–2.2× because the tok/q ratio is roughly scale-invariant at 2.4–3.6×), but a deployment owner picking 8B for its accuracy must budget several-fold higher carbon per query.

F3 — RAG adds prefill cost without meaningful quality gain on in-distribution data. RAG's quality lift over Base is small on SQuAD (+0.01 at 1B, +0.05 at 8B) and negative on 1B/TriviaQA (−0.02); only on 8B/TriviaQA does it deliver a meaningful +0.13 Token F1. Yet per-query carbon rises by up to 95% (8B/TriviaQA Base 0.020 g/q → RAG 0.039 g/q), and latency by up to 1.8× (0.31 s → 0.55 s on the same cell). RAG also fails to compress output: tok/q stays at 16–27, close to Base's 23–30 and well above fine-tune's 4–11. The model echoes retrieved context rather than producing a terse extractive answer. For in-distribution QA where fine-tune is feasible, the retrieval cost buys little. *This finding is scoped to static, in-distribution knowledge.* RAG's value rises sharply when the corpus changes faster than the re-training cadence, when answers depend on rare entities the base model has not memorized, or when users must be able to trace each answer back to the passage it came from [22]. We revisit this boundary in §6.

F4 — Training cost amortizes within hundreds of queries; regional grid placement swings break-even by ~ 6.5×. At US-average grid intensity, fine-tune-vs-RAG break-even sits at ~230 queries (1B+SQuAD), ~190 (8B+SQuAD), ~1,010 (1B+TriviaQA), and ~1,070 (8B+TriviaQA) — all within typical production lifetimes. Under real US-state grid mixes (Fig. 4, EPA eGRID 2022), the same 8B+TriviaQA break-even shifts from 422 queries when training is in clean wind-heavy Iowa and inference is in coal-heavy West Virginia, to 2,740 queries when the placement is reversed — a ~ 6.5× swing driven purely by where the energy is drawn. *Smaller break-even is not necessarily better.* The break-even is fastest when inference runs on a dirty grid: a high carbon intensity magnifies the per-query carbon that fine-tune saves with its shorter answers, so the training cost is paid back in fewer queries — but the same dirty grid makes every query more expensive. Scenario (a) above wins the break-even race but emits the most absolute carbon (236 g at 5,000 queries) because every inference query is run on a dirty grid; scenario (d) Iowa-for-both has a near-baseline break-even but the lowest total carbon (101 g) since clean inference multiplies across every query. Inference grid placement, not training grid, dominates long-run carbon. Training carbon is dominated by sequence length, not model size: 1B+TriviaQA emits 5.34 g while 8B+SQuAD emits only 3.27 g despite 8× the parameters; 8B+TriviaQA tops at 18.9 g. Per-query carbon is 67%–78% operational and 22%–33% embodied across all 16 configurations; ignoring embodied would understate total impact by roughly a quarter [5, 25]. The strategy ranking is robust to ±50% perturbation of grid carbon intensity.

Reading F1–F4 together. The four findings cohere into a small set of practitioner rules:

- *Default to LoRA on the stable corpus, with prefix caching layered on top as a serving optimization* (F1, F2). When the corpus is static and matches the deployment's evaluation distribution, you pay a one-time training cost that amortizes inside a few hundred to a thousand queries (F4), and you reap multiplicative savings on every query thereafter through shorter outputs.
- *Reach for RAG in the boundary cases of F3* — a fast-changing corpus, rare entities the base model has not memorized, or a need to trace answers back to their source — where the prefill cost is the price of correctness, not waste.
- *Put inference where the grid is cleanest* (F4). Inference carbon is paid on every query while training is one-time, so the inference region (wind- or hydro-heavy grids such as Iowa or Iceland), not the training region or how quickly the model amortizes, sets the long-run carbon ceiling; a clean training region still helps but is the smaller lever.
- *Size the model to the accuracy you actually need* (F2). The 8B model is more accurate but emits several times more carbon per query than the 1B model, so when top accuracy is not essential the smaller model is the carbon-efficient default.

6 DISCUSSION AND FUTURE WORK

Our experiments span a narrow slice of the design space; we discuss how far the findings travel and where they need re-measurement.

Generalizability of the carbon ordering. Our numbers come from a single A100 40 GB GPU at a fixed datacenter-typical PUE of 1.2. Two axes are worth discussing. *(a) Datacenter overhead.* Real-world PUE ranges from ∼1.1 in hyperscale facilities to 1.5–2.0 in on-premise rooms, and cooling shifts with climate and load; since per-query carbon scales linearly in PUE, absolute numbers in F1–F3 move proportionally, but the *rank ordering* fine-tune < cache < RAG is preserved because all three strategies share the same PUE multiplier on the same hardware. *(b) Spatial grid heterogeneity.* F4 already replays the 8B+TriviaQA cell against four US state grids spanning 0.282–0.895 kg CO_2/kWh, a 3.2× swing; the strategy ranking holds across all four. We have not measured cross-continent grids (e.g., Iceland hydro, Australia coal-heavy) or distributed training and inference, both of which could perturb the per-query energy distribution; we return to them in the cross-region and distributed-execution directions below. The mechanism behind the ordering (LoRA cuts output tokens; decode dominates per-query energy at fixed model size) is hardware- and grid-agnostic, which gives us some confidence the qualitative finding travels even as the absolute numbers do not.

Four directions follow from this study:

- **Workload families beyond QA.** LoRA's carbon win hinges on terse extractive answers. Summarization, multi-hop QA with chain-of-thought, and code generation produce long structured outputs that weaken the decode-saving mechanism, and RAG's value rises when answers require multi-document synthesis. Because our entire carbon advantage traces to shorter outputs, the ranking could narrow or invert on these long-output tasks; locating that crossover is the natural next step.
- **Hybrid LoRA + selective RAG for streaming knowledge.** A hybrid that fine-tunes on the stable corpus and retrieves only documents updated since the last training cycle bounds RAG's per-query cost to a small "freshness delta" while preserving LoRA's decode advantage on the stable majority. The open question is the re-training cadence that minimizes total carbon: too frequent and the training cost dominates, too rare and the freshness delta (and its retrieval cost) grows.
- **Cross-region carbon-intensity replay.** Figure 4 fixes train/infer pairs; replaying GPU-hour traces against real-time per-region CI would test whether the ranking survives hour-by-hour fluctuation. And because a production service runs many instances whose CI varies by location and time, routing queries to cleaner instances — and deferring the one flexible cost, training, to greener hours — are further carbon levers.
- **Distributed inference and training.** Splitting a single model's inference across multiple GPUs spreads the KV cache over those machines, changing how often cached prefixes can be reused, while training across GPUs spends extra energy keeping them synchronized — either effect could shift the per-query carbon balance. Inter-GPU communication and idle-time energy are invisible to single-GPU measurement, so the cache-vs-fine-tune gap in particular may look different at multi-GPU scale.

7 CONCLUSION

When an LLM answers questions over a fixed, unchanging body of knowledge, training that knowledge into the model with LoRA fine-tuning is the best choice on every axis we measured — answer quality, speed, and carbon — across both model sizes and both context lengths. The carbon and speed wins come from a simple mechanism: a fine-tuned model produces much shorter answers (2–4× fewer generated words than the next-best method), and generating words is where most of the energy is spent, so these wins hold regardless of the test set. The accuracy win, by contrast, holds only when the questions resemble the data the model was trained on. A larger model answers more accurately but costs several times more carbon per query (2.6–5.7× going from 1.1B to 8B), and RAG's retrieval step adds cost without buying much accuracy here. The one-time training cost is repaid within a few hundred to about a thousand questions; after that, what dominates a deployment's total carbon is not how quickly training pays off but *where* the system runs — placing inference on a clean electricity grid matters most. In short, for stable knowledge: fine-tune the model, serve it on the cleanest grid available, and pick the smallest model that meets your accuracy bar.

REFERENCES

[1] Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, et al. 2024. RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture. *arXiv preprint arXiv:2401.08406* (2024). https://doi.org/10.48550/arXiv.2401.08406

[2] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Chukwunyere Osi, Prateek Sharma, Fan Chen, and Lei Jiang. 2024. LLMCarbon: Modeling the End-to-End Carbon Footprint of Large Language Models. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=aIok3ZD9to

[3] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997* (2024). https://doi.org/10.48550/arXiv.2312.10997

[4] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024). https://doi.org/10.48550/arXiv.2407.21783

[5] 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 *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. https://doi.org/10.1109/HPCA51647.2021.00076

[6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=nZeVKeeFYf9

[7] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

[8] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. https://doi.org/10.18653/v1/P17-1147

[9] Dorota Kopczyk and Abhishek Chandra. 2025. Re-Evaluating Storage Carbon Emissions in Machine Learning Workloads. In *4th Workshop on Sustainable Computer Systems (HotCarbon)*. https://hotcarbon.org/assets/2025/paper-34.pdf

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. 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 (SOSP)*. https://doi.org/10.1145/3600006.3613165

[11] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the Carbon Emissions of Machine Learning. *arXiv preprint arXiv:1910.09700* (2019). https://doi.org/10.48550/arXiv.1910.09700

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[13] Yueying Li, Omer Graif, and Udit Gupta. 2024. Towards Carbon-Efficient LLM Life Cycle. In *3rd Workshop on Sustainable Computer Systems (HotCarbon)*. https://hotcarbon.org/assets/2024/pdf/hotcarbon24-final154.pdf

[14] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2023. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *Journal of Machine Learning Research* 24, 253 (2023), 1–15. https://www.jmlr.org/papers/v24/23-0069.html

[15] Sophia Nguyen, Beihao Zhou, Yi Ding, and Sihang Liu. 2024. Towards Sustainable Large Language Model Serving. In *3rd Workshop on Sustainable Computer Systems (HotCarbon)*. https://doi.org/10.1145/3727200.3727220

[16] Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.18653/v1/2024.emnlp-main.15

[17] 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* (2021). https://doi.org/10.48550/arXiv.2104.10350

[18] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.18653/v1/D16-1264

[19] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.18653/v1/D19-1410

[20] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. In *IEEE High Performance Extreme Computing Conference (HPEC)*. https://doi.org/10.1109/HPEC58863.2023.10363447

[21] S. Schlömer, T. Bruckner, L. Fulton, E. Hertwich, A. McKinnon, D. Perczyk, J. Roy, R. Schaeffer, R. Sims, P. Smith, and R. Wiser. 2014. Annex III: Technology-specific Cost and Performance Parameters. In *Climate Change 2014: Mitigation of Climate Change. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 1329–1356. https://www.ipcc.ch/site/assets/uploads/2018/02/ipcc_wg3_ar5_annex-iii.pdf Table A.III.2, lifecycle GHG emissions of electricity supply technologies (gCO_2eq/kWh, medians).

[22] Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge. In *SIGIR-AP*. https://doi.org/10.1145/3673791.3698415

[23] U.S. Environmental Protection Agency. 2024. Emissions & Generation Resource Integrated Database (eGRID), 2022 Data. https://www.epa.gov/egrid. Summary Tables, released January 2024; state and subregion total-output CO_2e rates.

[24] Grant Wilkins, Srinivasan Keshav, and Richard Mortier. 2024. Offline Energy-Optimal LLM Serving: Workload-Based Energy Models for LLM Inference on Heterogeneous Systems. In *3rd Workshop on Sustainable Computer Systems (HotCarbon)*. https://doi.org/10.1145/3727200.3727217

[25] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, et al. 2022. Sustainable AI: Environmental Implications, Challenges and Opportunities. In *Proceedings of Machine Learning and Systems (MLSys)*. https://proceedings.mlsys.org/paper_files/paper/2022/hash/462211f67c7d858f663355eff93b745e-Abstract.html

[26] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An Open-Source Small Language Model. *arXiv preprint arXiv:2401.02385* (2024). https://doi.org/10.48550/arXiv.2401.02385

[27] Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. RAFT: Adapting Language Model to Domain Specific RAG. In *Conference on Language Modeling (COLM)*. https://openreview.net/forum?id=rzQGHXNReU