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

PhD. Carlos Nelson Henríquez Miranda[1], **Ing. Malak Andrés Sánchez Cataño**[1], **PhD. German Sánchez Torres**[1]

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

Correspondence: chenriquezm@unimagdalena.edu.co

Received: march 24, 2026. **Revised:** june 12, 2026. **Accepted:** july 10, 2026. **Published:** july 15, 2026.

How to cite: C. N. Henríquez Miranda, M. A. Sánchez Cataño, and G. Sánchez Torres, "Exploring the environmental effect of multi-GPU scaling on the energy-time-CO_2e relationship during the training of vision deep learning models", *Revista Colombiana de Tecnologías de Avanzada (RCTA)*, vol. 2, no. 48, pp. 124–139, jul. 2026, doi: 10.24054/rcta.v2i48.4569

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.

1. INTRODUCTION

Training deep vision models has become a central component of multiple scientific and industrial applications [1–3], enabled by the widespread adoption of GPU (Graphics Processing Unit) accelerators and hybrid CPU (Central Processing Unit) +GPU architectures in both workstations and HPC (High-Performance Computing) clusters [4]. Evidence on historical trends shows that training compute has increased rapidly [5] in the Deep Learning era [6], [7], with doubling times on the order of months, alongside an additional transition toward large-scale models [5]. This raises environmental concerns due to their high energy consumption [8]. In compute-intensive domains such as medical imaging for instance, MRI tasks using Deep Learning [9], the pressure to reduce training time 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 the HPC scale, large aggregate power draws have been reported in leading systems, highlighting 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 to obtain marginal performance improvements, making the energy cost of such gains relevant and necessitating a distinction between acceleration and energy efficiency [10]. In this context, GPU scaling (moving from 1 to multiple GPUs) is frequently justified by reduced training time; however, total energy (kWh) and its translation into emissions (kgCO^2e) may not decrease proportionally. Communication and synchronization overheads, utilization variability, and system baseline power can erode (or even reverse) expected energy benefits, especially when the analysis is end-to-end and includes CPU+GPU under comparable assumptions.

The recent literature offers relevant but fragmented contributions to understand this energy–time trade-off. On the one hand, GPU energy control via power capping and its effect on metrics such as EDP (Energy–Delay Product) or EDS (Energy–Delay Sum) has been studied, with consumption reductions that depend on the compute regime and the optimization criterion [4]. On the other hand, tools and approaches to track or estimate energy and carbon footprint during training have become popular, enabling hardware and scenario comparisons via regional emission factors [10–12]. However, it has also been noted that a primary limitation for comparability and traceability across studies stems from heterogeneity in assumptions [13], namely what is measured, with which instrument, and how it is converted to emissions. In parallel, HPC research has explored models to predict job power before execution, emphasizing that observed consumption depends both on the application and the operating context [14]. Complementarily, notions such as energy bloat and time–energy frontiers help separate consumption that contributes to throughput from consumption associated with waiting, synchronization, or other inefficiencies that can distort conclusions if attention is restricted to time alone [15].

Despite this, a practical gap remains: there is a lack of directly comparable evidence characterizing how

scaling from mono-GPU to multi-GPU alters the energy–time relationship when common efficiency practices are combined with explicit power-limiting mechanisms, under strict comparison conditions. In particular, many reports do not control effective workload under a fixed compute budget nor precisely delimit the measurement scope, thereby amplifying methodological heterogeneity [13].

This work explores this gap through a controlled evaluation of training a deep vision model (dataset: CIFAR-10, architecture: ResNet-50) under a fixed compute budget. The effects 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 are studied. Energy is measured at the component level by combining Intel RAPL counters for CPU and GPU accumulated 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]. With this, the work aims to contribute (i) reproducible evidence on when scaling accelerates without necessarily saving energy, (ii) practical criteria to select efficient configurations with AMP and power capping [4], and (iii) a consistent reading of the energy–performance trade-off grounded in the time–energy frontier framework and in 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 operating context in observed consumption [14].

2. RELATED WORKS

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

2.1. Optimization and control for E–T–CO_2

A practical way to reduce the environmental impact of training—regarding consumed energy and associated CO_2e—is to avoid assuming that faster necessarily means cleaner. In practice, total energy depends on how long training lasts (T) and how much power the hardware draws while operating (P); therefore, many strategies aim to identify configurations that reduce total energy (E) or achieve better compromises between energy and time. For consistent comparisons, composite metrics are used, such as the Energy–Delay Product (EDP = E·T) and the Energy–Delay Sum (EDS = E + T).

In a first line of work, the focus is on limiting how much power the GPU can consume (power capping) or on the operating frequency (via Dynamic Voltage and Frequency Scaling, DVFS). Using measurements based on the NVIDIA Management Library (NVML), energy savings of up to 30% are reported with performance penalties below 12%; additionally, measurement quality is evaluated by comparison against 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 with power—enables 8.7%–23.1% energy savings for deep neural network training (and 19.6%–26.4% for inference), and even reports performance improvements of up to 33% relative to default settings [17]. Under the same real-time control rationale, the 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 >22% reductions in energy (E) with <20% slowdowns when prioritizing energy, and 15%–18% savings with <8% penalty when prioritizing the trade-off (EDP/EDS) [4].

In the second line of work, the emphasis is that multi-GPU training introduces additional consumption that does not always contribute to learning: waiting, 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 frontier for the workload, reporting savings of up to 30% for GPT-3 and BLOOM without loss of effective performance (throughput). PowerFlow [18] proposes a simple model in which consumption depends on the number of GPUs, local minibatch size, and frequency; it then allocates resources to maximize energy efficiency without exceeding a budget, improving time-to-completion by 1.57× to 3.39× relative to baselines under the same consumption [18]. There is also evidence that increasing minibatch size can improve performance per unit of energy: a representative result shows ResNet-50 on ImageNet with minibatch 8192 trained in 1 hour using 256 GPUs (~90% efficiency) with linear learning-rate scaling and warmup, without loss of accuracy [19]. For Large Language Models (LLMs),

where collective communication can dominate, PCCL adjusts frequency during communication to meet a minimum bandwidth, reducing communication energy by up to 27% and total training energy by 17.3%, with negligible impact on performance [20].

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

From this body of work, it follows that evaluation with composite metrics (such as EDP/EDS) helps avoid incomplete conclusions based only on finishing earlier or only on consuming less, when the underlying goal is to balance performance and sustainability [4].

2.2. Measurement and reporting of E–T–CO_2

This group of studies quantifies and reports the environmental impact of computing, focusing 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 may report lower CO_2 only because they adopted different assumptions.

A first line of work seeks to anticipate consumption before executing an HPC job, which is useful for planning. Here, the goal 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 (e.g., dividing average power by allocated nodes), achieving errors below 12% on Fugaku [23] and below 22% on Marconi100 [14]. This approach supports resource allocation and energy budgeting, but estimating total energy requires complementary duration models; moreover, the integration of robust mechanisms against drift and the generation of predictive intervals to represent uncertainty remain open problems [14].

A second line measures and reports during training through instrumentation. EnergyVis [11] captures live consumption using CPU counters (Intel Running Average Power Limit, RAPL) and GPU readings (NVIDIA SMI), and reports per-epoch energy by integrating power over time ($E_{epoch} = \int P(t)dt$) to visualize kWh and estimated CO_2. Its extension incorporates preloaded and 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 variation in emission factors and propose mitigations, although without direct measurement or continuous updates [24].

A key result for comparability is that different assumptions can substantially change conclusions: a named entity recognition (NER) comparison reported variations of up to 40% due to discrepancies in PUE, carbon factors, and measurement methodology, reinforcing the need to explicitly declare assumptions [13]. In the same direction, estimation frameworks have been proposed for large models (e.g., T5 [25] and GPT-3 [26]) and explicit transparency recommendations regarding PUE and energy mix [27], alongside reporting guidelines that emphasize hyperparameter sensitivity and hardware independence [28]. Empirically, large-scale measurements with power meters show that consumption is not well explained solely by FLOPs or parameter counts, and reveal behavior linked to the memory hierarchy (a piecewise model with a threshold associated with cache capacity) [29]. For classical CNNs, a relationship between architectural complexity and energy/emissions is also observed, and a simple metric such as Score = Accuracy / Energy is proposed, validating profilers as a low-cost alternative to power meters due to high correlation [30].

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

3. MATERIALS AND METHODS

The objective of this experimentation is a controlled evaluation of training a deep vision model under a fixed compute budget assumption. Experiments were conducted on a workstation equipped with two NVIDIA RTX A6000 GPUs (49,140 MiB per GPU; 300 W power limit), an Intel Xeon Gold 6230R CPU (26 cores, 52 threads), and 256 GB of RAM, running 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 across configurations, all runs were executed on the same machine under a reproducible protocol.

3.1. Dataset and deep learning model

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

3.2 Fixed compute budget and variability control

To reduce run-to-run variability, all configurations were trained under a fixed compute budget defined as a constant number of optimization steps, $N_{\text{steps}} = 250$. A constant global batch size of 512 samples per step was maintained. In the multi-GPU setting, data parallelism was used and the batch was evenly split across GPUs (256 per GPU), preserving the global batch size. Five repetitions were performed per configuration, varying only the random seed while keeping all hyperparameters and the compute budget fixed. For each run, seeds were set in Python/NumPy/PyTorch, and a reproducible sampling scheme for the dataloader was defined, including per-worker seeding, to minimize variability due to randomness.

The fixed budget of $N_{steps} = 250$ was adopted as an experimental control to compare all configurations using the same number of optimization updates and processed samples, rather than to train the model until convergence. Consequently, the reported results characterize the energy and time efficiency during a partial training phase and do not necessarily represent the energy consumption of a complete training process. By keeping both the global batch size and the number of optimization steps constant, each run processed a total of 128,000 samples (250 × 512), with the batch evenly distributed across the active GPUs.

3.3. Experimental design

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

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

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

- Multi-GPU implementation: Multi-GPU training was implemented with PyTorch DDP and the NCCL backend, using torchrun and standard gradient synchronization. Optimizer settings were kept constant, using SGD (lr = 0.1, momentum = 0.9, weight_decay = 1e−4) with Cross-Entropy loss, no scheduler, and consistent augmentations (Resize + RandomCrop with padding + horizontal flip + normalization).
- Mixed precision (AMP): For AMP configurations, autocast and gradient scaling were used (e.g., torch.cuda.amp.autocast + GradScaler), keeping all other hyperparameters identical.
- Power capping: Before each experimental run, the power limit of every active GPU was configured using the nvidia-smi -pl command with the value specified for the corresponding experimental configuration (300, 250, or 200 W). The confirmation message returned by the command was used as operational verification that the GPU driver had accepted the requested power limit. However, this confirmation should not be interpreted as an independent electrical validation of the instantaneous power delivered by the GPU.

3.4. Time measurement

Total training time $T$was measured as the elapsed interval 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

(time.monotonic() in Python) was used and time was recorded in seconds.

3.5. Software-based energy measurement (CPU + GPU)

Since external electrical instrumentation was not 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 by reading the CPU package accumulated energy from /sys/class/powercap/intel-rapl:0/energy_uj. For each run, E_{cpu}^{pre} and E_{cpu}^{post} (in µJ) were recorded, and the following was computed:

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

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

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

  Total GPU energy is defined as:

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

  Total component-attributed energy is defined as:

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

  and is converted to kWh for reporting:

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

  Explicit limitation (validity): E_{comp} represents energy attributed to CPU+GPU and does not include platform-level consumption that is not directly attributable (PSU losses, fans, storage, peripherals, or cooling). Therefore, results are interpreted as consistent relative comparisons within the same system and protocol.
- Average power: For the energy–time analysis, the average power was defined as:

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

  Average power was used as an aggregate descriptor of each run. It does not represent instantaneous power consumption and cannot distinguish the contributions of different execution phases, such as data loading, computation, communication, synchronization, or idle waiting. A phase-level power characterization would require higher temporal resolution measurements and is beyond the scope of this study.

3.6. Carbon emissions estimation (CO_2e)

Training-related emissions were reported as a deterministic function of measured energy. Specifically, for each run the following was computed:

Estimation based on measured energy:

$$CO_2e_{meas} = E_{comp}[kWh] \times IC \quad (7)$$

- where IC denotes the carbon intensity of the electrical grid ($\frac{kgCO^2e}{kWh}$). Because carbon intensity depends on both regional and temporal factors, CO_2 emissions were computed in a post-processing step using three sensitivity scenarios: 0.10, 0.20 y 0.30 $kgCO_2e/kWh$. The value of 0.20 $kgCO_2e/kWh$ was selected as the reference scenario because it represents the midpoint of the analyzed interval and facilitates comparative interpretation; it does not correspond to a direct measurement of the local grid carbon intensity during the experimental runs. Since CO_2e emissions are derived linearly from the measured energy consumption, varying the carbon intensity changes only the absolute magnitude of the estimated emissions and does not affect the relative ranking of the evaluated configurations.
- CodeCarbon estimation: Each run was instrumented with CodeCarbon to obtain CO_2e_{cc} and related estimates. Measured E_{comp} was compared against CodeCarbon-estimated energy, and the relative error was reported as:

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

3.7. Statistical analysis

For each run, the following were recorded: total time T, E_{cpu}, E_{gpu}, E_{comp}, average power $\bar{P}$, CO_2e (via measured energy and via CodeCarbon), and a learning sanity indicator ($\text{accuracy}_{\text{top1}}$ on the test split). The primary study measurements were $T$and E_{comp}; the remaining metrics are considered derived or descriptive. The $\text{accuracy}_{\text{top1}}$metric is reported only to verify that the fixed-budget training regime yields non-trivial learning and to rule out training failures/collapse; it is neither optimized nor used as a criterion to compare configurations. Accordingly, the inferential analysis focuses on $T$and E_{comp}. For CIFAR-10 (10 classes), $\text{accuracy}_{\text{top1}}$was checked to exceed the chance-level baseline (0.10) as a sanity control.

Results were aggregated per configuration as mean ± standard deviation ($\mu \pm \sigma$) over five repetitions. When applicable, 95% confidence intervals (CI95%) for the mean were estimated via bootstrap. To quantify the effects of factors $(G, P, C)$and their interactions on $T$and E_{comp}, a three-way factorial analysis (ANOVA/GLM with interaction terms) was used, together with post-hoc comparisons with multiple-testing correction (Holm), maintaining $\alpha = 0.05$.

4. RESULTS

This section reports the outcomes of the 60 runs corresponding to the 12 configurations of the factorial design, while keeping the compute budget ($N_{\text{steps}} = 250$) and the global batch size (512) constant to isolate the effects of the evaluated factors.

4.1 Overall results by configuration

Table 1 summarizes the configuration-level aggregated results at the end of the fixed compute budget. In this set, mean training times ranged from 91.33 s (2G-AMP-300W) to 313.75 s (1G-FP32-200W), whereas the component-attributed energy E_{comp}varied between 48,386 J (1G-AMP-200W) and 88,766 J (1G-FP32-300W). Figure 1 separately presents $T$and $E_{\text{comp}}$by configuration.

Table 1: *Descriptive results by configuration (μ ± σ, n=5).*

Config	T (s)	E_{cpu} (J)	E_{gpu} (J)	E_{comp} (J)	E_{comp} (kWh)	$\bar{P}$ (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

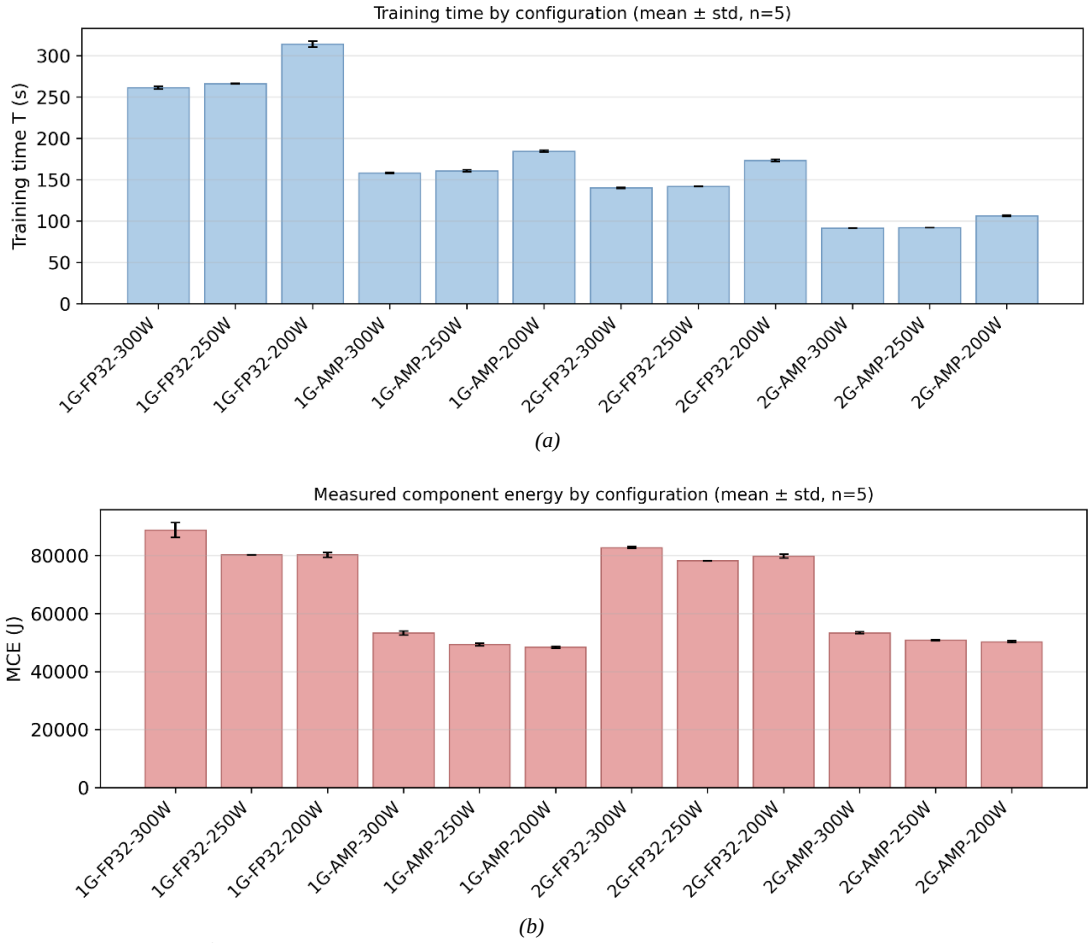


Fig. 1. *Configuration-level results: (a) total training time T and (b) component-attributed energy* E_{comp}.

4.2 Energy–time trade-off and operating region

Figure 2 summarizes the energy–time trade-off across the 12 evaluated configurations using configuration means ($n = 5$). A clear separation between FP32 and AMP is observed: AMP configurations concentrate energy values in the ~ 48–53kJ range, whereas FP32 configurations lie in ~ 78–89kJ, 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 high-energy and relatively high-time region, serving as a reference to quantify improvements.

The dashed trace denotes the Pareto frontier, i.e., the set of configurations for which no other configuration in the study can simultaneously reduce both energy and time. Only AMP configurations appear on this frontier, indicating that FP32 alternatives are dominated in the energy–time plane by mixed-precision options. The frontier extremes correspond to the lowest-time configuration (2G-AMP-300W) and the lowest-energy configuration (1G-AMP-200W), reflecting two distinct operational preferences: maximizing speed or minimizing consumption.

Between these extremes, an operating region (trade-off zone) is identified around 2G-AMP-250W, where energy is reduced with an almost negligible time penalty relative to the fastest point on the frontier. Specifically, relative to 2G-AMP-300W, the 250 W limit reduces energy by ~ 4–5%with a time increase of <1%, whereas lowering to 200 W yields only a small additional energy saving at a much larger time penalty. This behavior suggests that, on this system and under this protocol, the operating point near 250 W per GPU provides a practical balance between acceleration and energy sustainability.

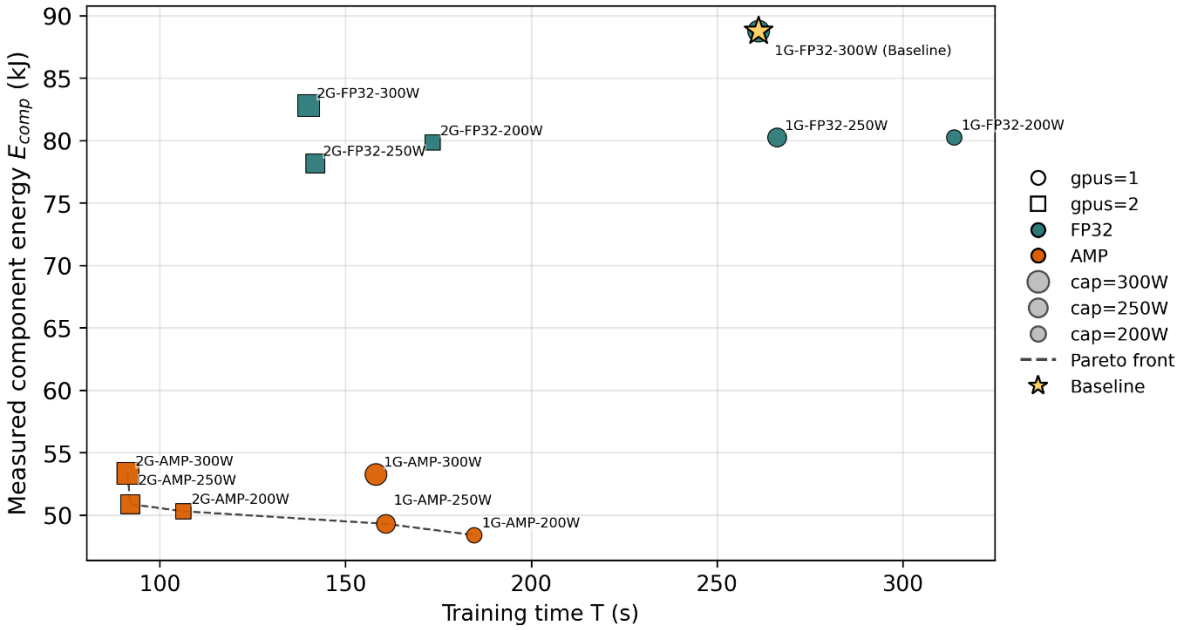


Fig. 2. *Energy–time trade-off at the configuration level (means; $n = 5$). The dashed line highlights the non-dominated configurations (Pareto frontier), and the star marks the baseline (1G-FP32-300W).*

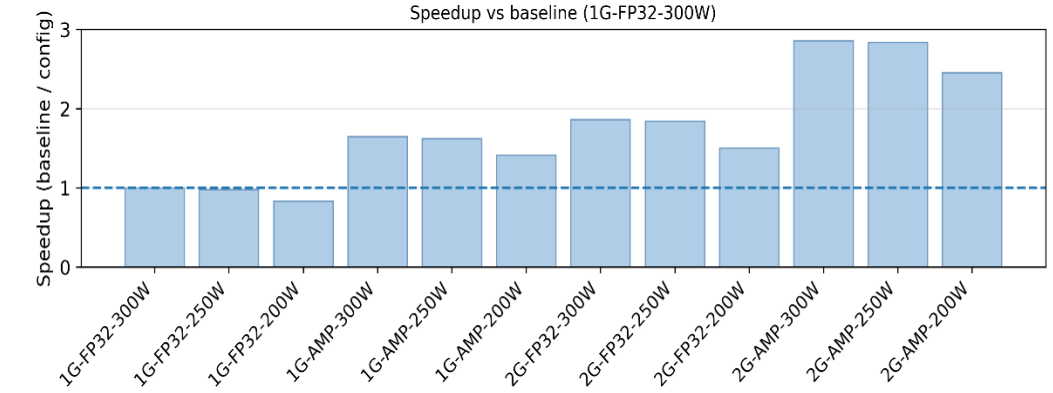
4.3 Effect of scaling vs. AMP vs. power capping

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

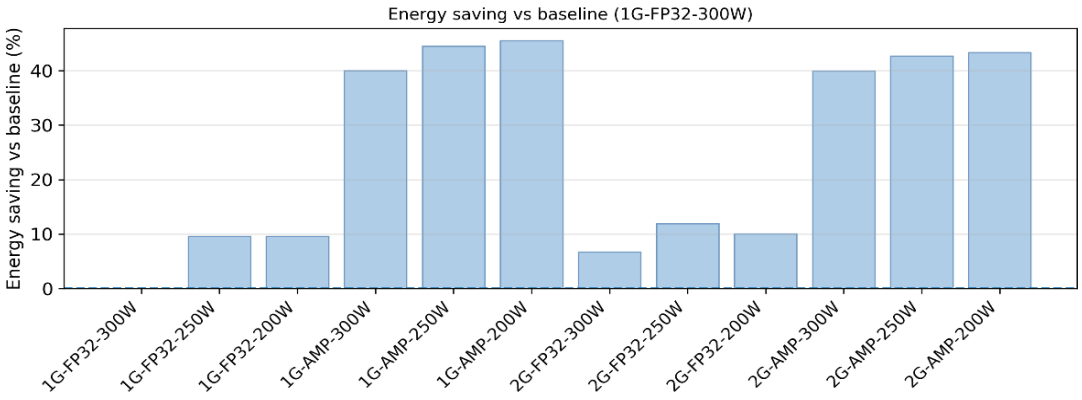
- GPU scaling (DDP): Under FP32, moving from 1 to 2 GPUs yielded a consistent speedup of 1.81×–1.88× (time reduction of ~ 44.7%–46.7%, depending on the cap) (see Figure 3a). In total measured energy E_{comp}, the effect was small, ranging from −0.5% (200W) to −6.7% (300W) (see Figure 3b). Thus, scaling clearly reduces time, but energy savings depend on whether the reduced duration compensates for the higher average power when two GPUs are active. Under AMP, scaling also accelerated training (speedup ~ 1.73 ×–1.75×; time reduction ~ 42%–43%), but it did not guarantee energy savings: E_{comp}increased slightly by +0.1% to +3.9% when moving to 2 GPUs (see Figure 3b). This suggests that, under the fixed-budget regime and on this machine, multi-GPU synchronization/execution overhead can erode the expected energy benefit even when training finishes sooner.
- A plausible explanation is that AMP decreases the duration of local computation, whereas DDP still requires gradient synchronization and collective communication operations managed by NCCL. As computation becomes faster, these synchronization overheads and waiting periods may account for a larger fraction of the total execution time. Combined with the additional power consumption of a second GPU, this effect may prevent the reduction in training time from translating into proportional energy savings. This interpretation is based on aggregate measurements, as the experimental protocol did not decompose energy consumption into computation, communication, and idle waiting phases.
- Effect of AMP: Holding the number of GPUs and the power cap fixed, AMP shifted the system into a clearly more efficient region: it reduced time by ~ 35%–41% and simultaneously reduced E_{comp} by ~ 35%–40% relative to FP32 (see Figure 3a–b). This pattern was stable for both 1 GPU and 2 GPUs and across all three caps. In other words, AMP provides a robust improvement in the energy–time trade-off (see Figure 3c), not merely acceleration.
- Effect of power capping: Power capping exhibited the expected trade-off: lowering the limit reduces energy but increases time. However, the effect is non-linear (see Figure 4).
    - Moving from 300W to 250W (with GPU count and precision fixed) reduced energy consistently (~ 4.6%–

9.6%) with a minimal time penalty (~ 0.7%–1.9%). Consequently, the energy–time trade-off improved (a ~ 3.9%–7.9% reduction in EDP_{rel} within the same group).

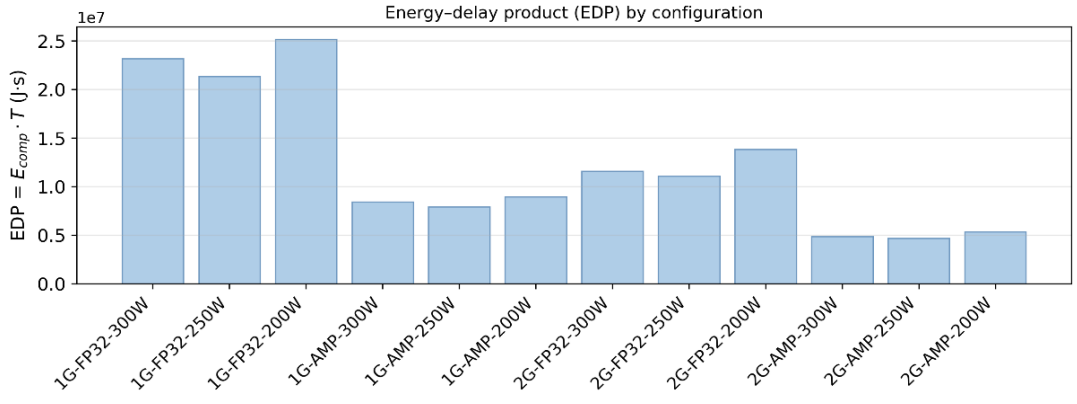
- Moving from 300W to 200W reduced energy (~ 3.6%–9.6%) but increased time markedly (~ 16%–24%), worsening the overall trade-off (EDP_{rel} increases by ~ 6%–19%). Therefore, 200W may be useful if the objective is to minimize energy unconditionally, but it is not the best option when seeking a practical energy–time balance.



(a)

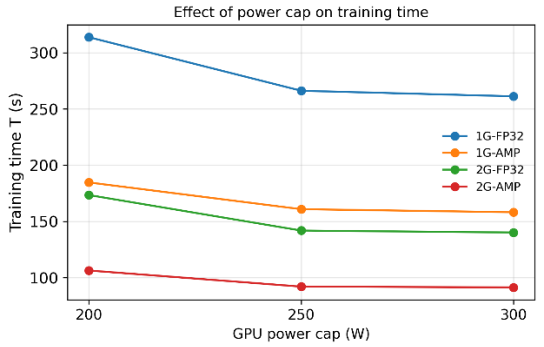


(b)

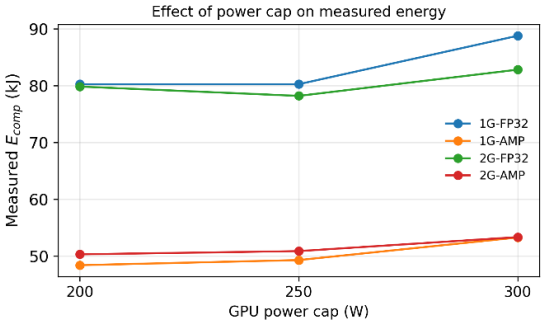


(c)

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).*



(a)



(b)

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

4.4 Consistency, variability, and CI95

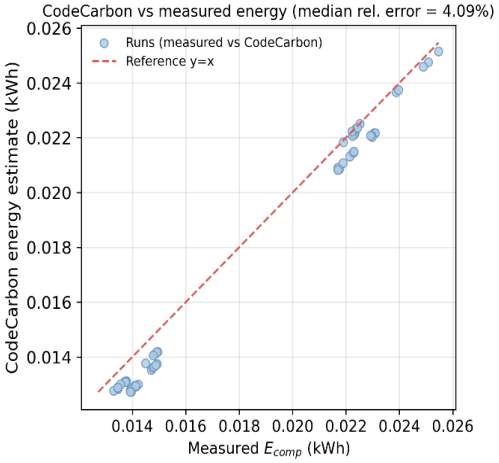
To verify measurement stability under the experimental protocol (five repetitions per configuration, varying only the random seed), intra-configuration variability was quantified for total training time $T$and measured energy E_{comp}. For each of the 12 configurations, the coefficient of variation $CV = 100 \cdot \sigma/\mu$and the standard deviation σwere computed from the five repetitions; these values were then summarized across all configurations. Results show low and consistent dispersion, indicating that the differences observed in the previous sections are primarily driven by the experimental factors (GPU count, AMP, and power cap) rather than by seed-level randomness (Table 2).

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

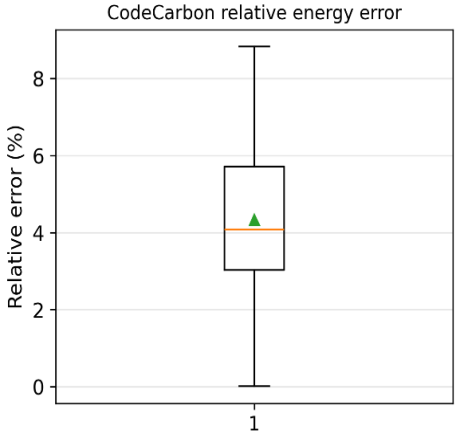
Variable	n	CV% (mediana $mín, máx$)	σ (mediana $mín, máx$)
Tiempo (T) (s)	5	0.559% [0.168, 1.135]	0.729 s [0.155, 3.560]
Energía (E_{comp}) (kJ)	5	0.646% [0.136, 2.849]	0.326 kJ [0.107, 2.529]

4.5. CodeCarbon vs. direct measurement

To assess agreement between a widely used estimator and direct measurement, CodeCarbon-estimated energy $E_{\text{cc}}$was compared against the component-attributed measured energy E_{comp}, both in kWh, at the run level ($N = 60$). The relationship between $E_{\text{cc}}$and $E_{\text{comp}}$is shown in Figure 5a, indicating that CodeCarbon reproduces the overall trend across configurations, albeit with a systematic deviation relative to direct measurement. To quantify this discrepancy, the relative error $\epsilon = |E_{\text{cc}} - E_{\text{comp}}|/E_{\text{comp}}$was computed and its distribution summarized in Figure 5b. CodeCarbon exhibited a median relative error of 4.09% (IQR: 3.03%–5.71%) and tended to underestimate measured energy (median bias $\approx -4.09\%$).



(a)



(b)

Fig. 5. *Comparison between CodeCarbon and direct energy measurement: (a) CodeCarbon-estimated energy vs. component-attributed measured energy, and (b) distribution of the relative error.*

4.6 Factorial analysis (ANOVA/GLM) and post-hoc

A three-way factorial model was fitted to quantify the effects of G(number of GPUs), P(precision:

FP32 vs. AMP), and C(power cap), including interactions, on $T$and E_{comp}. ANOVA results (Table 3) show that, for training time, the main effects of $G$and Pdominate the observed variation ($\eta_p^2 \approx 0.999$ for both), while Calso contributes a substantial effect ($\eta_p^2 = 0.993$). Interactions were significant, indicating that the impact of $C$and scaling depends on precision and GPU count (see Figure 6).

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

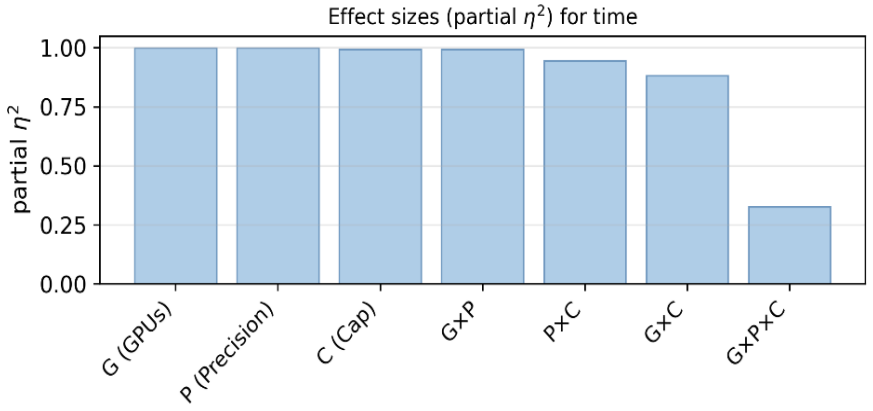


Fig. 6. *Effect sizes from the three-way factorial ANOVA (G×P×C) on training time T, including main effects and interactions.*

Table 3: *Three-way factorial ANOVA results for T and E_{comp}.*

Efecto	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

4.7 CO_2e sensitivity to carbon intensity (CI)

Since CO^2e_{meas} was computed in post-processing as a deterministic function of the measured energy consumption,($CO^2e_{meas} = Ecomp[kWh] \times IC$), its sensitivity to carbon intensity is strictly linear. As the value of IC increases, the estimated CO_2e emissions for all configurations increase proportionally. However, the relative ranking of the evaluated configurations remains unchanged, and the set of preferred configurations in the energy–time trade-off is unaffected. Figure 7 illustrates this behavior for carbon intensity values of 0.10, 0.20 y 0.30 $kgCO_2e/kWh$, while keeping the measured energy consumption constant for each experimental configuration.

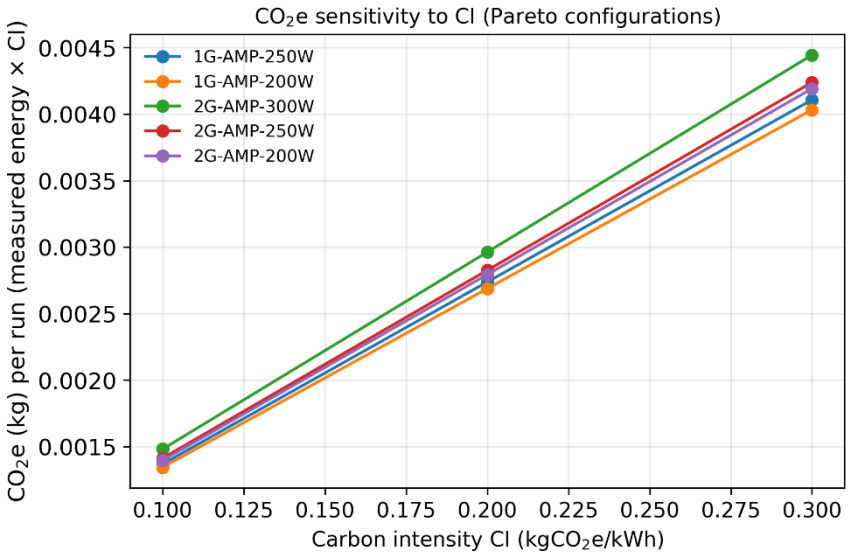


Fig. 7. *Sensitivity of estimated emissions to carbon intensity (CI).*

5. DISCUSSION

This work isolated the effects of scaling, precision, and power capping on the energy–time trade-off during training. Under these conditions, the central finding is that acceleration does not necessarily imply lower energy consumption, and that system-level adjustments such as mixed precision and power limits can be more decisive for efficiency than parallelism alone.

First, AMP was the dominant factor, as it shifted configurations toward substantially lower time and energy, clearly separating from the FP32 group in the energy–time plane. This separation explains why the Pareto frontier is composed exclusively of AMP configurations. That is, within the evaluated space, FP32 is dominated by AMP alternatives that are simultaneously faster and less energy-intensive. Practically, 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: under FP32, modest savings are observed, whereas under AMP, savings are not guaranteed and can even reverse. This reflects that activating two GPUs increases the system's average power level, and total energy decreases only if the reduction in duration compensates for this increase and for synchronization/communication overheads.

Finally, power capping exhibited a practical balance point. Reducing the cap from 300W to 250W lowered energy with a minimal time penalty, improving composite metrics such as EDP. In contrast, 200W introduced much larger time penalties, which tend to degrade 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 compromise between sustainability and productivity.

The comparison with CodeCarbon indicates that the tool reproduces trends but shows a systematic discrepancy relative to direct measurement (median relative error ~ 4%, with underestimation). Thus, CodeCarbon is useful for monitoring and broad comparisons, but for fine-grained analyses it is advisable to accompany it with direct measurement or to explicitly report its error. Finally, sensitivity to CI is strictly linear: it changes the CO_2e scale but does not alter the relative ordering across configurations.

The scope of these findings is constrained by the adopted experimental protocol, which consists of training a ResNet-50 model on the CIFAR-10 dataset using a workstation equipped with NVIDIA RTX A6000 GPUs under a fixed-budget partial training regime. Consequently, the magnitude of the observed effects should not be directly extrapolated to transformer architectures or foundation models, larger-scale datasets, alternative hardware accelerators, or systems comprising more than two GPUs, where the interactions among computation, memory, communication, execution time, and energy consumption may differ substantially.

Furthermore, the evaluated scaling scenario was limited to the transition from one to two GPUs within a single workstation. Therefore, this study does not characterize large-scale distributed or multi-node training, in which collective synchronization, inter-node communication, network topology and interconnect bandwidth, communication congestion, and the overlap between computation and communication may substantially alter the energy–time trade-off.

6. CONCLUSIONS

This work provides reproducible evidence, under a fixed compute budget, on how common operational decisions reshape the energy–time trade-off and, by extension, the carbon footprint associated with training. First, it confirms that multi-GPU scaling (from 1 to 2 GPUs) consistently accelerates training but does not guarantee energy savings; that is, time decreases markedly, whereas total CPU+GPU-attributed energy can change little or even increase depending on the operating regime. This reinforces a key point from an environmental perspective: finishing sooner reduces the system's exposure time, but 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 into a more efficient region by simultaneously reducing time and energy, making it a robust choice when the goal is to reduce consumption and emissions without sacrificing productivity. Power capping enables finer control of the balance: on this system, 250 W per GPU offers a favorable compromise by consistently reducing energy with minimal time penalties, whereas 200 W tends to impose disproportionate time penalties for marginal energy savings, making it more appropriate when the priority is strictly minimizing consumption even at the cost of longer duration.

Third, the time–energy frontier analysis provides a clear and comparable interpretation of configurations, separating extremes (minimum time vs. minimum energy) and identifying an intermediate operating region where the balance between acceleration and sustainability is most practical. This representation is particularly useful for guiding real decisions because it makes explicit that there is no universally best configuration, but rather optimal choices conditioned on the objective (speed, energy, or balance).

Finally, regarding emissions, this work reports CO_2e as a deterministic function of measured energy, enabling transparent interpretation of environmental impact: energy is the bridge between execution and climate, and the factor that converts kWh into CO_2e is the grid carbon intensity (CI). Under this formulation, changing CI does not alter the configuration ranking (it only scales emissions), but it does affect absolute impact, underscoring that training sustainability depends both on system efficiency and on regional energy context. Within this framing, CodeCarbon is useful for reproducing trends and facilitating comparable reporting, but in this experiment, it exhibited 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 complement CodeCarbon-style estimates with direct measurement or, at minimum, to explicitly report its error and bias, so that estimated CO_2e is not interpreted as an unquestionable absolute value but as an approximation with quantified uncertainty. Consequently, the findings should be interpreted within the scope of the experimental protocol adopted in this study and should not be regarded as a general characterization of training processes carried out to convergence or of larger-scale distributed systems.

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.