ISSN: 1692-7257 - Volume 2 – Issue 48 - 2026

ORIGINAL

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Exploring the environmental effect of multi-GPU scaling on the energy-time-CO_2e relationship during the training of vision deep learning models

Explorando el efecto ambiental del escalamiento multi-GPU en la relación energía-tiempo-CO_2e durante el entrenamiento de modelos de aprendizaje profundo para visión

Carlos Nelson Henríquez Miranda[1]
Malak Andrés Sánchez Cataño[1]
German Sánchez Torres[1]

[1] **Universidad del Magdalena, Grupo de Investigación y Desarrollo en Sistemas y Computación, Santa Marta, Magdalena, Colombia.**

Received: 24 March 2026. Revised: 12 June 2026. Accepted: 10 July 2026. Published: 15 July 2026.
Corresponding Author: Carlos Nelson Henríquez Miranda

How to cite: C. N. Henríquez Miranda, M. A. Sánchez Cataño, and G. Sánchez Torres, "Explorando el efecto ambiental del escalamiento multi-GPU en la relación energía-tiempo-CO_2e durante el entrenamiento de modelos de aprendizaje profundo para visión", *Revista Colombiana de Tecnologías de Avanzada (RCTA)*, vol. 2, no. 48, jul. 2026, art. no. 4569, doi: 10.24054/rcta.v2i48.4569

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Abstract:

GPU scaling is a widely used strategy to reduce the training time of deep vision models. However, a reduction in time does not necessarily imply a reduction in energy consumption. This work investigates the impact of GPU scaling on the energy–time–CO_2 relationship during the training of a deep vision model (dataset: CIFAR-10, architecture: ResNet-50) under a fixed compute budget to minimize variability. Energy was measured directly at the component level by combining Intel RAPL counters for the CPU and accumulated GPU energy via NVIDIA NVML, and CO_2e was estimated with CodeCarbon using a carbon intensity of 0.20 $kgCO_2e/kWh$ (sensitivity: 0.10–0.30). Twelve configurations were evaluated by combining scaling (from 1 to 2 GPUs), precision (FP32 vs. AMP), and per-GPU power limits (300/250/200 W). The results show that multi-GPU scaling does not always reduce total energy, and that system-level adjustments such as AMP and power capping can more consistently improve energy efficiency, identifying an operating region near 250 W per GPU that reduces energy by 4.6%–9.6% with a time penalty of 0.7%–1.9%. In addition, CodeCarbon estimates reproduce the trends but exhibit a median relative error of 4.09% compared to direct measurements. These findings suggest practical criteria for selecting training configurations that balance acceleration and energy sustainability in deep vision.

Keywords: energy-aware computing; green computing; energy-efficient deep learning; carbon footprint.

Resumen:

El escalamiento de GPU es una estrategia ampliamente usada para reducir el tiempo de entrenamiento de modelos profundos de visión. Sin embargo, una reducción del tiempo no implica necesariamente una reducción del consumo energético. Este trabajo explora el impacto del escalamiento de GPU en la relación energía–tiempo-CO_2e durante el entrenamiento de un modelo profundo de visión (dataset: CIFAR-10, arquitectura: ResNet-50) bajo un presupuesto fijo de cómputo para minimizar la variabilidad. La energía se midió directamente a nivel de componentes combinando contadores Intel RAPL para CPU y energía acumulada por GPU vía NVIDIA NVML, y se estimó el CO_2e con CodeCarbon usando una intensidad de carbono de 0.20 $kgCO_2e/kWh$ (sensibilidad: 0.10–0.30). Se evaluaron 12 configuraciones que combinan escalamiento (de 1 a 2 GPU), precisión (FP32 vs AMP) y límites de potencia por GPU (300/250/200 W). Los resultados muestran que el escalamiento multi-GPU no siempre reduce la energía total, y que ajustes de sistema como AMP y power capping pueden mejorar de forma más consistente la eficiencia energética, identificando una región operativa cercana a 250 W por GPU que reduce energía en 4.6%–9.6% con una penalización temporal de 0.7%–1.9%. Además, las estimaciones con CodeCarbon reproducen las tendencias, pero presentan un error relativo mediano de 4.09% frente a las mediciones directas. Los resultados sugieren criterios prácticos para seleccionar configuraciones de entrenamiento que equilibren aceleración y sostenibilidad energética en visión profunda.

Palabras clave: computación consciente del consumo energético; computación verde; aprendizaje profundo eficiente en energía; huella de carbono.

INTRODUCTION

Training deep vision models has become a central component of many scientific and industrial applications[1–3], enabled by the widespread adoption of Graphics Processing Unit (GPU) accelerators and hybrid Central Processing Unit (CPU) + GPU architectures in both workstations and High-Performance Computing (HPC) clusters[4]. Evidence on historical trends shows that training compute has grown rapidly [5] in the Deep Learning era[6],[7], with doubling times on the order of months and a further shift toward large-scale models[5]. This raises environmental concerns because of its high energy consumption[8]. In compute-intensive domains such as medical imaging, for example, tasks on magnetic resonance imaging (MRI) with Deep Learning[9], the pressure to reduce training times and improve performance is often addressed by increasing parallelism and resources, with direct effects on energy consumption, operating costs, cooling demand, and associated emissions. At HPC scale, leading systems report high-magnitude aggregate consumption, which shows the need to evaluate not only execution time but also the total energy required by different training configurations[4].

At the same time, deep learning tends to require substantial increases in compute for marginal performance gains, which makes the energy cost of those gains relevant and requires distinguishing between acceleration and energy efficiency [10]. In this context, GPU scaling (moving from 1 to multiple GPUs) is often justified by reduced training time; however, total energy (kWh) and its translation into emissions (^{kgCO_2e}) may not decrease proportionally. Communication and synchronization overheads, variations in utilization, and system base consumption can erode (or even reverse) the expected energy benefits, especially when the analysis is carried out end to end and includes CPU+GPU under comparable assumptions.

Recent literature offers relevant but fragmented contributions for understanding this energy–time trade-off. On the one hand, GPU power control through power capping and its effect on metrics such as Energy–Delay Product (EDP) or Energy–Delay Sum (EDS) have been studied, with consumption reductions that depend on the compute regime and on the optimization criterion[4]. On the other hand, tools and approaches for tracking or estimating energy and carbon footprint during training have become widespread, allowing hardware and scenarios to be compared by means of regional emission factors[10–12]. However, it has also been noted that the main limitation to comparability and traceability across studies is the heterogeneity of assumptions[13], that is, what is measured, with which instrument, and how it is converted to emissions. In parallel, HPC research has explored models to predict the power of jobs before their execution, underscoring that the observed consumption depends both on the application and on the operational context[14]. Complementarily, notions such as energy bloat and time–energy frontiers help separate consumption that contributes to throughput from consumption associated with waits, synchronization or other inefficiencies that can distort conclusions if only time is considered[15].

Despite this, a practical weakness persists: there is a lack of directly comparable evidence characterizing how scaling from single-GPU to multi-GPU modifies the energy–time relationship when common efficiency practices are combined with explicit power capping control mechanisms, under strict comparison conditions. In particular, many reports do not control the effective load with a fixed compute budget nor precisely delimit the scope of the measurement, amplifying methodological heterogeneity[13].

This work explores that weakness through a controlled evaluation of the training of a deep vision model (dataset: CIFAR-10, architecture: ResNet-50) under a fixed compute budget. It studies the effect of scaling (from 1 to 2 GPUs), precision (FP32 vs. AMP) and per-GPU power limits (300/250/200 W) on the energy–time trade-off. Energy is measured at the component level by combining Intel RAPL counters for CPU and accumulated GPU energy via NVIDIA NVML, and CO_2e is estimated with CodeCarbon [16] using a carbon intensity of 0.20 ($kgCO_2e$)/kWh (sensitivity: 0.10–0.30)[10–12]. This work thus attempts to provide (i) reproducible evidence on when scaling accelerates without necessarily saving energy, (ii) practical criteria for selecting efficient configurations with AMP and power capping [4] and (iii) a consistent reading of the energy–performance trade-off supported by the time–energy frontier framework and by the recognition of end-to-end inefficiencies[15]. In addition, implications for operation and planning in CPU+GPU environments are discussed, considering the role of the operational context in the observed consumption[14]. The document is organized as follows: first, previous work and the literature related to the trade-off among energy, time and emissions in training processes are reviewed. Then, materials and methods are presented, including the experimental methodology to analyze scaling from single-GPU to multi-GPU and its impact on energy consumption and training time. Finally, the results section reports the evaluations of each comparison, together with the discussion of the findings, and the conclusions are formulated.

RELATED WORK

To structure the evidence on the energy–time and emissions trade-off in training, the literature has been grouped into two groups: (i) optimization or control techniques that modify power, frequency, communications or resource allocation to improve efficiency; (ii) measurement or estimation and reporting methods that quantify E–T–CO_2e with instrumentation or models.

A practical way to reduce the environmental impact of training in relation to the energy consumed and the associated CO_2e

is not to assume that faster implies cleaner. In practice, total energy depends on how long training lasts (T) and on how much power the hardware demands while working (P); therefore, many strategies seek to find configurations that reduce total energy (E) or that achieve better trade-offs between energy and time. For consistent comparisons, composite metrics are used, such as the energy–delay product ($EDP = E \cdot T$) and the energy–delay sum ($EDS = E + T$).

E–T–CO_2 optimization and control

A first line of work adjusts how much power the GPU can consume (power capping) or the frequency at which it operates (through Dynamic Voltage and Frequency Scaling, Dynamic Voltage and Frequency Scaling, DVFS). Measurements based on the NVIDIA Management Library (NVML) report energy savings of up to 30% with performance penalties below 12%; measurement quality is also assessed by comparing it with other interfaces such as the Performance Application Programming Interface (PAPI), using error metrics (MAPE 6.39% and RMSE 3.97%) [4]. Complementarily, adjusting the core frequency (f_{core}) with DVFS, guided by how performance changes relative to power, saves $8,7\% - 23,1\%$ of energy in deep neural network training (and $19,6\% - 26,4\%$ in inference), and even reports performance improvements of up to 33% over default values [17]. Following the same real-time control logic, Dynamic Energy-Performance Optimiser (DEPO) adjusts power limits online using NVML and the CUDA Profiling Tools Interface (CUPTI) to optimize an objective of the form $\min(\alpha E + \beta T)$; it reports reductions >22% in energy (E) with drops <20% when energy is prioritized, and savings of 15%–18% with <8% penalty when the trade-off (EDP/EDS) is prioritized [4].

A second line of work focuses on the extra consumption that appears when training with multiple GPUs and that does not always contribute to learning: waits, synchronization and communication. This can increase total energy, even if training finishes earlier. Perseus [15] addresses this energy bloat phenomenon by constructing a time–energy Pareto front of the job, and reports savings of up to 30% in GPT-3 and Bloom without loss of effective performance (throughput). PowerFlow [18] proposes a simple model where consumption depends on the number of GPUs, the local minibatch size and the frequency; it then allocates resources to maximize energy efficiency without exceeding a budget, improving completion time between 1.57× and 3.39× compared with baselines at the same consumption[18]. There is also evidence that increasing the minibatch can improve performance per hour of energy: a representative result shows ResNet-50 on ImageNet with minibatch 8192 trained in 1 hour using 256 GPUs (~90% efficiency) with linear scaling of the learning rate and warmup, without loss of precision[19]. In Large Language Models (LLM), where collective communication can dominate, PCCL adjusts the frequency during communication to meet a minimum bandwidth, reducing the energy of these operations by up to 27% and the total training energy by 17.3%, with negligible effect on performance[20].

At the system scale, one analysis suggests that combining good practices (model improvements, hardware choice, automation and energy mix) could reduce emissions by between 65× and 747×; however, it emphasizes practical limitations: the lack of public data on Power Usage Effectiveness (PUE) and on carbon intensity (CI), and the difficulties of integrating carbon into online control[21]. In this direction, Zeus [22] proposes a just-in-time approach that explores configurations (batch size and power limit) and builds a per-job energy–time Pareto front; it reports improvements of 15.3% to 75.8% in energy efficiency, albeit with exploration overhead and challenges for its use in multi-user clusters.

This group of works suggests that evaluating with composite metrics (such as EDP/EDS) helps avoid incomplete conclusions based only on finishing earlier or only on spending less, when the real objective is to balance performance and sustainability[4].

E–T–CO_2 measurement and reporting

This group of works quantifies and reports the environmental impact of compute, that is, it focuses on measuring and transparently communicating how much energy is consumed, how long training takes and what CO_2e emissions are associated with that energy. The central idea is that, without consistent measurement, two training runs can report less CO_2 simply because they used different assumptions.

A first line seeks to anticipate consumption before running a job on HPC, which is useful for planning. Here, the aim is to predict the minimum/average/maximum power of a job using only information available at submission time, with online models (periodic retraining) and normalization by number of nodes (for example, dividing average power by the assigned nodes), achieving errors below 12% on Fugaku [23] and below 22% on Marconi100[14]. This approach helps allocate resources and energy budgets, but to estimate total energy it needs to be complemented with duration models; moreover, the integration of mechanisms robust to drift and the generation of prediction intervals that represent uncertainty remain open[14].

The second line measures and reports during training using instrumentation. EnergyVis [11] It captures live consumption using CPU counters (Intel Running Average Power Limit, RAPL) and GPU readings (NVIDIA SMI), and presents energy by epoch by integrating power over time to display estimated kWh and CO_2. Its extension includes preloaded and $(E_{epoch} = \int P(t)dt)$ live-tracking modes, as well as tools to compare hardware and extrapolate results. In a similar tracking approach, Carbontracker [10] estimates aggregate energy by incorporating Power Usage Effectiveness (PUE) from NVML/RAPL readings, while

eco2AI measures GPU/CPU/RAM and converts to CO_2e using regional coefficients and PUE [12]. There are also location- and hardware-based calculators that show wide variations in the emission factor and propose mitigations, although without direct measurement or continuous updating [24].

A key result for comparability is that different assumptions can substantially change the conclusions: a comparison in named entity recognition (NER) reported variations of up to 40% due to discrepancies in PUE, carbon factors, and measurement method, which reinforces the need to state the assumptions explicitly[13]. Work in the same direction has proposed estimation frameworks for large models (for example, T5 [25] and GPT-3[26]), explicit transparency recommendations on PUE and energy mix[27], and reporting guidelines that emphasize sensitivity to hyperparameters and hardware independence[28]. Empirically, large-scale measurements with wattmeters show that consumption is not well explained by FLOPs or by the number of parameters alone, and they reveal behavior linked to the memory hierarchy (a piecewise model with a threshold associated with cache capacity)[29]. In classical CNNs, a relationship between architectural complexity and energy/emissions is also observed, and a simple metric of the type Score = Accuracy / Energy is proposed, validating profilers as an economical alternative to wattmeters due to their high correlation[30].

The value of E–T–CO_2 reports depends on aligning definitions (what is measured and how it is converted) and on connecting micro metrics (components, pipeline phases) with macro metrics (kWh and CO_2e); otherwise, efficiency may be more of a methodological artifact than a real result[13],[29].

MATERIALS AND METHODS

The objective of this experimentation is the controlled evaluation of the training of a deep vision model under the fixed compute assumption. The experiments were run on a workstation with two NVIDIA RTX A6000 GPUs (49,140 MiB per GPU; 300 W power limit), an Intel Xeon Gold 6230R (26 cores, 52 threads), and 256 GB of RAM, under Ubuntu 22.04.5 LTS with Linux (kernel 6.8.0-94-generic) and NVIDIA driver 560.35.05. Training was implemented in PyTorch (version 2.8.0+cu126), using CUDA 12.6 and cuDNN 91002. To ensure comparability between configurations, all runs were executed on the same equipment and under a reproducible protocol.

Dataset and deep learning model

For the study, a standard and reproducible computer vision scenario based on CIFAR-10 (50,000 training images and 10,000 test images) was selected, using a representative deep training model, ResNet-50. To increase the computational load and make GPU scaling relevant, the images were resized to 224×224 and consistent training transformations were applied, keeping the data pipeline constant across all configurations.

Fixed compute budget and variability control

To reduce variability between runs, all configurations were trained with a fixed compute budget defined as a constant number of optimization steps $N_{steps} = 250$. A constant global batch size of samples per step was maintained. In multi-GPU mode, data parallelism was used and the batch was evenly divided by GPU (256 per GPU), preserving the global batch. Five repetitions were run per configuration, varying only the random seed (seeds) and keeping all hyperparameters and computational budget constant. For each run, 512seeds were set in Python/NumPy/PyTorch, and a reproducible sampling scheme for the dataloader was defined, including seed by worker, in order to minimize random variations.

The fixed budget of $N_{steps} = 250$ was adopted as an experimental control to compare all configurations with the same number of updates and processed samples, not to train the model to convergence. The results therefore describe energy efficiency and time during a partial training phase, and not necessarily the consumption of a complete training. Because the global batch and the number of steps were held constant, each run processed 128,000 samples (250 × 512), with the batch distributed evenly across the active GPUs.

Experimental design

A controlled factorial design was used to evaluate the impact of GPU scaling on the energy–time relationship and the effect of efficiency mechanisms at the system level:

- GPU scaling (G): 1 GPU vs 2 GPUs, using Distributed Data Parallel (DDP).
- Numerical precision (P): FP32 and AMP (mixed precision) were used.
- Per-GPU power limit (C): Varied between 300 W, 250 W, and 200 W.

This produces 12 configurations (2×2×3). Each configuration was repeated 5 times, for a total of 60 runs. The considerations were:

- Multi-GPU implementation: Multi-GPU training was implemented with PyTorch DDP and the NCCL backend, using torchrun and standard gradient synchronization. Optimizer parameters were held constant; SGD (lr=0.1,

momentum=0.9, weight_decay=1e−4) was used with Cross Entropy Loss, without a scheduler, with augmentations consisting of Resize + RandomCrop with padding + flip and normalization.

- Mixed precision (AMP): In the AMP configurations, autocast and gradient scaling were used (for example, torch.cuda.amp.autocast + GradScaler), keeping the other hyperparameters identical.
- Power capping: Before each run, the power limit of each active GPU was set via nvidia-smi -pl, with the value corresponding to the configuration (300, 250 or 200 W). The confirmation returned by the command was taken as operational verification that the driver had accepted the requested limit. This check does not constitute an independent electrical validation of instantaneous power.

Time measurement

The total training time $T$was measured as the interval elapsed from the start of the first step to the completion of step N_{steps}, excluding an unreported warm-up run. For consistency, a monotonic clock was used (time.monotonic() in Python) and the time was recorded in seconds.

Software-based energy measurement (CPU + GPU)

Because no external electrical instrumentation was used, energy was quantified at the component level using system-supported counters, with readings taken before and after each run.

CPU energy (Intel RAPL): The Linux powercap interface was used, reading accumulated CPU package energy from /sys/class/powercap/intel-rapl:0/energy_uj. For each run, E_{cpu}^{pre} and E_{cpu}^{post} were recorded (in μJ) and the following was calculated:

$$E_{cpu} = \frac{E_{cpu}^{post} - E_{cpu}^{pre}}{10^6} \quad (1)$$

If counter wrap-around occurs, it is corrected using max_energy_range_uj (recorded at the start of the experiment).

GPU energy (NVIDIA NVML): Accumulated GPU energy was measured via NVML, using the NVMLDeviceGetTotalEnergyConsumption counter (mJ). For each GPU i:

$$E_{\text{gpu}_i}\,[\text{J}] = \frac{E_{\text{gpu}_i}^{post} - E_{\text{gpu}_i}^{pre}}{10^3} \quad (2)$$

Total GPU energy is defined as:

$$E_{\text{gpu}} = \sum_i E_{\text{gpu}_i} \quad (3)$$

Total energy attributed to components is defined as:

$$E_{\text{comp}} = E_{\text{cpu}} + E_{\text{gpu}} \quad (4)$$

and converted to kWh for reporting:

$$E_{\text{comp}}\,[\text{kWh}] = \frac{E_{\text{comp}}\,[\text{J}]}{3.6 \times 10^6} \quad (5)$$

Explicit limitation (validity): E_{comp}represents energy attributed to CPU+GPU and does not include platform consumption not directly attributable (PSU losses, fans, storage, peripherals or cooling). Therefore, the results are interpreted as consistent relative comparisons within the same system and protocol.

- Average power: For the energy–time analysis, average power is defined as:

$$\tilde{P} = \frac{E_{\text{comp}}}{T} \quad (6)$$

Average power was used as an aggregate descriptor of each run. It does not represent instantaneous power, nor does it allow variations between input/output, compute, communication, synchronization and waiting to be distinguished; a phase-by-phase characterization would require higher-resolution temporal measurement and is beyond the scope of the study.

Estimation of carbon emissions (CO_2e)

Emissions associated with training were reported as a deterministic function of the measured energy. In particular, for each run the following was calculated:

Estimation based on measured energy:

$$CO_2e_{\text{meas}} = E_{\text{comp}}\,[\text{kWh}] \quad 7$$

where CI is the carbon intensity of the electricity grid ($\frac{kgCO^2e}{kWh}$). Because it depends on region and time, emissions were calculated in post-processing using three sensitivity scenarios: 0.10, 0.20 and 0.30 $kgCO_2e/kWh$. The value 0.20 $kgCO_2e/kWh$ was selected as the central scenario because it is the midpoint of the analyzed interval and facilitates comparative interpretation; it does not represent an instantaneous local measurement during the runs. Since CO_2e is derived linearly from the measured energy,

changing CI modifies the absolute magnitude of the emissions, but not the relative order of the configurations.

Estimation with CodeCarbon: Each run was instrumented with CodeCarbon to obtain $CO_2e_{cc}$and associated estimates. E_{comp} was compared vs the energy estimated by CodeCarbon and the relative error was reported:

$$\epsilon = \frac{|E_{cc} - E_{\text{comp}}|}{E_{\text{comp}}} \quad (8)$$

Statistical analysis

For each run, we recorded total time T, E_{cpu}, E_{gpu}, E_{comp}, average power $\overline{P}$, CO_2e (via measured energy and via CodeCarbon), and a learning sanity indicator ($accuracy_{top1}$ on the test split). The primary measurements of the study were T and E_{comp}; the remaining metrics are treated as derived or descriptive. The $accuracy_{top1}$ metric is reported only to verify that training under a fixed compute budget yields non-trivial learning and to rule out training failures or collapses; it is neither optimized nor used as a criterion for comparing configurations. The inferential analysis therefore focuses on T and E_{comp}. For CIFAR-10 (10 classes), we verified that $accuracy_{top1}$ exceeded the level expected by chance (0.10) as a sanity check.

Results were aggregated by configuration as mean ± standard deviation ($\mu \pm \sigma$) from five repetitions. Where applicable, 95% confidence intervals (IC95%) of the mean estimated via bootstrap were reported. To quantify the effect of the factors (G, P, C) and their interactions on T and E_{comp}, a three-way factorial analysis was used (ANOVA/GLM with interaction terms), together with post-hoc comparisons with multiple-testing correction (Holm), maintaining $\alpha = 0.05$.

RESULTS

We present the results of the 60 runs corresponding to the 12 configurations of the factorial design, with the compute budget ($N_{\text{steps}} = 250$) and the global batch (512) held constant to isolate the effect of the factors evaluated.

Overall results by configuration

Table 1 summarizes the results aggregated by configuration at the end of the fixed compute budget. In this set, average times ranged from 91.33 s (2G-AMP-300W) to 313.75 s (1G-FP32-200W), while the energy attributed to components E_{comp}Ecomp varied between 48,386 J (1G-AMP-200W) and 88,766 J (1G-FP32-300W). Figure 1 shows separately TT and E_{comp}Ecomp by configuration.

Table 1: Descriptive results by configuration ($\mu \pm \sigma$, n=5).

Config	T (s)	E_{cpu}(J)	E_{gpu}(J)	E_{comp}(J)	E_{comp}(kWh)	$\overline{P}$(W)	Acc Top-1
1G-FP32-300W	261.20±1.66	21484±2255	67281±341	88766±252	0.025±0.000702	339.8±7.7	0.2449±0.036
1G-FP32-250W	266.14±0.49	19708±63	60534±118	80241±113	0.022±0.000031	301.5±0.5	0.2390±0.048
1G-FP32-200W	313.75±3.56	22525±457	57720±632	80246±874	0.022±0.000243	255.8±1.6	0.2170±0.060
1G-AMP-300W	158.15±0.63	12869±252	40394±464	53263±660	0.014±0.000183	336.8±3.0	0.1858±0.052
1G-AMP-250W	160.82±1.27	12815±91	36471±396	49286±481	0.014±0.000134	306.5±0.9	0.2062±0.065
1G-AMP-200W	184.63±1.31	14453±164	33933±231	48386±337	0.014±0.000094	262.1±0.5	0.2323±0.038
2G-FP32-300W	140.13±0.68	12649±78	70151±315	82800±258	0.023±0.000072	590.9±4.2	0.2133±0.034
2G-FP32-250W	141.82±0.25	12781±50	65401±111	78182±107	0.022±0.000030	551.3±0.9	0.2149±0.035
2G-FP32-200W	173.42±1.41	15103±104	64720±526	79823±625	0.022±0.000174	460.3±1.1	0.2149±0.044
2G-AMP-300W	91.33±0.28	9269±55	44062±313	53331±315	0.015±0.000087	583.9±4.6	0.2252±0.045
2G-AMP-250W	92.01±0.15	9362±41	41503±182	50865±187	0.014±0.000052	552.8±1.9	0.2748±0.034
2G-AMP-200W	106.33±0.78	10440±17	39856±297	50296±300	0.014±0.000083	473.0±2.5	0.2211±0.042

Energy–time trade-off and operating region

Figure 2 summarizes the energy–time trade-off for the 12 evaluated configurations, using the per-configuration means ($n = 5$). A clear separation appears between FP32 and AMP: AMP configurations concentrate energy values in the range $\sim 48 - 53\ kJ$, whereas FP32 configurations fall in $\sim 78 - 89\ kJ$, indicating that, under the same compute budget, mixed precision shifts training toward a more energy-efficient region. The baseline (1G-FP32-300W) lies in the zone of highest energy and highest relative time, serving as a reference to quantify improvements.

The dashed line shows the Pareto front, that is, the set of configurations where energy and time cannot be simultaneously reduced by another configuration in the study. Only AMP configurations appear on this front, reflecting that FP32 alternatives are dominated in the energy–time plane by mixed-precision options. At the extremes of the front are the configuration with the lowest time (2G-AMP-300W) and the one with the lowest energy (1G-AMP-200W), which represent two distinct operating preferences: maximizing speed or minimizing consumption.

Between both extremes, an operating region (trade-off zone) is identified around 2G-AMP-250W, where energy is reduced with a practically negligible time penalty relative to the fastest point on the front. In particular, compared with 2G-AMP-300W, the 250 W limit reduces energy by $\sim 4 - 5\%$ with a time increase of $< 1\%$, whereas dropping to 200 W provides a small additional energy saving with a much larger time penalty. This behavior suggests that, in this system and under this protocol,

the point near 250 W per GPU offers a practical balance between acceleration and energy sustainability.

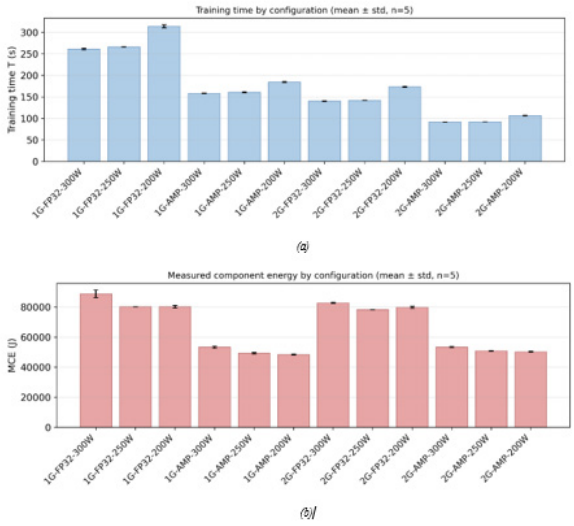

Fig. 1. Results by configuration, (a) total training time T and (b) energy attributed to components E_{comp}.

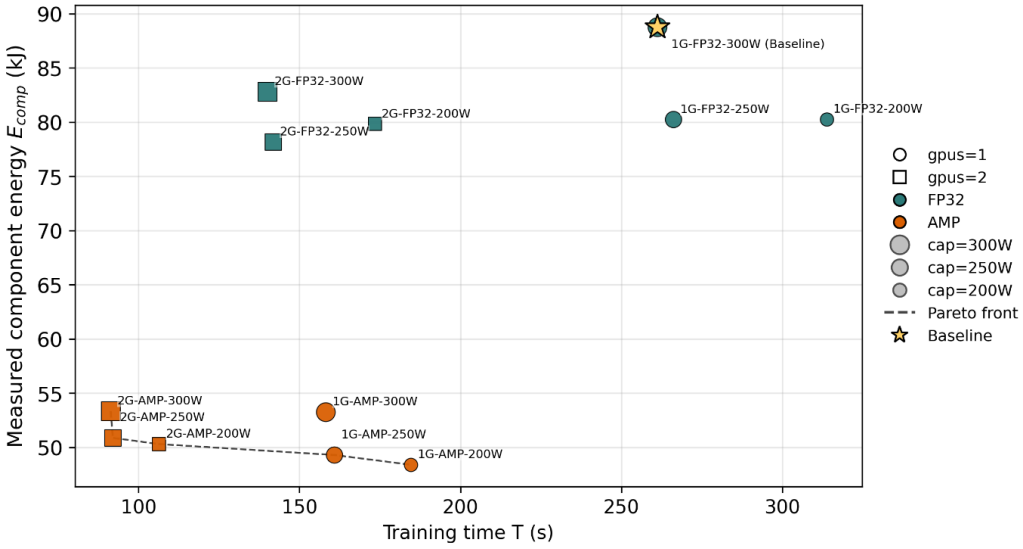

Fig. 2. *Energy–time trade-off at the configuration level (means; n=5). The dashed line highlights the non-dominated configurations (Pareto front) and the star marks the baseline (1G-FP32-300W).*

Effect of scaling vs AMP vs power cap

To compare the factors directly, the 1G–FP32–300W configuration was used as a baseline and relative metrics were reported: speedup, relative energy change, and normalized composite metrics for the energy–time tradeoff: and . These measures allow separating when a technique accelerates and when it actually improves efficiency (see Figure 3) $S = T_{\text{base}}/T \Delta E = 1 - E/E_{\text{base}} EDP_{\text{rel}} = (E \cdot T)/(E_{\text{base}} \cdot T_{\text{base}}) EDS_{\text{rel}} = (E/E_{\text{base}}) + (T/T_{\text{base}})$.

- GPU scaling (DDP): In FP32, moving from 1 to 2 GPUs produced a consistent speedup of $1.81\times - 1.88\times$ (time reduction of $\sim 44.7\% - 46.7\%$, depending on the cap) (see Figure 3a). In measured total energy E_{comp}, the effect was small: from -0.5% (200W) to -6.7% (300W) (see Figure 3b). That is, scaling clearly reduces time, but energy savings depend on whether the reduction in duration offsets the higher average power level when activating two GPUs. In AMP, scaling also accelerated (speedup $\sim 1.73\times - 1.75\times$; time reduction $\sim 42\% - 43\%$), but did not guarantee energy savings: E_{comp} increased slightly between $+0.1\%$ and $+3.9\%$ when moving to 2 GPUs (see Figure 3b). This suggests that, under this fixed budget and on this equipment, the overhead associated with multi-GPU synchronization/execution can erode the energy benefit even when training finishes earlier.
- A plausible explanation is that AMP reduces local compute time, whereas DDP maintains gradient synchronization and the

collective operations managed by NCCL. As compute accelerates, these overheads and waiting periods can gain greater relative weight; together with the additional power of a second GPU, this can prevent the time reduction from producing proportional energy savings. This interpretation is based on aggregated metrics, since the protocol did not decompose energy into compute, communication, and waiting.

- Effect of AMP: Keeping the number of GPUs and the power cap fixed, AMP shifted the system toward a clearly more efficient region: it reduced time by between $\sim 35\%$ and $\sim 41\%$ and, simultaneously, reduced energy E_{comp} by between $\sim 35\%$ and $\sim 40\%$ compared with FP32 (see Figure 3a-b). This pattern was stable both on 1 GPU and on 2 GPUs and across the three caps. In other words, AMP provides a robust improvement of the energy–time trade-off (see Figure 3c), not just an acceleration.
- Effect of power capping: Power capping showed the expected trade-off: lowering the limit reduces energy but increases time. However, the effect is not linear (see Figure 4).
- When moving from 300W to 250W (with GPU and precision fixed), energy decreased consistently (~4.6%–9.6%) with a minimal time penalty $(\sim 0.7\% - 1.9\%)$. Consequently, the energy–time trade-off improves (reduction of EDP_{rel} of $\sim 3.9\% - 7.9\%$ within the same group).
- When moving from 300W to 200W, although energy decreased ($\sim 3.6\% - 9.6\%$), time increased markedly ($\sim 16\% - 24\%$), which worsens the overall trade-off (EDP_{rel} increases $\sim 6\% - 19\%$). Therefore, 200W may be useful if the goal is to minimize energy unconditionally, but it is not the best option when seeking a practical energy–time balance.

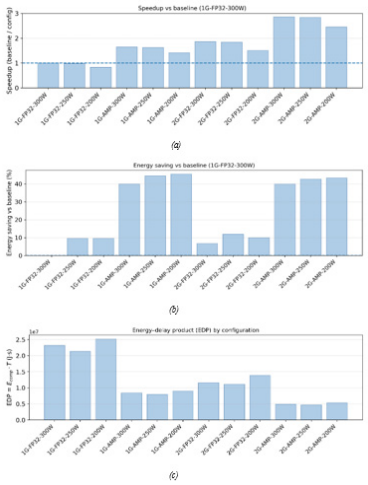

Fig. 3. *Combined impact of scaling, AMP, and power capping relative to the baseline, in a) speedup, b) relative energy savings, and c) normalized energy-delay product EDP.*

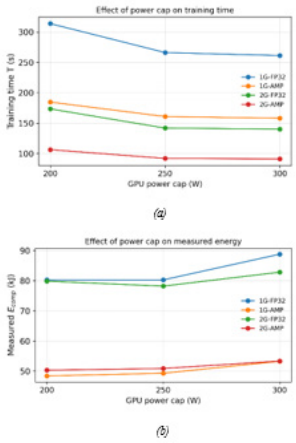

Fig. 4. *Effect of power capping on training time. a) effect on time and b) effect on measured energy.*

Consistency, variability and CI95

In order to verify the stability of the measurements under the experimental protocol (5 repetitions per configuration, varying only the seed), intra-configuration variability was quantified for the total training time T\u00A0and the measured energy E_{comp}. For each of the 12 configurations, the coefficient of variation $CV = 100 \cdot \sigma / \mu$\u00A0and the standard deviation

(σ)\u00A0were calculated from the five repetitions; these values were then summarized across all configurations. The results show low and consistent dispersion, indicating that the differences observed in the previous sections are dominated by the experimental factors (GPU, AMP and power cap) and not by randomness between seeds (Table 2).

Table 2: Intra-configuration variability (n=5 repetitions per configuration).

Variable	n	CV% (median$^{min,\ max}$)	σ(median$^{min,\ max}$)
Time (T) (s)	5	0.559% [0.168, 1.135]	0.729 s [0.155, 3.560]
Energy ($^{E_{comp}}$) (kJ)	5	0.646% [0.136, 2.849]	0.326 kJ [0.107, 2.529]

Comparison of CodeCarbon and direct measurement

To assess the agreement between a widely used estimate and direct measurement, the energy estimated by CodeCarbon was compared E_{cc} with the measured energy attributed to components E_{comp}, both in kWh, at the race level ($N = 60$). The relationship E_{cc} vs. E_{comp} is shown in Figure 5a, where it can be seen that CodeCarbon reproduces the overall trend across configurations, although with a systematic deviation from direct measurement. To quantify this discrepancy, the relative error was calculated $\varepsilon = |E_{cc} - E_{comp}| \ / E_{comp}$ and its distribution was summarized in Figure 5b. CodeCarbon showed a median relative error of 4.09% (IQR: 3.03%–5.71%) and tended to underestimate the measured energy (median bias ≈ −4.09%).

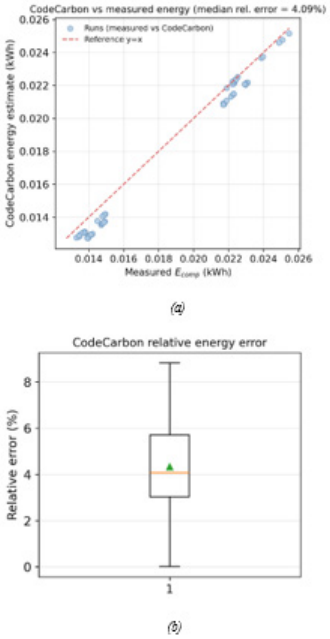

Fig. 5. *Comparison between CodeCarbon and direct energy measurement, (a) Energy estimated by CodeCarbon vs. measured energy attributed to components, (b) Distribution of the relative error.*

Factorial analysis (ANOVA/GLM) and post-hoc

A three-way factorial model was fitted to quantify the effect of G (number of GPUs), P (precision: FP32 vs AMP) and C (power cap), including interactions, on T and E_comp. The ANOVA results (Table 3) show that, for training time, the main effects of G and P dominate the observed variation ($\eta_p^2 \approx 0.999$ in both cases), while C also contributes a considerable effect ($\eta_p^2 = 0.993$). The interactions were significant, indicating that the impact of C and of scaling depends on precision and on the number of GPUs (see Figure 6).

For measured energy E_{comp}, the dominant factor was P (AMP) ($\eta_p^2 = 0.998$), followed by C ($\eta_p^2 = 0.902$), while the direct effect of G was smaller but statistically significant (η²_p=0.224), see Table 3. The interactions (especially $G \times P$ and $G \times C$) were also relevant, confirming that the energy changes due to scaling and to power capping are not additive and depend on the operating regime. Finally, the post-hoc comparisons (Holm) restricted to C within each combination (G, P) confirmed that 250W reduces E_{comp} compared with 300W in all cases ($p_Holm < 0.01$); the additional reduction to 200W was consistent under AMP, but not necessarily under FP32 (for example, with 2 GPUs in FP32, 200W slightly increased E_{comp} compared with 250W, $p_{Holm} < 0.01$).

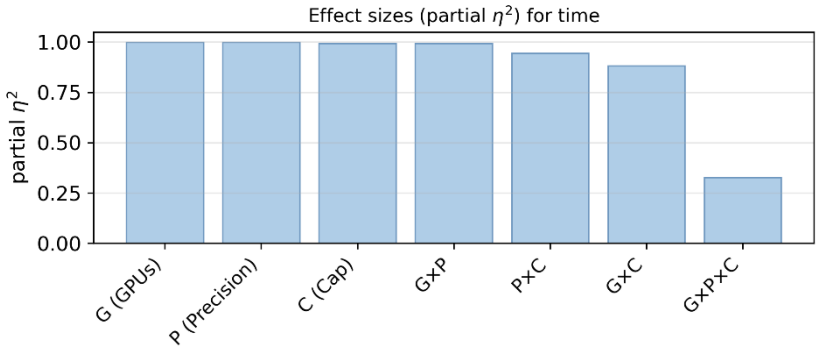

Fig. 6. *Effect sizes of the factorial ANOVA (G×P×C) on training time T, including main effects and interactions.*

Table 3: Results of the factorial ANOVA (three-way) for T and E_comp.

Effect	F(T)	p(T)	η²p(T)	F(E_comp)	p(E_comp)	η²p(E_comp)
G	79508.0	<1e-4	0.999	13.8	<1e-3	0.224
P	55985.5	<1e-4	0.999	19716.8	<1e-4	0.998
C	3325.0	<1e-4	0.993	219.9	<1e-4	0.902
G×P	6525.9	<1e-4	0.993	83.3	<1e-4	0.635
G×C	179.4	<1e-4	0.882	25.4	<1e-4	0.514
P×C	405.5	<1e-4	0.944	19.5	<1e-4	0.448
G×P×C	11.6	<1e-4	0.326	6.1	0.004	0.203

Sensitivity of CO_2e to carbon intensity (CI)

Since CO^2e_{meas} was defined in post-processing as a deterministic function of measured energy ($CO^2e_{meas} = Ecomp[kWh] \times IC$), the sensitivity to CI is strictly linear: as CI increases, all emissions scale proportionally, but the relative order among configurations and the set of preferable configurations in the energy–time plane do not change. Figure 7 shows this variation for 0.10, 0.20 and 0.30 $kgCO_2e/kWh$, keeping the measured energy per configuration fixed.

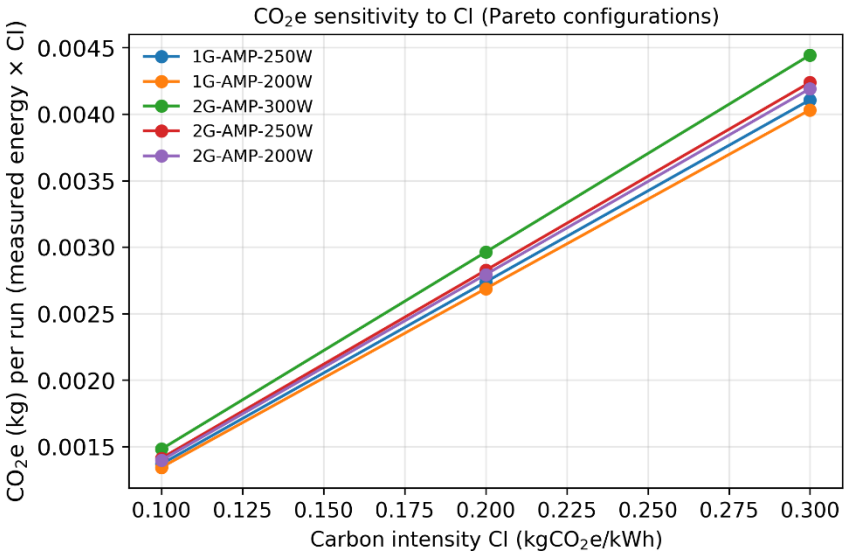

Fig. 7. *Sensitivity of estimated emissions to carbon intensity (CI).*

DISCUSSION

This work isolated the effect of scaling, precision and power capping on the energy–time trade-off during training. Under these conditions, the central result is that speeding up does not necessarily mean consuming less energy, and that system settings such as mixed precision and power limits can be more decisive for efficiency than parallelism alone.

First, AMP was the dominant factor: it shifted configurations toward substantially lower times and energies, clearly separating them from the FP32 group in the energy–time plane. This separation explains why the Pareto front consists solely of configurations with AMP. That is, within the evaluated space, FP32 is dominated by AMP alternatives that are simultaneously faster and less energy-expensive. In practical terms, AMP behaves as a robust improvement of the energy–time trade-off, not

merely as an accelerator.

Second, DDP consistently reduced time, but its impact on energy was small and regime-dependent: in FP32 modest savings are observed, whereas in AMP the saving is not guaranteed and may be reversed. This reflects that enabling two GPUs increases the average power level of the system, and total energy only decreases if the reduction in duration offsets that increase and the synchronization/communication overheads.

Finally, power capping showed a practical balance point. Lowering from 300W to 250W reduced energy with a minimal time penalty, improving composite metrics such as EDP. In contrast, 200W introduced much larger time penalties, which tend to deteriorate the energy–time balance despite reductions in absolute energy. Under this hardware and protocol, this identifies an operating region near 250W per GPU as a reasonable trade-off between sustainability and productivity.

As a complement, the comparison with CodeCarbon indicates that the tool reproduces trends but shows a systematic discrepancy with respect to direct measurement (median relative error $\sim 4\%$, with underestimation). Therefore, CodeCarbon is useful for monitoring and comparisons, but in fine-grained analyses it should be accompanied by direct measurement or its error should be explicitly reported. Lastly, sensitivity to CI is strictly linear: it changes the scale of CO_2e, but does not alter the relative order among configurations.

The scope of these results is conditioned by the experimental protocol: ResNet-50 on CIFAR-10, a workstation with NVIDIA RTX A6000 GPUs and partial training with a fixed compute budget. The magnitude of the observed effects should therefore not be extrapolated directly to transformer architectures or foundation models, larger-scale datasets, other accelerators or systems with more than two GPUs, where the relationships among compute, memory, communication, time and energy may change.

The scaling evaluated was limited to the transition from one to two GPUs within a single workstation. Consequently, the study does not characterize massive distributed or multi-node scaling, where collective synchronization, inter-node communication, interconnect topology and bandwidth, congestion and overlap between compute and communication may modify the energy-time trade-off.

CONCLUSIONS

This work reports reproducible evidence, under a fixed compute budget, on how common operational decisions reconfigure the energy-time trade-off and, by extension, the carbon footprint associated with training. First, it is confirmed that multi-GPU scaling (1 to 2 GPUs) consistently accelerates, but does not guarantee energy savings; that is, time decreases markedly, whereas the total energy attributed to CPU+GPU compute may change little or even increase depending on the regime. This reinforces a key idea from an environmental perspective: finishing earlier reduces the system's exposure time, but the climate impact depends on the product of duration and power, not on time alone.

Second, practical criteria for improving efficiency are derived. AMP shifts training toward a more efficient region, simultaneously reducing time and energy, which makes it a robust decision when the goal is to lower consumption and emissions without sacrificing productivity. Power capping allows the balance to be fine-tuned; in this system, 250W per GPU offers a favorable trade-off, consistently reducing energy with a minimal time penalty, whereas 200W tends to penalize time disproportionately for marginal energy savings, being more appropriate when strictly prioritizing reduced consumption even if duration increases.

Third, the analysis by means of time-energy frontiers provides a clear and comparable reading of the configurations, separates extremes (minimum time vs. minimum energy) and identifies an intermediate operating region where the balance between acceleration and sustainability is more practical. This representation is especially useful for guiding real decisions, because it makes explicit that there is no universal best configuration, but rather optimal choices conditioned by the objective (speed, energy or balance).

Finally, regarding emissions, this work reports CO_2e as a deterministic function of measured energy, which allows the environmental impact to be interpreted transparently; energy is the bridge between execution and climate, and the factor that translates kWh into CO_2e is the carbon intensity (IC) of the electricity grid. Under this formulation, changing CI does not alter the ranking of configurations (it only scales emissions), but it does affect the absolute impact, underscoring that the sustainability of training depends both on system efficiency and on the regional energy context. In this framework, CodeCarbon is useful for reproducing trends and facilitating comparable reports, but in this experiment it showed a systematic underestimation relative to direct measurement (median relative error $\sim 4\%$). Therefore, for environmental conclusions involving fine differences between configurations, it is advisable to accompany CodeCarbon-type estimates with direct measurement or, at least, to explicitly report their error and bias, so that the estimated CO_2e is not interpreted as an unquestionable absolute value but as an approximation with quantified uncertainty. Consequently, the results should be interpreted within this protocol and not as a general characterization of training runs to convergence or of larger-scale distributed systems.

Form start

REFERENCES

[1] Zhou, H.A.; Wolfschläger, D.; Florides, C.; Werheid, J.; Behnen, H.; Woltersmann, J.-H.; Pinto, T.C.; Kemmerling, M.; Abdelrazeq, A.; Schmitt, R.H. Generative AI in Industrial Machine Vision: A Review. J Intell Manuf 2025, doi:10.1007/s10845-025-02604-6.

[2] Chinnaiyan, B.; Balasubaramanian, S.; Jeyabalu, M.; Warrier, G.S. AI Applications – Computer Vision and Natural Language Processing. In Model Optimization Methods for Efficient and Edge AI; John Wiley & Sons, Ltd, 2025; pp. 25–41 ISBN 978-1-394-21923-0.

[3] Dorigo, T.; Brown, G.D.; Casonato, C.; Cerdà, A.; Ciarrochi, J.; da Lio, M.; D'Souza, N.; Gauger, N.R.; Hayes, S.C.; Hofmann, S.G.; et al. Artificial Intelligence in Science and Society: The Vision of USERN. IEEE Access 2025, 13, 15993–16054, doi:10.1109/ACCESS.2025.3529357.

[4] Krzywaniak, A.; Czarnul, P.; Proficz, J. Dynamic GPU Power capping with Online Performance Tracing for Energy Efficient GPU Computing Using DEPO Tool. Future Generation Computer Systems 2023, 145, 396–414, doi:10.1016/j.future.2023.03.041.

[5] Sevilla, J.; Heim, L.; Ho, A.; Besiroglu, T.; Hobbhahn, M.; Villalobos, P. Compute Trends Across Three Eras of Machine Learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*; IEEE, 2022; pp. 1–8, doi:10.1109/IJCNN55064.2022.9891914.

[6] LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. Nature 2015, 521, 436–444, doi:10.1038/nature14539.

[7] Matsuo, Y.; LeCun, Y.; Sahani, M.; Precup, D.; Silver, D.; Sugiyama, M.; Uchibe, E.; Morimoto, J. Deep Learning, Reinforcement Learning, and World Models. Neural Networks 2022, 152, 267–275, doi:10.1016/j.neunet.2022.03.037.

[8] Hussen, J.; Polas, M.R.H.; Momotaz, T.; Eva, R.A.; Emu, M.N.A. Green AI for Sustainable Development in Circular Economies: Building a Low-Carbon, High-Impact Future in Industry 4.0 and 5.0. In Achieving the Sustainable Development Goals Through Green Innovation; IGI Global Scientific Publishing, 2026; pp. 71–102 ISBN 979-8-3373-7694-3.

[9] Litjens, G.; Kooi, T.; Bejnordi, B.E.; Setio, A.A.A.; Ciompi, F.; Ghafoorian, M.; van der Laak, J.A.W.M.; van Ginneken, B.; Sánchez, C.I. A Survey on Deep Learning in Medical Image Analysis. Medical Image Analysis 2017, 42, 60–88, doi:10.1016/j.media.2017.07.005.

[10] Anthony, L.F.W.; Kanding, B.; Selvan, R. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. *arXiv* 2020, arXiv:2007.03051, doi:10.48550/arXiv.2007.03051.

[11] Shaikh, O.; Saad-Falcon, J.; Wright, A.P.; Das, N.; Freitas, S.; Asensio, O.I.; Chau, D.H. EnergyVis: Interactively Tracking and Exploring Energy Consumption for ML Models. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*; ACM, 2021; pp. 1–7, doi:10.1145/3411763.3451780.

[12] Budennyy, S.A.; Lazarev, V.D.; Zakharenko, N.N.; Korovin, A.N.; Plosskaya, O.A.; Dimitrov, D.V.; Akhripkin, V.S.; Pavlov, I.V.; Oseledets, I.V.; Barsola, I.S.; et al. Eco2ai: Carbon Emissions Tracking of Machine Learning Models as the First Step towards Sustainable Ai. Doklady Mathematics 2022, 106, S118–S128, doi:10.1134/S1064562422060230.

[13] Bannour, N.; Ghannay, S.; Névéol, A.; Ligozat, A.-L. Evaluating the Carbon Footprint of Nlp Methods: A Survey and Analysis of Existing Tools. In Proceedings of the Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing; Moosavi, N.S., Gurevych, I., Fan, A., Wolf, T., Hou, Y., Marasović, A., Ravi, S., Eds.; Association for Computational Linguistics: Virtual, November 2021; pp. 11–21.

[14] Antici, F.; Borghesi, A.; Kiziltan, Z.; Domke, J.; Bartolini, A. An Online Algorithm for Power Consumption Prediction of HPC Workload. *Future Generation Computer Systems* 2026, 175, 108064, doi:10.1016/j.future.2025.108064.

[15] Chung, J.-W.; Gu, Y.; Jang, I.; Meng, L.; Bansal, N.; Chowdhury, M. Reducing Energy Bloat in Large Model Training. In Proceedings of the Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles; ACM: Austin TX USA, November 2024; pp. 144–159.

[16] CodeCarbon Available online: https://codecarbon.io/ (accessed on 14 February 2026).

[17] Tang, Z.; Wang, Y.; Wang, Q.; Chu, X. The Impact of GPU DVFS on the Energy and Performance of Deep Learning: An Empirical Study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*; ACM, 2019; pp. 315–325, doi:10.1145/3307772.3328315.

[18] Gu, D.; Xie, X.; Huang, G.; Jin, X.; Liu, X. Energy-Efficient GPU Clusters Scheduling for Deep Learning. *arXiv* 2023, arXiv:2304.06381, doi:10.48550/arXiv.2304.06381.

[19] Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv* 2017, arXiv:1706.02677, doi:10.48550/arXiv.1706.02677.

[20] Jia, Z.; Bhuyan, L.N.; Wong, D. PCCL: Energy-Efficient LLM Training with Power-Aware Collective Communication. In *2024 IEEE 42nd International Conference on Computer Design (ICCD)*; IEEE, 2024; pp. 84–91, doi:10.1109/ICCD63220.2024.00023.

[21] Patterson, D.; Gonzalez, J.; Hölzle, U.; Le, Q.; Liang, C.; Munguia, L.-M.; Rothchild, D.; So, D.R.; Texier, M.; Dean, J. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *Computer* 2022, 55(7), 18–28, doi:10.1109/MC.2022.3148714.

[22] You, J.; Chung, J.-W.; Chowdhury, M. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*; USENIX Association, 2023; pp. 119–139. Available online: https://www.usenix.org/conference/nsdi23/presentation/you.

[23] Fugaku Available online: https://www.r-ccs.riken.jp/en/fugaku/index.html (accessed on 14 February 2026).

[24] Lacoste, A.; Luccioni, A.; Schmidt, V.; Dandres, T. Quantifying the Carbon Emissions of Machine Learning. *arXiv* 2019, arXiv:1910.09700, doi:10.48550/arXiv.1910.09700.

[25] Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 2020, 21(140), 1–67.

[26] Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et

al. Language Models Are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33*; 2020; pp. 1877–1901.

[27] Patterson, D.; Gonzalez, J.; Le, Q.; Liang, C.; Munguia, L.-M.; Rothchild, D.; So, D.; Texier, M.; Dean, J. Carbon Emissions and Large Neural Network Training. *arXiv* 2021, arXiv:2104.10350, doi:10.48550/arXiv.2104.10350.

[28] Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*; Association for Computational Linguistics, 2019; pp. 3645–3650, doi:10.18653/v1/P19-1355.

[29] Tripp, C.E.; Perr-Sauer, J.; Gafur, J.; Nag, A.; Purkayastha, A.; Zisman, S.; Bensen, E.A. Measuring the Energy Consumption and Efficiency of Deep Neural Networks: An Empirical Analysis and Design Recommendations. *arXiv* 2024, arXiv:2403.08151, doi:10.48550/arXiv.2403.08151.

[30] Xu, Y.; Martínez-Fernández, S.; Martinez, M.; Franch, X. Energy Efficiency of Training Neural Network Architectures: An Empirical Study. In *Proceedings of the 56th Hawaii International Conference on System Sciences*; 2023; pp. 781–790, doi:10.24251/HICSS.2023.098.

FUNDING

No financing

CONFLICT OF INTEREST

None

AUTHORSHIP CONTRIBUTIONS

Drafting – original draft: Carlos Nelson Henríquez Miranda, Malak Andrés Sánchez Cataño, German Sánchez Torres.

Writing–review and editing: Carlos Nelson Henríquez Miranda, Malak Andrés Sánchez Cataño, German Sánchez Torres.