Investigating Carbon Footprint of Recommender Systems Beyond Training Time

Josef Schodl
Johannes Kepler University Linz
Linz, Austria
josef.schodl@gmail.com

Oleg Lesota
Johannes Kepler University Linz
Linz, Austria
oleg.lesota@jku.at

Antonela Tommasel
ISISTAN
CONICET-UNCPBA
Tandil, Argentina
Johannes Kepler University Linz
Linz, Austria
antonela.tommasel@isistan.unicen.edu.ar

Markus Schedl
Johannes Kepler University Linz
Linz, Austria
markus.schedl@jku.at

Abstract

The environmental footprint of recommender systems has received growing attention in the research community. While recent work has examined the trade-off between model accuracy and the estimated carbon emissions during training, we argue that a comprehensive evaluation should also account for the emissions produced during inference time, especially in applications where models are deployed for extended periods with frequent inference cycles. In this study, we extend previous carbon footprint analyses from the literature by incorporating the inference phase into the carbon footprint assessment and exploring how variations in training configurations affect emissions. Our findings reveal that models with higher training emissions can, in some cases, offer lower environmental costs at inference time. Moreover, we show that minimizing the number of validation metrics computed during training can lead to significant reductions in overall carbon footprint, highlighting the importance of thoughtful experimental design in sustainable machine learning.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Hardware** → **Impact on the environment**.

Keywords

Sustainability, Carbon footprint, Energy efficiency, Green AI

ACM Reference Format:
Josef Schodl, Oleg Lesota, Antonela Tommasel, and Markus Schedl. 2025. Investigating Carbon Footprint of Recommender Systems Beyond Training Time. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25), September 22–26, 2025, Prague, Czech Republic.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3705328.3759324

RecSys '25, Prague, Czech Republic

ACM ISBN 979-8-4007-1364-4/25/09
https://doi.org/10.1145/3705328.3759324

1 Introduction

Artificial Intelligence research has seen remarkable breakthroughs in recent years which have come at a growing environmental cost. Between 2012 and 2018, the computational resources required for state-of-the-art training runs increased by over 300,000x [2], while training a single LLM can emit hundreds of kilograms of CO_2 [22]. Power-intensive generative AI training/inference is growing much faster than other applications, and global data center electricity consumption could nearly double by 2030. Although this still represents a modest fraction of overall energy use, the growth trajectory is concerning, especially as large-scale AI deployments become increasingly common.

These figures sparked the *Green AI* initiative [18], which argues that energy and carbon costs should be reported alongside model accuracy. Accordingly, several tools and frameworks now support this goal by tracking emissions for generic deep learning workflows [10, 13]. Despite this progress, Recommender Systems (RS), which power the personalization engines of most modern media and commerce platforms, have received comparatively less attention in terms of sustainability. This is surprising given their wide deployment and increasingly complex architectures. Also, given that RS are commonly retrained at daily and weekly intervals, their environmental footprint can accumulate rapidly over time, magnifying even small inefficiencies in model or infrastructure choices.

A key step forward addressing this gap was the study by Spillo et al. [20], which evaluated 18 RS across three datasets. Their work revealed that complex deep and graph-based models often provide only marginal accuracy improvements while significantly increasing carbon emissions. However, their analysis focused only on the training phase, neglecting the energy cost of routine validation and model inference, both critical components in real-world deployment. In a follow-up study, Spillo et al. [19] focused on the impact on data reduction on training, while Vente et al. [23] evaluated the training cost of 16 RS used in 2013-2023 RecSys papers.

In this work, we revisit and extend Spillo et al. [20]'s study with the goal of *broadening the environmental lens through which RS are evaluated*. We do so by addressing the following research questions:

RQ1: Training setup. *How do variations in the training setup affect the reproducibility of performance and energy footprint measurements for RS training?*

RQ2: Inference. *Do models that incur higher energy footprints during training also exhibit higher footprints at inference time?*

By broadening the scope of environmental benchmarking to include both training and inference phases, we aim to provide a more holistic and reproducible understanding of RS sustainability. Specifically, we make the following contributions:

• **Validation overhead.** We quantify the (largely invisible) energy cost associated with computing validation metrics during training.
• **Life-cycle perspective.** We evaluate the inference cost, often overlooked despite running millions of times per deployment.
• **Reproducibility conditions.** We replicate previous experiments in the literature [20] under a different setup, showing how observations shift even with modest changes to the execution environment.

Our findings emphasize that responsible design must look beyond training accuracy and cost alone. We hope this study supports sustainable RS research and offers actionable guidance for researchers and practitioners working towards low-carbon RS.

2 Experimental setup

To address our RQs, we compare a range of RS in terms of energy consumption during training and inference. We first reproduce the experiments from Spillo et al. [20] to establish a reference point and then extend them with the investigation of the footprint caused by the inference of the same models.

Energy-centric measurement approach

Estimating the environmental impact of RS requires careful tracking of energy consumption. Prior work [19, 20] estimated emissions by using `CodeCarbon` [4] to monitor energy usage, then converted this energy (in kWh) into CO_2 emissions by multiplying it by a fixed global Carbon Intensity (CI) of 475 gCO_2/kWh, based on the 2019 *International Energy Agency Global Energy and* CO_2 *Status Report*[1]. While this simplifies standardization, it treats all computational activity as equally carbon-intensive, regardless of location, energy source or infrastructure, and may distort results when CI varies significantly across locations or time. Moreover, using a fixed CI introduces only a constant scaling factor, which offers limited analytical value and potentially leading to misleading conclusions.

In contrast, our study emphasizes *direct energy tracking*. We also use `CodeCarbon`, but instead of immediately converting energy into CO_2 emissions using a static CI, we focus on reporting the actual *electrical energy consumption* (in kWh) as measured by the hardware. `CodeCarbon` operates by periodically sampling system-level power sensors[2] (every 15 seconds by default), accumulating the energy used by the CPU, GPU and RAM throughout each run. This provides a granular and hardware-specific view of energy consumption.

By reporting raw energy rather than emissions, our methodology provides a clean and interpretable value, supporting *greater flexibility and transparency*. Researchers can later apply localized CI values or account for infrastructure efficiency factors like Power Usage Effectiveness[3] (PUE) to estimate emissions more precisely, without needing to rerun experiments.

Recommendation algorithms

We evaluate a set of 18 well-known RS also used in [20]. These RS span five categories, capturing a range of modeling complexities and approaches: basic (Pop [21], ItemKNN [5]), matrix factorization and linear models (BPR [16, 17], DMF [31], FISM [12], NAIS [8], SLIM [15]), deep learning (NeuMF [9], MultiDAE [14], NGCF [26], DGCF [27]), graph-based (LightGCN [7], SGL [28]) and knowledge-aware models (CKE [32], CFKG [1], KGCN [25], KTUP [3], RippleNet [24]).

Datasets

We use three widely used datasets, also used in [20], to ensure comparability of results: *Amazon Books*, *MovieLens 1M* and *Mind*. As the original study applied some undocumented preprocessing steps we used the preprocessed versions from their repository[4].

Hardware, operating system and packages

Hardware. Spillo et al. [20] conducted their experiments on a 2012-era desktop workstation with a 3rd-gen Intel Core i7-3770 CPU (Ivy Bridge, 4 cores) and an NVIDIA GeForce GTX Titan X GPU (Maxwell architecture). While Linux servers dominate industrial pipelines, a significant portion of research, teaching, and early-stage prototyping occurs on commodity laptops, especially in resource-constrained settings. Then, to broaden the relevance of our findings beyond desktop setups, our experiments were carried out on a laptop with a 10th-gen 2020 Intel Core i7-10850H CPU (Comet Lake-H, 6 cores) and an NVIDIA Quadro T2000 GPU (Turing architecture). While our CPU is newer and has more cores, it is also a mobile processor with a focus on energy efficiency. This shift also enables us to assess how energy rankings and absolute footprints generalize across hardware profiles with different CPU/GPU efficiency ratios, thus contributing to a more comprehensive understanding of RS sustainability under diverse operational conditions. The Titan X GPU is more powerful in raw compute (3072 CUDA cores, ~6.6 TFLOPS FP32) compared to the Quadro T2000 (1024 CUDA cores, ~3.7 TFLOPS FP32), but also more power hungry.

Operating system. While the original experiments ran on Ubuntu 20.04 with Linux kernel 5.15.0, to expand the range of tested configurations, ours were conducted on Windows 11 24H2. This choice reflects real-world diversity in research and prototyping environments, as supported in surveys[5] and previous works [6, 11].

Python and packages. We attempted to replicate the original Python environment by Spillo et al. [20], but encountered compatibility issues preventing a stable setup. Instead, we ensured consistency by using the same `RecBole` [33] version (1.1.1) [30] and building a working package configuration around it. The original code ran on Python 3.7.16 with PyTorch 1.13.1. Our experiments used Python 3.8.20 and PyTorch 2.3.1 (including optimizations that can influence performance and energy usage [29]). Although Python

[1] https://www.iea.org/reports/global-energy-co2-status-report-2019/emissions
[2] https://mlco2.github.io/codecarbon/methodology.html
[3] https://www.thegreengrid.org/en/resources/library-and-tools/20-PUE%3A-A-Comprehensive-Examination-of-the-Metric
[4] https://github.com/swapUniba/CarbonRecommenderRecSys23
[5] Stack Overflow 2024 Survey: https://survey.stackoverflow.co/2024/technology#1-operating-system

3.8 is no longer maintained, we selected it to preserve broad compatibility with the original environment and `RecBole`'s dependencies at the time of the original study and avoid introducing additional variabilities from major language version changes. This allowed for a faithful yet functional reproduction of the original setup.

Model training methodology

For all experiments, as Spillo et al. [20], we followed the standard `RecBole` implementations and kept most hyper-parameters at their default values. However, a few controlled adjustments were made to guarantee a consistent and reproducible experimental setup. First, we enabled `RecBole`'s `reproducibility` mode, which ensures the training, validation and test splits remain consistent across runs. We also standardized the number of training epochs [20], setting it to 50 for MovieLens and Mind, and to 20 for Amazon Books. `RecBole` adopts a commonly-used default early stopping configuration (as used in [20]); however, since this can lead to models being trained for different numbers of epochs, we chose to disable early stopping and train all models for the full epoch number to maintain uniformity across experiments. Model evaluation was performed after training each epoch. To assess the impact of evaluation metric computation during epochs, we considered two configurations. First, *Reproduction*, in which we computed the full set of 12 ranking-oriented metrics reported in [20] (AveragePopularity, GAUC, GiniIndex, Hit, ItemCoverage, MAP, MRR, NDCG, Precision, Recall, ShannonEntropy, TailPercentage), extending `RecBole`'s default set of 5 metrics. Second, *One-Metric*, in which we limited evaluation during training to a single metric, MRR (`RecBole`'s default early stopping metric), to reduce computational overhead.

Model inference methodology

Inference experiments used the same `RecBole` implementations as in training. For comparability with training energy results, we generated top-10 recommendations for all valid users in each dataset in a single batch using `RecBole`'s built-in inference function. Short inference calls (<1 s) showed high variance in `CodeCarbon` readings due to initialization overhead and architectural effects. To address this, each run began with a warm-up inference, followed by 10 identical inference queries whose energy was measured. This procedure was repeated 10 times per configuration, and we report the mean and standard deviation across runs.

3 Experimental results

RQ1: Training footprint

Table 1[6] compares the energy usage reported by Spillo et al. [20] (*Reference Work*) with our *Reproduction* and *One-Metric* experiments. Due to GPU memory constraints, we omit FISM and NAIS on the Amazon Books dataset[7].

Relative efficiency ranking is largely stable. Across all three datasets, simpler models such as Pop, ItemKNN and SLIM consistently remain among the most efficient, while more complex models like DGCF and SGL continue to rank highest in energy use. The Spearman's rank correlation coefficient between *Reference Work* and our *Reproduction* for nDCG ranged between $\rho = 0.659$ (Mind) and $\rho = 0.876$ (MovieLens) for nDCG showing moderate to strong alignment[8]. On the other hand, the correlation for energy consumption ranged between and $\rho = 0.724$ (Amazon Books) and $\rho = 0.973$ (MovieLens), supporting the view that relative energy cost is primarily driven by algorithm complexity and is largely independent of hardware/software differences.

Absolute energy use decreases in most, but not all cases. Our mobile hardware setup reduces training energy consumption in 31 of the 47 model/dataset combinations compared to the original desktop environment. However, the scale and direction of the shift vary considerably by dataset. For instance, SLIM sees an 85% energy drop on Amazon, but a 198% increase on MovieLens; DGCF drops 23% on MovieLens but rises 159% on Amazon and 270% on Mind; MultiDAE shows minimal change on Mind (+2%), yet drops by 60% on Amazon and increases by 822% on MovieLens. These inconsistencies highlight the complex interaction between model architecture, dataset characteristics (e.g. size, sparsity), and hardware/software efficiency profiles. They reinforce the point that absolute energy figures are not easily generalizable across environments.

Validation metric computation carries notable overhead. Replacing the original 12 validation metrics per-epoch with a single metric in the *One-Metric* configuration yields substantial energy savings, ranging from 1.03% to 71.71%, with an average reduction of 28.77%. Lightweight models benefit the most, as metric evaluation constitutes a large portion of their total training cost. Even deep graph-based models such as DGCF and SGL show measurable savings, indicating that validation bookkeeping is a meaningful component of overall energy usage. Importantly, reducing the number of validation metrics to a single one does not cause any changes in the models' accuracy, as in both cases the same one metric (MRR) is used as a stopping criterion. This way careful selection of validation metrics can lead to energy savings without any loss of performance.

We observe that quantitative energy values and model rankings are sensitive to changes in hardware, dataset, and evaluation setup. While newer hardware is expected to be more efficient, our controlled study shows that 16 of 47 model/dataset pairs actually consume more energy. These discrepancies are not random, memory-bound graph models often can underutilize modern GPUs, while matrix-factorization models benefit more from hardware upgrades. This non-uniform pattern underscores that energy profiles must be validated per model as scaling assumptions can be misleading.

RQ2: Inference footprint

Table 1 reports the energy to serve 10 recommendation queries for each model trained in *Reproduction*. We omit CFKG on Amazon Books, KGCN, KTUP and RippleNet on MovieLens and Amazon Books, and NeuMF on all datasets due to GPU RAM constraints[9].

[6] All code is available at the companion repository https://github.com/hcai-mms/RecSys-Footprint-Inference. To ease reproducibility, a Docker image is also provided. The repository also includes additional accuracy results, rank plots and heatmaps.
[7] NAIS on Amazon Books was also not reported by Spillo et al. [20], due to presumably performance or stability problems.
[8] Given that we used the same dataset and models than Spillo et al. [20], these could be related to some differences in the under-the-hood configuration.
[9] We acknowledge that this issue could have been solved by customizing per-model GPU settings, but we preferred to prioritize the comparability of results under `RecBole` standard configurations.

Table 1: Energy consumed during training and inference of different models with the corresponding rankings per dataset (largest absolute differences highlighted).

		Reference Work		RQ1: Reproduction (vs Reference)				RQ1: One-Metric (vs Reproduction)				RQ2: Inference (10 Queries)				
	Algorithm	Energy kWh·10^{-4}	Rank	Energy kWh·10^{-4}	Rank	Energy %	Rank Δ	Energy kWh·10^{-4}	Rank	Energy %	Rank Δ	Energy kWh·10^{-4}	Std kWh·10^{-6}	Rank Infer.	Rank Train	Rank Δ
Mind Dataset	Pop	50.93	1	8.46	1	-83.39	0	3.25	1	-61.52	0	2.41	12.91	5	1	-4
	ItemKNN	78.68	2	12.34	2	-84.32	0	8.71	2	-29.36	0	20.24	13.04	10	2	-8
	BPR	850.86	8	358.48	3	-57.87	-5	153.89	3	-57.07	0	2.34	11.44	4	3	-1
	DMF	820.09	7	662.91	6	-19.17	-1	464.00	7	-30.01	+1	2.48	5.85	7	6	-1
	FISM	600.63	6	853.88	9	+42.17	+3	669.85	9	-21.55	0	49.59	35.99	11	8	-3
	NAIS	1,293.77	10	2,577.22	11	+99.20	+1	2,234.17	12	-13.31	+1	362.76	80.08	12	10	-2
	SLIM	125.29	3	415.55	4	+231.68	+1	220.82	4	-46.86	0	297.09	7.02	8	4	-4
	NeuMF	498.04	4	673.80	7	+35.29	+3	418.04	6	-37.96	-1	-	-	-	-	-
	MultiDAE	514.13	5	525.79	5	+2.27	0	279.84	5	-46.78	0	3.10	3.79	9	5	-4
	NGCF	1,586.28	12	1,141.15	10	-28.06	-2	915.52	10	-19.77	0	2.47	9.40	6	9	+3
	DGCF	6,298.43	13	23,313.07	13	+270.14	0	21,655.51	13	-7.11	0	2.27	2.17	1	12	+11
	LightGCN	949.24	9	841.29	8	-11.37	-1	608.56	8	-27.66	0	2.31	7.02	2	7	+5
	SGL	1,492.38	11	3,110.73	12	+108.44	+1	2,208.26	11	-29.01	-1	2.31	7.10	3	11	+8
MovieLens Dataset	Pop	6.28	1	2.21	1	-64.78	0	1.17	1	-46.97	0	0.58	1.89	6	1	-5
	ItemKNN	15.19	3	3.05	2	-79.93	-1	2.50	2	-18.02	0	3.88	5.84	11	2	-9
	BPR	30.19	4	95.51	3	+216.38	-1	42.54	3	-55.46	0	0.53	1.67	2	3	+1
	DMF	188.38	10	170.60	8	-9.44	-2	119.38	8	-30.02	0	0.60	1.02	8	7	-1
	FISM	1,768.94	14	574.19	14	-67.54	0	509.79	14	-11.22	0	34.67	20.41	13	11	-2
	NAIS	5,466.90	17	6,525.24	17	+19.36	0	5,390.70	17	-17.39	0	109.44	9.35	14	13	-1
	SLIM	43.03	5	128.10	5	+197.70	0	82.99	5	-35.21	0	0.96	2.65	10	5	-5
	NeuMF	77.38	6	146.15	7	+88.86	+1	118.80	7	-18.71	0	-	-	-	-	-
	MultiDAE	13.28	2	122.38	4	+821.50	+2	67.97	4	-44.46	0	0.69	2.81	9	4	-5
	NGCF	717.59	13	465.88	13	-35.08	0	422.19	13	-9.38	0	0.58	2.43	7	10	+3
	DGCF	10,255.42	18	7,936.47	18	-22.61	0	7,841.05	18	-1.20	0	0.54	2.41	5	14	+9
	LightGCN	443.28	12	327.65	12	-26.09	0	280.63	12	-14.35	0	0.53	1.54	1	9	+8
	SGL	1,890.25	15	1,112.91	15	-41.12	0	1,058.46	15	-4.89	0	0.53	2.54	4	12	+8
	CKE	204.92	11	202.73	9	-1.07	-2	151.96	9	-25.04	0	0.53	1.80	3	8	+5
	CFKG	83.67	7	134.90	6	+61.22	-1	84.03	6	-37.71	0	5.50	6.38	12	6	-6
	KGCN	155.20	9	228.97	10	+47.53	+1	173.47	10	-24.24	0	-	-	-	-	-
	KTUP	95.44	8	245.47	11	+157.19	+3	193.41	11	-21.21	0	-	-	-	-	-
	RippleNet	3,825.05	16	1,719.41	16	-55.05	0	1,701.75	16	-1.03	0	-	-	-	-	-
Amazon Books Dataset	Pop	69.80	1	12.58	1	-81.97	0	3.56	1	-71.71	0	14.41	10.43	9	1	-8
	ItemKNN	102.79	2	16.41	2	-84.04	0	9.73	2	-40.72	0	24.32	15.21	10	2	-8
	BPR	650.06	6	146.46	3	-77.47	-3	84.06	4	-42.61	+1	4.37	2.97	2	3	+1
	DMF	1,076.80	10	255.63	7	-76.26	-3	184.87	7	-27.68	0	12.44	7.15	8	6	-2
	SLIM	1,277.41	12	191.69	6	-84.99	-6	96.01	6	-49.91	0	5.85	4.05	7	5	-2
	NeuMF	2,345.50	15	529.30	13	-77.43	-2	432.42	13	-18.30	0	-	-	-	-	-
	MultiDAE	1,077.70	11	433.98	12	-59.73	+1	279.83	10	-35.52	-2	38.45	11.67	11	9	-2
	NGCF	1,572.66	13	294.23	8	-81.29	-5	222.57	9	-24.36	+1	4.89	6.92	6	7	+1
	DGCF	7,095.90	16	18,347.75	16	+158.57	0	17,758.41	16	-3.21	0	4.48	2.75	5	11	+6
	LightGCN	931.31	9	308.43	9	-66.88	0	221.70	8	-28.12	-1	4.42	4.98	3	8	+5
	SGL	2,086.83	14	1,049.14	15	-49.73	+1	933.52	15	-11.02	0	4.43	9.25	4	10	+6
	CKE	371.30	3	151.80	4	-59.12	+1	73.92	3	-51.31	-1	2.05	9.17	1	4	+3
	CFKG	444.36	4	173.38	5	-60.98	+1	94.36	5	-45.58	0	-	-	-	-	-
	KGCN	655.46	7	388.62	11	-40.71	+4	302.12	12	-22.26	+1	-	-	-	-	-
	KTUP	718.78	8	361.18	10	-49.75	+2	283.90	11	-21.40	+1	-	-	-	-	-
	RippleNet	582.30	5	559.73	14	-3.88	+9	480.06	14	-14.23	0	-	-	-	-	-

The Spearman's rank correlation coefficient between training and inference rankings was weak and not significant, ranging between $\rho = -0.210$ (Mind) and $\rho = 0.029$ (MovieLens). This suggests that models which are expensive to train are not necessarily cheap to run at inference time, nor the reverse. Figure 1 illustrates this finding for the Mind dataset, where higher training energy consumption is not always followed by higher energy expenses during inference (marker size). Individual break-even points can arise when complex models benefit from efficient batch inferencing on GPUs, while simpler models rely on slower, CPU-bound scoring.

To estimate the break-even point between models, let $E_{\text{train},i}$ and $E_{\text{infer},i}$ denote the training and *per-query* inference footprint of model i. When model A has a higher training cost but a lower inference cost than model B, the number of inference queries required to offset A's initial cost is depicted in Eq. 1, assuming $E_{\text{train},A} > E_{\text{train},B}$ and $E_{\text{infer},A} < E_{\text{infer},B}$ to ensure $N > 0$.

$$N = \left\lceil \frac{E_{\text{train},A} - E_{\text{train},B}}{E_{\text{infer},B} - E_{\text{infer},A}} \right\rceil \tag{1}$$

For example, on the Mind dataset, both BPR and ItemKNN achieve nDCG of 0.56, but BPR requires $358.48 \cdot 10^{-4}$ kWh to train, compared

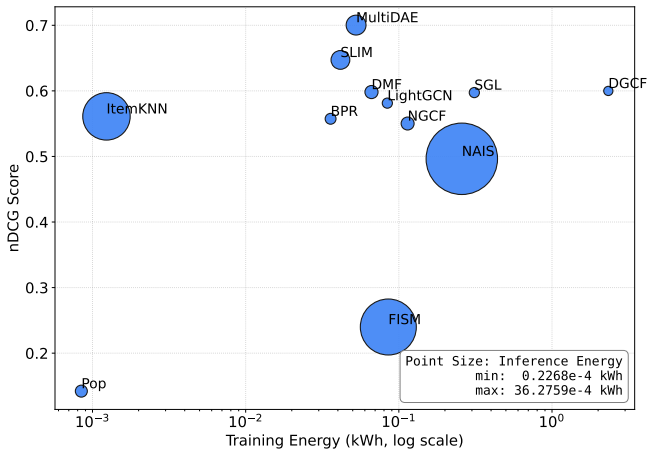

Figure 1: The relation of the training and inference time energy consumption to the resulting accuracy (Mind dataset).

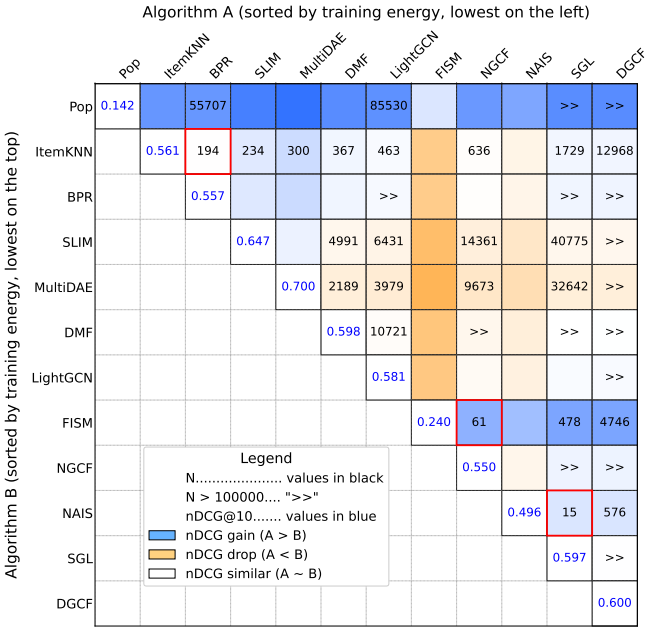

Figure 2: Break-even analysis on the Mind dataset. After 61 recommendation cycles to all users NGCF equals the energy consumption of FISM while providing better accuracy.

to $12.34 \cdot 10^{-4}$ kWh for ItemKNN, whereas BPR's per-query inference cost is $0.234 \cdot 10^{-4}$ kWh, while ItemKNN consumes $2.024 \cdot 10^{-4}$ kWh. Using Eq. 1, we observe that BPR becomes more energy-efficient, making it the greener choice for workloads over 194 queries per model update. Figure 2 summarizes break even analysis on the Mind dataset, highlighting in red three points where the fewest recommendation cycles is needed to reach equality in consumption between two models. Absence of a value means that the break-even point is unachievable for the two models[10].

Our results show that training and inference efficiencies are not strongly aligned. As a result, the greener model depends on workload. For low-traffic or frequently updated systems, simpler models retain their advantage. In high-traffic deployments, however, the initial training overhead of a complex model can quickly be amortized over many queries, reversing the energy preference. To optimize for sustainability, both training and inference phases must be considered jointly.

4 Conclusion and future work

Our study offers new insights into the energy efficiency of recommender systems, expanding prior work [20] with a more detailed analysis of lifecycle trade-offs and evaluation overhead. First, we showed that while energy rankings during training are broadly consistent across setups, absolute energy use is highly sensitive to the underlying infrastructure. Second, we quantify the substantial energy cost of per-epoch evaluation, a factor often overlooked in prior analyses, and show that logging only one metric can reduce training energy by ≈ 30% without affecting accuracy. Most notably, we introduce a break-even analysis that reveals how a more expensive-to-train model can become the greener option under frequent inference, shifting the energy balance over a model's life cycle. This framework allows practitioners to select the most efficient model based on expected query volume. Together, these findings support more granular, context-aware decisions and offer actionable, low-effort guidelines for MLOps engineers, capacity planners, and researchers aiming to build sustainable recommendation pipelines. Future work should address: (1) benchmarking training and inference energy on cloud GPUs with real-time carbon intensity and job-level PUE, (2) while `RecBole` is a widely used library endorsed by the RecSys community, extending the study to other recommendation frameworks remains valuable to ensure broader generalizability, (3) exploring whether the generated energy trace datasets could be used as input for predictive tasks, for example, forecasting training or inference energy consumption based on model, data and hardware characteristics, (4) designing inference footprint evaluation procedures better reflecting real world scenarios, (5) the impact of software-level factors (such as newer Python versions or updated library releases) on energy efficiency. Building on the foundation of the work at hand, these steps will foster shifting carbon-aware recommender research from isolated measurements toward reproducible and comparable life-cycle assessments.

Acknowledgments

This research was funded in whole or in part by the Austrian Science Fund (FWF): https://doi.org/10.55776/COE12. This work received financial support from the State of Upper Austria and the Federal Ministry of Education, Science, and Research, through grant LIT-2024-13-SEE-111.

[10] More analyses and break-even plots can be found in the companion repository.

References

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137. doi:10.3390/A11090137

[2] Dario Amodei, Danny Hernandez, Girish Sastry, Jack Clark, Greg Brockman, and Ilya Sutskever. 2018. AI and Compute. (2018).

[3] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 151–161. doi:10.1145/3308558.3313705

[4] Benoit Courty, Victor Schmidt, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, SabAmine, inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Amine Saboni, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, alencon, Michał Stęchły, Christian Bauer, Lucas-Otavio, JPW, and MinervaBooks. 2024. *mlco2/codecarbon: v2.4.1*. doi:10.5281/zenodo.11171501

[5] Mukund Deshpande and George Karypis. 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Inf. Syst.* 22, 1 (2004), 143–177. doi:10.1145/963770.963776

[6] Kaushik Dutta and Debra Vandermeer. 2023. A case for sustainability and environment friendliness in software development and architecture decisions by taking energy-efficient design decisions. *arXiv preprint arXiv:2311.01680* (2023).

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 639–648. doi:10.1145/3397271.3401063

[8] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Trans. Knowl. Data Eng.* 30, 12 (2018), 2354–2366. doi:10.1109/TKDE.2018.2831682

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 173–182. doi:10.1145/3038912.3052569

[10] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *CoRR* abs/2002.05651 (2020). arXiv:2002.05651

[11] Congfeng Jiang, Dongyang Ou, Yumei Wang, Xindong You, Jilin Zhang, Jian Wan, Bing Luo, and Weisong Shi. 2016. Energy efficiency comparison of hypervisors. In *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*. 1–8. doi:10.1109/IGCC.2016.7892607

[12] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. ACM, 659–667. doi:10.1145/2487575.2487589

[13] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the Carbon Emissions of Machine Learning. *CoRR* abs/1910.09700 (2019). arXiv:1910.09700

[14] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 689–698. doi:10.1145/3178876.3186150

[15] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, Diane J. Cook, Jian Pei, Wei Wang, Osmar R. Zaïane, and Xindong Wu (Eds.). IEEE Computer Society, 497–506. doi:10.1109/ICDM.2011.134

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, Jeff A. Bilmes and Andrew Y. Ng (Eds.). AUAI Press, 452–461.

[17] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. *CoRR* abs/1205.2618 (2012). arXiv:1205.2618

[18] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM* 63, 12 (2020), 54–63. doi:10.1145/3381831

[19] Giuseppe Spillo, Allegra De Filippo, Cataldo Musto, Michela Milano, and Giovanni Semeraro. 2024. Towards Green Recommender Systems: Investigating the Impact of Data Reduction on Carbon Footprint and Algorithm Performances. In *Proceedings of the 18th ACM Conference on Recommender Systems* (Bari, Italy) *(RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 866–871. doi:10.1145/3640457.3688160

[20] Giuseppe Spillo, Allegra De Filippo, Cataldo Musto, Michela Milano, and Giovanni Semeraro. 2023. Towards Sustainability-aware Recommender Systems: Analyzing the Trade-off Between Algorithms Performance and Carbon Footprint. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*, Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song (Eds.). ACM, 856–862. doi:10.1145/3604915.3608840

[21] Keshetti Srikala. 2020. Popularity Based Recommendation System. *International Journal of Engineering and Advanced Technology* 9 (02 2020), 1561–1566. doi:10.35940/ijeat.B4660.029320

[22] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 3645–3650. doi:10.18653/V1/P19-1355

[23] Tobias Vente, Lukas Wegmeth, Alan Said, and Joeran Beel. 2024. From Clicks to Carbon: The Environmental Toll of Recommender Systems. In *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024, Bari, Italy, October 14-18, 2024*, Tommaso Di Noia, Pasquale Lops, Thorsten Joachims, Katrien Verbert, Pablo Castells, Zhenhua Dong, and Ben London (Eds.). ACM, 580–590. doi:10.1145/3640457.3688074

[24] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang (Eds.). ACM, 417–426. doi:10.1145/3269206.3271739

[25] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 3307–3313. doi:10.1145/3308558.3313417

[26] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France*. ACM, 165–174. doi:10.1145/3331184.3331267

[27] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1001–1010. doi:10.1145/3397271.3401137

[28] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 726–735. doi:10.1145/3404835.3462862

[29] Peng Wu. 2023. PyTorch 2.0: The Journey to Bringing Compiler Technologies to the Core of PyTorch (Keynote). In *Proceedings of the 21st ACM/IEEE International Symposium on Code Generation and Optimization, CGO 2023, Montréal, QC, Canada, 25 February 2023- 1 March 2023*, Christophe Dubach, Derek Bruening, and Ben Hardekopf (Eds.). ACM, 1. doi:10.1145/3579990.3583093

[30] Lanling Xu, Zhen Tian, Gaowei Zhang, Lei Wang, Junjie Zhang, Bowen Zheng, Yifan Li, Yupeng Hou, Xingyu Pan, Yushuo Chen, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. 2022. Recent Advances in RecBole: Extensions with more Practical Considerations.

[31] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 3203–3209. doi:10.24963/IJCAI.2017/447

[32] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 353–362. doi:10.1145/2939672.2939673

[33] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *CIKM*. ACM, 4653–4664.