ISSN: 1692-7257 - Volumen 2 – Número 48 - 2026

ORIGINAL

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

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

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

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.**

Recibido: 24 marzo 2026. Revisado: 12 junio 2026. Aceptado: 10 julio 2026. Publicado: 15 julio 2026.
Autor de correspondencia: Carlos Nelson Henríquez Miranda

Cómo citar: 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

Esta obra está bajo una licencia internacional Creative Commons Atribución-NoComercial 4.0.

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.

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.

INTRODUCCIÓN

El entrenamiento de modelos profundos de visión se ha convertido en un componente central de múltiples aplicaciones científicas e industriales[1–3], habilitado por la adopción generalizada de aceleradores Graphics Processing Unit (GPU) y arquitecturas híbridas Central Processing Unit (CPU) + GPU tanto en estaciones de trabajo como en clústeres High-Performance Computing (HPC)[4]. La evidencia sobre tendencias históricas muestra que el cómputo de entrenamiento ha aumentado de forma acelerada [5] en la era del Deep Learning[6],[7], con tiempos de duplicación del orden de meses, y con una transición adicional hacia modelos large-scale[5]. Lo que plantea preocupaciones ambientales debido a su alto consumo energético[8]. En dominios intensivos en cómputo como la imagen médica por ejemplo, tareas sobre imagen de resonancia magnética (IRM) con Deep Learning[9], la presión por reducir tiempos de entrenamiento y mejorar el rendimiento suele resolverse incrementando paralelismo y recursos, con efectos directos sobre el consumo energético, los costos operativos, la demanda de enfriamiento y las emisiones asociadas. A escala HPC, se reportan consumos agregados de alta magnitud en sistemas líderes, evidenciando la necesidad de evaluar no solo el tiempo de ejecución, sino también la energía total requerida por distintas configuraciones de entrenamiento[4].

Al mismo tiempo, el aprendizaje profundo tiende a requerir incrementos sustantivos de cómputo para obtener mejoras marginales de desempeño, lo que vuelve relevante el costo energético de dichas ganancias y obliga a distinguir entre aceleración y eficiencia energética [10]. En este contexto, el escalamiento de GPU (pasar de 1 a múltiples GPU) se justifica con frecuencia por la reducción del tiempo de entrenamiento; sin embargo, la energía total (kWh) y su traducción a emisiones (^{kgCO_2e}) pueden no disminuir proporcionalmente. Sobrecargas de comunicación y sincronización, variaciones de utilización y consumo base del sistema pueden erosionar (o incluso revertir) los beneficios energéticos esperados, especialmente cuando el análisis se realiza de extremo a extremo e incluye CPU+GPU bajo supuestos comparables.

La literatura reciente ofrece contribuciones relevantes pero fragmentadas para entender este compromiso energía–tiempo. Por un lado, se ha estudiado el control energético en GPU mediante power capping y su efecto sobre métricas como Energy–Delay Product (EDP) o Energy–Delay Sum (EDS), con reducciones de consumo que dependen del régimen de cómputo y del criterio de optimización[4]. Por otro lado, se han popularizado herramientas y enfoques para rastrear o estimar energía y huella de carbono durante el entrenamiento, permitiendo comparar hardware y escenarios mediante factores regionales de emisión[10–12]. Sin embargo, también se ha señalado que la principal limitación de la comparabilidad y la trazabilidad entre estudios es debido a la heterogeneidad de supuestos[13], esto es qué se mide, con qué instrumento, y cómo se convierte a emisiones. En paralelo, desde HPC se han explorado modelos para predecir la potencia de trabajos antes de su ejecución, subrayando que el consumo observado depende tanto de la aplicación como del contexto operativo[14]. Complementariamente, nociones como energy bloat y fronteras tiempo–energía ayudan a separar consumo contributivo al throughput de consumos asociados a esperas, sincronización u otras ineficiencias que pueden distorsionar conclusiones si solo se atiende al tiempo[15].

A pesar de esto, persiste una debilidad práctica: falta evidencia directamente comparable que caracterice cómo el escalamiento de mono-GPU a multi-GPU modifica la relación energía–tiempo cuando se combinan prácticas comunes de eficiencia con mecanismos explícitos de control de limitación de potencia, bajo condiciones de comparación estrictas. En particular, muchos reportes no controlan la carga efectiva con un presupuesto fijo de cómputo ni delimitan con precisión el alcance de la medición, amplificando la heterogeneidad metodológica[13].

Este trabajo hace una exploración en la dirección de esa debilidad mediante una evaluación controlada del entrenamiento de un modelo de visión profunda (dataset: CIFAR-10, arquitectura: ResNet-50) bajo un presupuesto fijo de cómputo. Se estudia el efecto del escalamiento (de 1 a 2 GPU), la precisión (FP32 vs. AMP) y límites de potencia por GPU (300/250/200 W) sobre el compromiso energía–tiempo. La energía se mide a nivel de componentes combinando contadores Intel RAPL para CPU y energía acumulada por GPU vía NVIDIA NVML, y se estima el CO_2e con CodeCarbon [16] empleando una intensidad de carbono de 0.20 ($kgCO_2e$)/kWh (sensibilidad: 0.10–0.30)[10–12]. Con ello, este trabajo intenta aportar (i) evidencia reproducible sobre cuándo el escalamiento acelera sin necesariamente ahorrar energía, (ii) criterios prácticos para seleccionar configuraciones eficientes con AMP y power capping [4] y (iii) una lectura consistente del compromiso energía–rendimiento apoyado en el marco de fronteras tiempo–energía y en el reconocimiento de ineficiencias extremo a extremo[15]. Además, se discuten implicaciones para operación y planificación en entornos CPU+GPU, considerando el rol del contexto operativo en el consumo observado[14]. La organización del documento es, en primer lugar, se revisan los trabajos previos y la literatura relacionada con el compromiso entre energía, tiempo y emisiones en procesos de entrenamiento. Luego, se presentan los materiales y métodos, incluyendo la metodología experimental para analizar el escalamiento de mono-GPU a multi-GPU y su incidencia en el consumo energético y el tiempo de entrenamiento. Finalmente, en la sección de resultados se reportan las evaluaciones de cada comparación, junto con la discusión de los hallazgos, y se formulan las conclusiones.

TRABAJOS RELACIONADOS

Para estructurar la evidencia sobre el compromiso energía–tiempo y emisiones en entrenamiento, se ha agrupado la literatura en

dos grupos: (i) técnicas de optimización o control que modifican potencia, frecuencia, comunicaciones o asignación de recursos para mejorar eficiencia; (ii) métodos de medición o estimación y reporte que cuantifican E–T–CO_2e con instrumentación o modelos.

Optimización y control E–T–CO_2

Una forma práctica de reducir el impacto ambiental del entrenamiento en relación con la energía consumida y CO_2e asociado, es no asumir que más rápido implica más limpio. En la práctica, la energía total depende de cuánto tiempo dura el entrenamiento (T) y de cuánta potencia demanda el hardware mientras trabaja (P); por eso, muchas estrategias buscan encontrar configuraciones que reduzcan energía total (E) o que logren mejores compromisos entre energía y tiempo. Para comparaciones consistentes se usan métricas compuestas, como el producto energía–retardo ($EDP = E \cdot T$) y la suma energía–retardo ($EDS = E + T$).

En una primera línea, se ajusta cuánta potencia puede consumir la GPU (power capping) o a qué frecuencia opera (mediante escalamiento dinámico de voltaje y frecuencia, Dynamic Voltage and Frequency Scaling, DVFS). Con mediciones basadas en la NVIDIA Management Library (NVML) se reportan ahorros de energía de hasta 30% con penalizaciones de rendimiento menores al 12%; además, se evalúa la calidad de la medición al compararla con otras interfaces como Performance Application Programming Interface (PAPI) usando métricas de error (MAPE 6,39% y RMSE 3,97%) [4]. De manera complementaria, ajustar la frecuencia del núcleo (f_{core}) con DVFS, guiándose por cómo cambia el rendimiento frente a la potencia, permite ahorrar $8,7\% - 23,1\%$ de energía en entrenamiento de redes neuronales profundas (y $19,6\% - 26,4\%$ en inferencia), e incluso reporta mejoras de rendimiento de hasta 33% frente a valores por defecto [17]. En esa misma lógica de control en tiempo real, Dynamic Energy-Performance Optimiser (DEPO) ajusta en línea límites de potencia usando NVML y la interfaz CUDA Profiling Tools Interface (CUPTI) para optimizar un objetivo tipo $\min(\alpha E + \beta T)$; reporta reducciones >22% en energía (E) con caídas <20% cuando se prioriza energía, y ahorros de 15%–18% con <8% de penalización cuando se prioriza el compromiso (EDP/EDS) [4].

En una segunda línea, el foco está en que al entrenar con varias GPU aparece consumo extra que no siempre aporta aprendizaje: esperas, sincronización y comunicación. Esto puede aumentar la energía total, aunque el entrenamiento termine antes. Perseus [15] aborda este fenómeno energy bloat construyendo un frente de Pareto tiempo–energía del trabajo, y reporta ahorros de hasta 30% en GPT-3 y Bloom sin pérdida de rendimiento efectivo (throughput). PowerFlow [18] propone un modelo simple donde el consumo depende del número de GPU, el tamaño de minibatch local y la frecuencia; luego asigna recursos para maximizar eficiencia energética sin exceder un presupuesto, mejorando el tiempo de finalización entre 1,57× y 3,39× frente a baselines con el mismo consumo[18]. También hay evidencia de que aumentar el minibatch puede mejorar el rendimiento por hora de energía: un resultado representativo muestra ResNet-50 en ImageNet con minibatch 8192 entrenado en 1 hora usando 256 GPU (~90% de eficiencia) con escalado lineal del learning rate y warmup, sin pérdida de precisión[19]. En modelos de lenguaje grandes Large Language Models (LLM) , donde la comunicación colectiva puede dominar, PCCL ajusta la frecuencia durante comunicación para cumplir un ancho de banda mínimo, reduciendo hasta 27% la energía de estas operaciones y 17,3% la energía total de entrenamiento, con efecto despreciable en el rendimiento[20].

A escala de sistema, un análisis sugiere que combinar buenas prácticas (mejoras de modelo, elección de hardware, automatización y mezcla energética) podría reducir emisiones entre 65× y 747×; sin embargo, enfatiza limitaciones prácticas: falta de datos públicos de Power Usage Effectiveness (PUE) y de intensidad de carbono (IC), y dificultades para integrar carbono en control en línea[21]. En esa dirección, Zeus [22] propone un enfoque justo a tiempo que explora configuraciones (tamaño de batch y límite de potencia) y construye un frente Pareto energía–tiempo por trabajo; reporta mejoras de 15,3% a 75,8% en eficiencia energética, aunque con sobrecarga de exploración y retos para su uso en clústeres multiusuario.

De este grupo de trabajos se infiere que evaluar con métricas compuestas (como EDP/EDS) ayuda a evitar conclusiones incompletas basadas solo en terminar antes o solo en gastar menos, cuando el objetivo real es balancear desempeño y sostenibilidad[4].

Medición y reporte E–T–CO_2

Este grupo de trabajos cuantifica y reporta el impacto ambiental del cómputo, es decir se enfoca en medir y comunicar de forma transparente cuánta energía se consume, cuánto tiempo tarda el entrenamiento y qué emisiones de CO_2e se asocian a esa energía. La idea central es que, sin medición consistente dos entrenamientos pueden reportar menos CO_2 solo porque usaron supuestos distintos.

Una primera línea busca anticipar el consumo antes de ejecutar un trabajo en HPC, útil para planificación. Aquí se propone predecir la potencia mínima/media/máxima de un job usando únicamente información disponible al momento del envío, con modelos online (reentrenamiento periódico) y normalización por número de nodos (por ejemplo, dividir la potencia promedio entre los nodos asignados), logrando errores menores al 12% en Fugaku [23] y al 22% en Marconi100[14]. Este enfoque ayuda a asignar recursos y presupuestos energéticos, pero para estimar energía total necesita complementarse con modelos

de duración; además, queda abierta la integración de mecanismos robustos frente a la deriva y la generación de intervalos predictivos que representen la incertidumbre[14].

La segunda línea mide y reporta durante el entrenamiento mediante instrumentación. EnergyVis [11] captura consumo en vivo usando contadores de CPU (Intel Running Average Power Limit, RAPL) y lecturas de GPU (NVIDIA SMI), y presenta energía por época integrando potencia en el tiempo $(E_{epoch} = \int P(t)dt)$ para visualizar kWh y CO_2 estimado. Su extensión incorpora modos precargado y live-tracking, además de herramientas para comparar hardware y extrapolar resultados. En un enfoque similar de seguimiento, Carbontracker [10] estima energía agregada incorporando Power Usage Effectiveness (PUE) a partir de lecturas NVML/RAPL, mientras eco2AI mide GPU/CPU/RAM y convierte a CO_2e usando coeficientes regionales y PUE [12]. También existen calculadoras basadas en localización y hardware que muestran variaciones amplias del factor de emisión y proponen mitigaciones, aunque sin medición directa ni actualización continua [24].

Un resultado clave para comparabilidad es que distintos supuestos pueden cambiar sustancialmente las conclusiones, una comparación en reconocimiento de entidades nombradas (NER) reportó variaciones de hasta 40% por discrepancias en PUE, factores de carbono y método de medición, lo que refuerza la necesidad de declarar explícitamente los supuestos[13]. En esa misma dirección, se han propuesto marcos de estimación para modelos grandes (por ejemplo, T5 [25] y GPT-3[26]) y recomendaciones explícitas de transparencia sobre PUE y mezcla energética[27], además de guías de reporte que enfatizan sensibilidad a hiperparámetros e independencia de hardware[28]. A nivel empírico, mediciones masivas con vatímetros muestran que el consumo no se explica bien solo por FLOPs o por el número de parámetros, y evidencian un comportamiento ligado a jerarquía de memoria (un modelo por tramos con umbral asociado a capacidad de caché)[29]. En CNN clásicas, también se observa relación entre complejidad arquitectural y energía/emisiones, y se propone una métrica simple tipo Score = Accuracy / Energy, validando profilers como alternativa económica frente a vatímetros por su alta correlación[30].

El valor de los reportes E–T–CO_2 depende de alinear definiciones (qué se mide y cómo se convierte), y de conectar métricas micro (componentes, fases del pipeline) y con métricas macro (kWh y CO_2e), porque de lo contrario la eficiencia puede ser más un artefacto metodológico que un resultado real[13],[29].

MATERIALES Y MÉTODOS

El objetivo de esta experimentación es la evaluación controlada del entrenamiento de un modelo de visión profunda bajo el supuesto fijo de cómputo. Los experimentos se ejecutaron en una estación de trabajo con dos GPUs NVIDIA RTX A6000 (49,140 MiB por GPU; límite de potencia 300 W), Intel Xeon Gold 6230R (26 cores, 52 threads) y 256 GB de RAM, bajo Ubuntu 22.04.5 LTS con Linux (kernel 6.8.0-94-generic) y controlador NVIDIA 560.35.05. El entrenamiento se implementó en PyTorch (versión 2.8.0+cu126), utilizando CUDA 12.6 y cuDNN 91002. Para asegurar comparabilidad entre configuraciones, todas las corridas se ejecutaron en el mismo equipo y bajo un protocolo reproducible.

Dataset y modelo de Deep learning

Para el estudio se seleccionó un escenario estándar y reproducible de visión artificial basado en CIFAR-10 (50,000 imágenes de entrenamiento y 10,000 de prueba), utilizando un modelo representativo de entrenamiento profundo ResNet-50. Para incrementar la carga computacional y hacer relevante el escalamiento en GPU, las imágenes se reescalaron a 224×224 y se aplicaron transformaciones de entrenamiento consistentes, manteniendo constante el pipeline de datos en todas las configuraciones.

Presupuesto fijo de cómputo y control de variabilidad

Para reducir variabilidad entre corridas, todas las configuraciones se entrenaron con un presupuesto fijo de cómputo definido como un número constante de pasos de optimización $N_{steps} = 250$. Se mantuvo un tamaño de lote global constante de 512 muestras por paso. En modo multi-GPU, se utilizó paralelismo de datos y el lote se dividió uniformemente por GPU (256 por GPU), preservando el lote global. Se ejecutaron cinco repeticiones por configuración, variando únicamente la semilla aleatoria (seeds) y manteniendo constantes todos los hiperparámetros y el presupuesto computacional. Para cada corrida se fijaron seeds en Python/NumPy/PyTorch, y se definió un esquema de muestreo reproducible del dataloader, incluyendo seed por worker, con el fin de minimizar variaciones por aleatoriedad.

El presupuesto fijo de $N_{steps} = 250$ se adoptó como control experimental para comparar todas las configuraciones con el mismo número de actualizaciones y de muestras procesadas, no para entrenar el modelo hasta convergencia. Por tanto, los resultados describen la eficiencia energética y temporal durante una fase parcial de entrenamiento y no necesariamente el consumo de un entrenamiento completo. Al mantener constantes el lote global y el número de pasos, cada corrida procesó 128.000 muestras (250 × 512), con el lote distribuido uniformemente entre las GPUs activas.

Se empleó un diseño factorial controlado para evaluar el impacto del escalamiento de GPU sobre la relación energía–

tiempo y el efecto de mecanismos de eficiencia a nivel de sistema:

Diseño experimental

- Escalamiento de GPU (G): 1 GPU vs 2 GPUs, usando Distributed Data Parallel (DDP).
- Precisión numérica (P): Se usó FP32 y AMP (mixed precision).
- Límite de potencia por GPU (C): Se varió entre 300 W, 250 W, y 200 W.

Esto produce 12 configuraciones (2×2×3). Cada configuración se repitió 5 veces, para un total de 60 corridas. Las consideraciones fueron:

- Implementación multi-GPU: El entrenamiento multi-GPU se implementó con PyTorch DDP y backend NCCL, utilizando torchrun y sincronización estándar de gradientes. Se mantuvieron constantes los parámetros del optimizador, se utilizó SGD (lr=0.1, momentum=0.9, weight_decay=1e−4) con Cross Entropy Loss, sin scheduler, con aumentaciones consistente en Resize + RandomCrop con padding + flip y normalización.
- Mixed precision (AMP): En las configuraciones AMP se utilizó autocast y escalado de gradiente (por ejemplo, torch.cuda.amp.autocast + GradScaler), manteniendo idénticos los demás hiperparámetros.
- Power capping: Antes de cada corrida se fijó el límite de potencia de cada GPU activa mediante nvidia-smi -pl, con el valor correspondiente a la configuración (300, 250 o 200 W). La confirmación devuelta por el comando se tomó como verificación operativa de que el controlador había aceptado el límite solicitado. Esta comprobación no constituye una validación eléctrica independiente de la potencia instantánea.

Medición de tiempo

El tiempo total de entrenamiento T se midió como el intervalo transcurrido desde el inicio del primer paso hasta la finalización del paso N_{steps}, excluyendo una corrida de calentamiento (warm-up) no reportada. Para consistencia, se utilizó reloj monotónico (time.monotonic() en Python) y se registró el tiempo en segundos.

Medición de energía por software (CPU + GPU)

Dado que no se utilizó instrumentación eléctrica externa, la energía se cuantificó a nivel de componentes mediante contadores soportados por el sistema, con lecturas antes y después de cada corrida.

Energía de CPU (Intel RAPL): Se utilizó la interfaz powercap de Linux leyendo energía acumulada del paquete de CPU desde /sys/class/powercap/intel-rapl:0/energy_uj. Para cada corrida se registró E_{cpu}^{pre} y E_{cpu}^{post} (en μJ) y se calculó:

$$E_{cpu} = \frac{E_{cpu}^{post} - E_{cpu}^{pre}}{10^6} \tag{1}$$

Si ocurre wrap-around del contador, se corrige usando max_energy_range_uj (registrado al inicio del experimento).

Energía de GPU (NVIDIA NVML): Se midió energía acumulada por GPU mediante NVML, usando el contador NVMLDeviceGetTotalEnergyConsumption (mJ). Para cada GPU i:

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

La energía total de GPU se define como:

$$E_{\text{gpu}} = \sum_i E_{\text{gpu}_i} \tag{3}$$

Energía total atribuida a componentes se define como:

$$E_{\text{comp}} = E_{\text{cpu}} + E_{\text{gpu}} \tag{4}$$

y se convierte a kWh para reporte:

$$E_{\text{comp}}[\text{kWh}] = \frac{E_{\text{comp}}[\text{J}]}{3.6 \times 10^6} \tag{5}$$

Limitación explícita (validez): E_{comp} representa energía atribuida a CPU+GPU y no incluye consumos de plataforma no atribuibles directamente (pérdidas de PSU, ventiladores, almacenamiento, periféricos ni refrigeración). Por tanto, los resultados se interpretan como comparaciones relativas consistentes dentro del mismo sistema y protocolo.

- Potencia promedio: Para análisis energía–tiempo se define potencia promedio:

La potencia promedio se empleó como descriptor agregado de cada corrida. No representa la potencia instantánea ni permite distinguir variaciones entre entrada/salida, cómputo, comunicación, sincronización y espera; una caracterización por fase requeriría medición temporal de mayor resolución y queda fuera del alcance del estudio.

Estimación de emisiones de carbono (CO_2e)

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

Las emisiones asociadas al entrenamiento se reportaron como una función determinista de la energía medida. En particular, para cada corrida se calculó:

Estimación basada en energía medida:

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

donde IC es la intensidad de carbono de la red eléctrica ($\frac{kgCO^2e}{kWh}$). Debido a su dependencia regional y temporal, las emisiones se calcularon en posproceso mediante tres escenarios de sensibilidad: 0.10, 0.20 y 0.30 $kgCO_2e/kWh$. El valor 0.20 $kgCO_2e/kWh$ se seleccionó como escenario central por ser el punto medio del intervalo analizado y facilitar la interpretación comparativa; no representa una medición local instantánea durante las corridas. Como CO_2e se deriva linealmente de la energía medida, cambiar IC modifica la magnitud absoluta de las emisiones, pero no el orden relativo de las configuraciones.

Estimación con CodeCarbon: Cada corrida se instrumentó con CodeCarbon para obtener CO_2e_{cc} y estimaciones asociadas. Se comparó E_{comp} vs la energía estimada por CodeCarbon y se reportó el error relativo:

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

Análisis estadístico

Para cada corrida se registraron: tiempo total T, E_{cpu}, E_{gpu}, E_{comp}, potencia promedio $\overline{P}$, CO_2e (vía energía medida y vía CodeCarbon) y un indicador de sanidad del aprendizaje ($accuracy_{top1}$ en el split de test). Las mediciones primarias del estudio fueron T y E_{comp}; el resto de las métricas se considera derivado o descriptivo. La métrica $accuracy_{top1}$ se reporta únicamente para verificar que el régimen de entrenamiento bajo presupuesto fijo produce aprendizaje no trivial y para descartar fallos/colapsos de entrenamiento; no se optimiza ni se utiliza como criterio para comparar configuraciones. Por tanto, el análisis inferencial se centra en T y E_{comp}. Para CIFAR-10 (10 clases), se verificó que $accuracy_{top1}$ excediera el nivel esperado por azar (0.10) como control de sanidad.

Los resultados se agregaron por configuración como media ± desviación estándar ($\mu \pm \sigma$) a partir de cinco repeticiones. Cuando aplica, se reportaron los intervalos de confianza al 95% (IC95%) de la media estimados mediante bootstrap. Para cuantificar el efecto de los factores (G, P, C) y sus interacciones sobre T y E_{comp}, se empleó un análisis factorial de tres vías (ANOVA/GLM con términos de interacción), junto con comparaciones post-hoc con corrección por múltiples pruebas (Holm), manteniendo $\alpha = 0.05$.

RESULTADOS

Se presentan los resultados de las 60 corridas correspondientes a las 12 configuraciones del diseño factorial, manteniendo constante el presupuesto de cómputo ($N_{\text{steps}} = 250$) y el lote global (512) para aislar el efecto de los factores evaluados.

Resultados generales por configuración

La Tabla 1 resume los resultados agregados por configuración al finalizar el presupuesto fijo de cómputo. En este conjunto, los tiempos promedio abarcaron desde 91.33 s (2G-AMP-300W) hasta 313.75 s (1G-FP32-200W), mientras que la energía atribuida a componentes E_{comp} varió entre 48,386 J (1G-AMP-200W) y 88,766 J (1G-FP32-300W). La Figura 1 muestra de forma separada T y E_{comp} por configuración.

Tabla 1: Resultados descriptivos por configuración ($\mu \pm \sigma$, 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

C

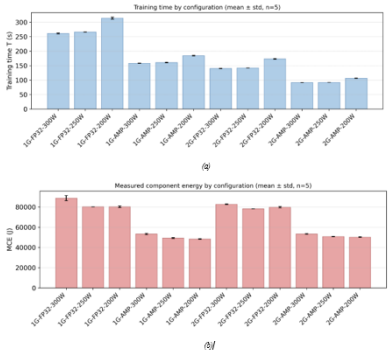

Fig. 1. Resultados por configuración, (a) tiempo total de entrenamiento T y (b) energía atribuida a componentes E_{comp}.

Compromiso energía–tiempo y región operativa

La Figura 2 resume el compromiso energía–tiempo para las 12 configuraciones evaluadas, usando las medias por configuración ($n = 5$). Se observa una separación marcada entre FP32 y AMP: las configuraciones con AMP concentran valores de energía en el rango $\sim 48 - 53\, kJ$, mientras que las de FP32 se ubican en $\sim 78 - 89\, kJ$, indicando que, bajo el mismo presupuesto de cómputo, la precisión mixta desplaza el entrenamiento hacia una región energéticamente más eficiente. El baseline (1G-FP32-300W) se ubica en la zona de mayor energía y mayor tiempo relativo, sirviendo como referencia para cuantificar mejoras.

El trazo punteado muestra el frente de Pareto, es decir, el conjunto de configuraciones donde no es posible reducir simultáneamente energía y tiempo mediante otra configuración del estudio. En este frente aparecen únicamente configuraciones con AMP, reflejando que las alternativas FP32 quedan dominadas en el plano energía–tiempo por opciones con precisión mixta. En los extremos del frente se encuentran la configuración de menor tiempo (2G-AMP-300W) y la de menor energía (1G-AMP-200W), que representan dos preferencias operativas distintas: maximizar velocidad o minimizar consumo.

Entre ambos extremos se identifica una región operativa (zona de compromiso) alrededor de 2G-AMP-250W, donde se logra una reducción de energía con una penalización temporal prácticamente despreciable respecto al punto más rápido del frente. En particular, frente a 2G-AMP-300W, el límite de 250 W reduce energía en $\sim 4 - 5\%$ con un aumento de tiempo $< 1\%$, mientras que bajar a 200 W aporta un ahorro energético adicional pequeño con una penalización temporal mucho mayor. Este comportamiento sugiere que, en este sistema y bajo este protocolo, el punto cercano a 250 W por GPU ofrece un balance práctico entre aceleración y sostenibilidad energética.

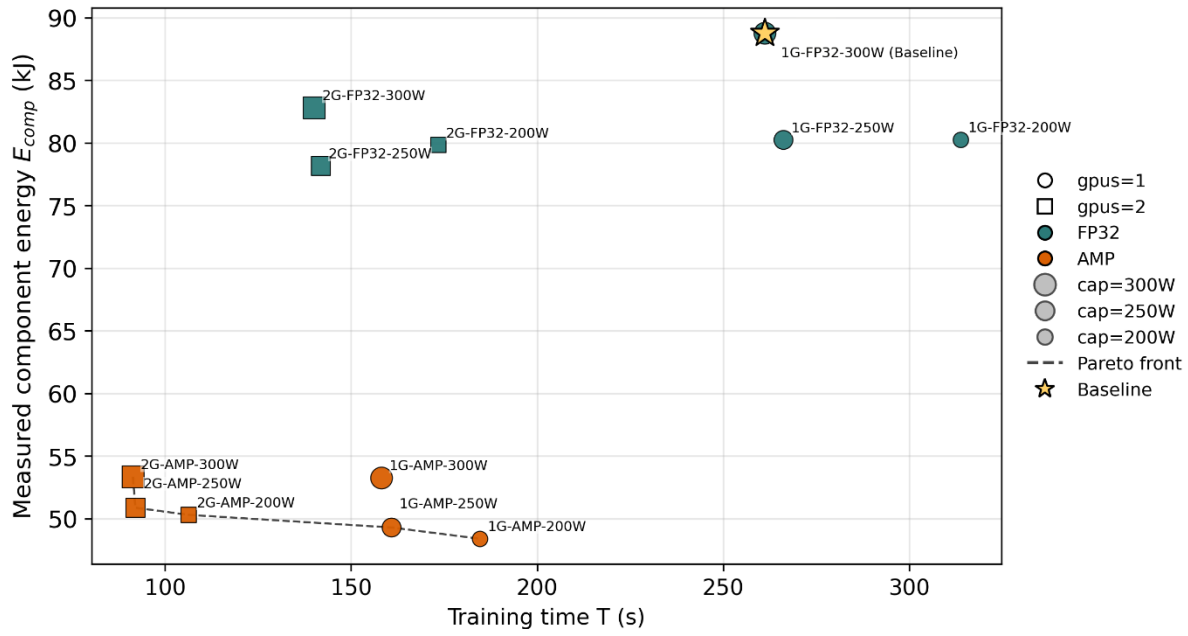

Fig. 2. *Compromiso energía–tiempo a nivel de configuración (medias; n=5). La línea punteada resalta las configuraciones no dominadas (frente de Pareto) y la estrella marca el baseline (1G-FP32-300W).*

Efecto de escalamiento vs AMP vs power cap

Para comparar los factores de forma directa, se usó como baseline la configuración 1G–FP32–300W y se reportaron métricas relativas: speedup $S = T_{\text{base}}/T$, cambio relativo de energía $\Delta E = 1 - E/E_{\text{base}}$, y métricas compuestas normalizadas para el compromiso energía–tiempo: $EDP_{\text{rel}} = (E \cdot T)/(E_{\text{base}} \cdot T_{\text{base}})$ y $EDS_{\text{rel}} = (E/E_{\text{base}}) + (T/T_{\text{base}})$. Estas medidas permiten separar cuándo una técnica acelera y cuándo realmente mejora eficiencia (ver Figura 3).

- Escalamiento de GPU (DDP): En FP32, pasar de 1 a 2 GPU produjo un speedup consistente de $1.81\times - 1.88\times$ (reducción de tiempo de $\sim 44.7\% - 46.7\%$, según el cap) (ver Figura 3a). En energía total medida E_{comp}, el efecto fue pequeño: desde -0.5% (200W) hasta -6.7% (300W) (ver Figura 3b). Es decir, el escalamiento reduce claramente el tiempo, pero el ahorro energético depende de si la reducción de duración compensa el mayor nivel de potencia promedio al activar dos GPU. En AMP, el escalamiento también aceleró (speedup $\sim 1.73\times - 1.75\times$; reducción de tiempo $\sim 42\% - 43\%$), pero no garantizó ahorro de energía: E_{comp} aumentó levemente entre $+0.1\%$ y $+3.9\%$ al pasar a 2 GPU (ver Figura 3b). Esto sugiere que, bajo este presupuesto fijo y en este equipo, la sobrecarga asociada a sincronización/ejecución multi-GPU puede erosionar el beneficio energético incluso cuando el entrenamiento termina antes.
- Una explicación plausible es que AMP reduce el tiempo de cómputo local, mientras que DDP mantiene la sincronización de gradientes y las operaciones colectivas gestionadas por NCCL. Al acelerarse el cómputo, estas sobrecargas y los periodos de espera pueden adquirir mayor peso relativo; junto con la potencia adicional de una segunda GPU, ello puede impedir que la reducción temporal produzca un ahorro energético proporcional. Esta interpretación se basa en métricas agregadas, pues el protocolo no descompuso la energía entre cómputo, comunicación y espera.
- Efecto de AMP: Manteniendo fijo el número de GPU y el power cap, AMP desplazó el sistema hacia una región claramente más eficiente: redujo el tiempo entre $\sim 35\%$ y $\sim 41\%$ y, simultáneamente, redujo la energía E_{comp} entre $\sim 35\%$ y $\sim 40\%$ frente a FP32 (ver Figura 3a-b). Este patrón fue estable tanto en 1 GPU como en 2 GPU y a través de los tres caps. En otras palabras, AMP aporta una mejora robusta del compromiso energía–tiempo (ver Figura 3c), no solo una aceleración.
- Efecto de power capping: El power cap mostró el trade-off esperado: al reducir el límite, baja la energía, pero aumenta el tiempo. Sin embargo, el efecto no es lineal (ver Figura 4).
- Al pasar de 300W a 250W (con GPU y precisión fijas), la energía bajó de forma consistente (~4.6%–9.6%) con una penalización temporal mínima $(\sim 0.7\% - 1.9\%)$. En consecuencia, el compromiso energía–tiempo mejora (reducción de EDP_{rel} de $\sim 3.9\% - 7.9\%$ dentro del mismo grupo).
- Al pasar de 300W a 200W, aunque la energía disminuyó ($\sim 3.6\% - 9.6\%$), el tiempo aumentó de forma marcada ($\sim 16\% - 24\%$), lo que empeora el compromiso global (EDP_{rel} aumenta $\sim 6\% - 19\%$). Por tanto, 200W puede ser útil si el objetivo es minimizar energía de forma incondicional, pero no es la mejor opción cuando se busca un balance práctico energía–tiempo.

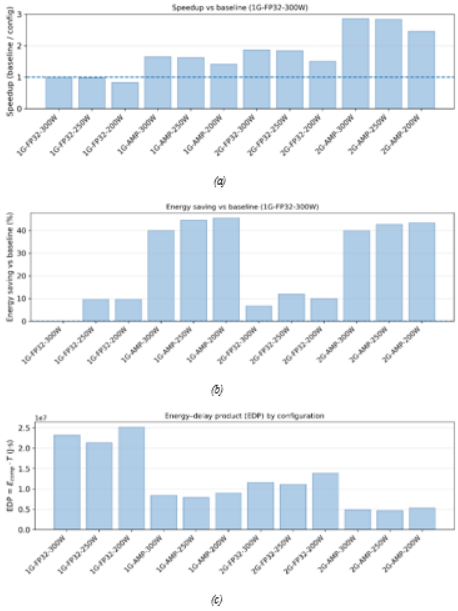

Fig. 3. *Impacto combinado de escalamiento, AMP y power capping relativo al baseline, en a) speedup, b) ahorro energético relativo y c) producto energía-retardo normalizado EDP.*

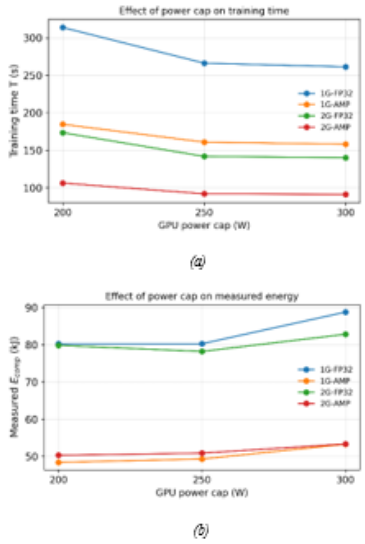

Fig. 4. *Efecto del power capping sobre el tiempo de entrenamiento. a) efecto en el tiempo y b) efecto en la energía medida.*

Consistencia, variabilidad y CI95

Con el fin de verificar la estabilidad de las mediciones bajo el protocolo experimental (5 repeticiones por configuración, variando únicamente la semilla), se cuantificó la variabilidad intra-configuración para el tiempo total de entrenamiento T y la energía medida E_{comp}. Para cada una de las 12 configuraciones, se calculó el coeficiente de variación $CV = 100 \cdot \sigma / \mu$ y la desviación estándar (σ) a partir de las cinco repeticiones; luego, estos valores se resumieron sobre todas las configuraciones. Los resultados muestran una dispersión baja y consistente, indicando que las diferencias observadas en las secciones anteriores están dominadas por los factores experimentales (GPU, AMP y power cap) y no por la aleatoriedad entre semillas (Tabla 2).

Tabla 2: Variabilidad intra-configuración (n=5 repeticiones por configuración).

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]

Comparación CodeCarbon y medición directa

Para evaluar la concordancia entre una estimación ampliamente usada y la medición directa, se comparó la energía estimada por CodeCarbon E_{cc} con la energía medida atribuida a componentes E_{comp}, ambas en kWh, a nivel de corrida ($N = 60$). La relación E_{cc} vs. E_{comp} se muestra en la Figura 5a, donde se observa que CodeCarbon reproduce la tendencia global entre configuraciones, aunque con una desviación sistemática respecto a la medición directa. Para cuantificar esta discrepancia, se calculó el error relativo $\varepsilon = |E_{cc} - E_{comp}| / E_{comp}$ y se resumió su distribución en la Figura 5b. CodeCarbon presentó un error relativo mediano de 4.09% (IQR: 3.03%–5.71%) y tendió a subestimar la energía medida (sesgo mediano $\approx -4.09\%$).

Análisis factorial (ANOVA/GLM) y post-hoc

Se ajustó un modelo factorial de tres vías para cuantificar el efecto de G (número de GPU), P (precisión: FP32 vs AMP) y C (power cap), incluyendo interacciones, sobre T y E_comp. Los resultados del ANOVA (Tabla 3) muestran que, para el tiempo de entrenamiento, los efectos principales de G y P dominan la variación observada ($\eta_p^2 \approx 0.999$ en ambos casos), mientras que C también aporta un efecto considerable ($\eta_p^2 = 0.993$). Las interacciones resultaron significativas, indicando que el impacto de C y del escalamiento depende de la precisión y del número de GPU (ver Figura 6).

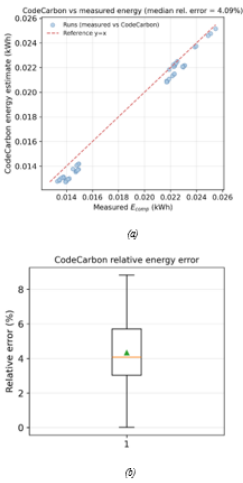

Fig. 5. *Comparación entre CodeCarbon y medición directa de energía, (a) Energía estimada por CodeCarbon vs. energía medida atribuida a componentes, (b) Distribución del error relativo.*

Para la energía medida E_{comp}, el factor dominante fue P (AMP) ($\eta_p^2 = 0.998$), seguido por C ($\eta_p^2 = 0.902$), mientras que el efecto directo de G fue menor pero estadísticamente significativo (η²_p=0.224), véase Tabla 3. Adicionalmente, las interacciones (en especial $G \times P$ y $G \times C$) fueron relevantes, lo que confirma que los cambios energéticos por escalamiento y por power capping no son aditivos y dependen del régimen operativo. Finalmente, las comparaciones post-hoc (Holm) restringidas a C dentro de cada combinación (G, P) confirmaron que 250W reduce E_{comp} frente a 300W en todos los casos ($p_Holm < 0.01$); la reducción adicional a 200W fue consistente bajo AMP, pero no necesariamente bajo FP32 (por ejemplo, con 2 GPU en FP32, 200W incrementó ligeramente E_{comp} frente a 250W, $p_{Holm} < 0.01$).

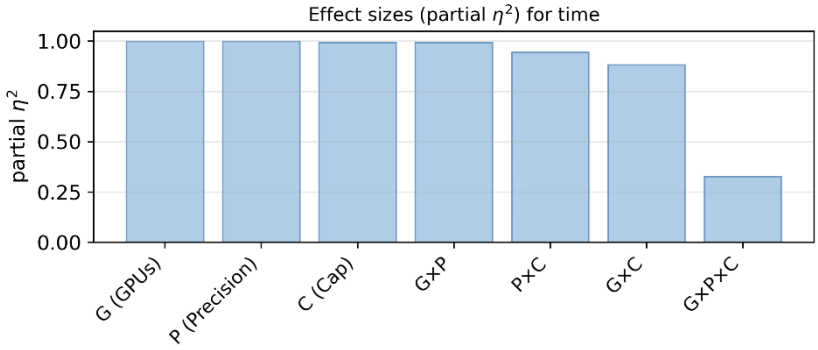

Fig. 6. *Tamaños de efecto del ANOVA factorial (G×P×C) sobre el tiempo de entrenamiento T, incluyendo efectos principales e interacciones.*

Tabla 3: Resultados del ANOVA factorial (tres vías) para T y 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

Sensibilidad de CO_2e a intensidad de carbono (IC)

Como CO^2e_{meas} se definió en posproceso como una función determinista de la energía medida ($CO^2e_{meas} = Ecomp[kWh] \times IC$), la sensibilidad a IC es estrictamente lineal: al aumentar IC, todas las emisiones escalan proporcionalmente, pero no cambia el orden relativo entre configuraciones ni el conjunto de configuraciones preferibles en el plano energía-tiempo. La Figura 7 muestra esta variación para 0.10, 0.20 y 0.30 $kgCO_2e/kWh$, manteniendo fija la energía medida por configuración.

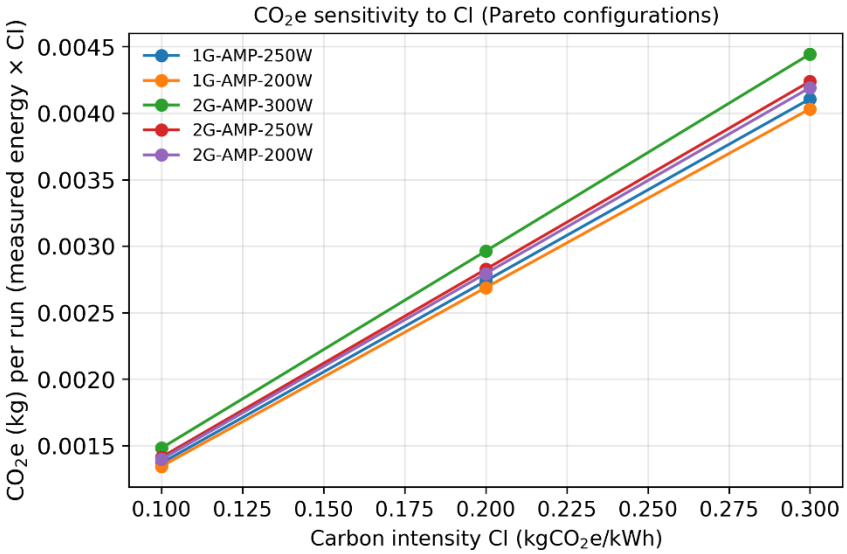

Fig. 7. *Sensibilidad de las emisiones estimadas a la intensidad de carbono (IC).*

DISCUSIÓN

Este trabajo aisló el efecto de escalamiento, precisión y power capping sobre el compromiso energía–tiempo durante el entrenamiento. Bajo estas condiciones, el resultado central es que acelerar no implica necesariamente consumir menos energía, y que los ajustes de sistema como la precisión mixta y los límites en potencias pueden ser más determinantes para la eficiencia que el paralelismo por sí solo.

En primer lugar, AMP fue el factor dominante, debido a que desplazó las configuraciones hacia tiempos y energías sustancialmente menores, separándose claramente del grupo FP32 en el plano energía–tiempo. Esta separación explica que el frente de Pareto esté compuesto únicamente por configuraciones con AMP. Esto es, dentro del espacio evaluado, FP32 queda dominado por alternativas AMP que son simultáneamente más rápidas y menos energéticamente costosas. En términos prácticos, AMP se comporta como una mejora robusta del compromiso energía–tiempo, no solo como un acelerador.

En segundo lugar, DDP redujo el tiempo de forma consistente, pero su impacto en energía fue pequeño y dependiente del régimen: en FP32 se observan ahorros modestos, mientras que en AMP el ahorro no está garantizado y puede revertirse. Esto refleja que al activar dos GPUs aumenta el nivel de potencia promedio del sistema, y la energía total solo baja si la reducción en duración compensa ese incremento y las sobrecargas de sincronización/comunicación.

Finalmente, power capping mostró un punto de equilibrio práctico. Bajar de 300W a 250W redujo energía con una penalización temporal mínima, mejorando métricas compuestas como EDP. En contraste, 200W introdujo penalizaciones temporales mucho mayores, que tienden a deteriorar el balance energía–tiempo pese a reducciones de energía absoluta. Bajo este equipo y protocolo, ello identifica una región operativa cerca de 250W por GPU como un compromiso razonable entre sostenibilidad y productividad.

Como complemento, la comparación con CodeCarbon indica que la herramienta reproduce tendencias, pero presenta una discrepancia sistemática respecto a la medición directa (error relativo mediano $\sim 4\%$, con subestimación). Por tanto, CodeCarbon es útil para monitoreo y comparaciones, pero en análisis finos conviene acompañarlo con medición directa o reportar explícitamente su error. Por último, la sensibilidad a IC es estrictamente lineal: cambia la escala de CO_2e, pero no altera el orden relativo entre configuraciones.

El alcance de estos resultados está condicionado por el protocolo experimental: ResNet-50 sobre CIFAR-10, una estación de trabajo con GPUs NVIDIA RTX A6000 y un entrenamiento parcial con presupuesto fijo. Por ello, la magnitud de los efectos observados no debe extrapolarse directamente a arquitecturas transformer o modelos fundacionales, conjuntos de datos de mayor escala, otros aceleradores ni sistemas con más de dos GPUs, donde pueden cambiar las relaciones entre cómputo, memoria, comunicación, tiempo y energía.

El escalamiento evaluado se limitó a la transición de una a dos GPUs dentro de una misma estación de trabajo. En consecuencia, el estudio no caracteriza el escalamiento distribuido masivo ni multi-nodo, donde la sincronización colectiva, la comunicación inter-nodo, la topología y el ancho de banda de la interconexión, la congestión y el solapamiento entre cómputo y comunicación pueden modificar el compromiso energía-tiempo.

CONCLUSIONES

Este trabajo reporta evidencia reproducible, bajo un presupuesto fijo de cómputo, sobre cómo decisiones operativas comunes reconfiguran el compromiso energía–tiempo y, por extensión, la huella de carbono asociada al entrenamiento. En primer lugar, se confirma que el escalamiento multi-GPU (1 a 2 GPU) acelera de forma consistente, pero no garantiza ahorro energético, esto es, el tiempo disminuye de manera marcada, mientras que la energía total atribuida al cómputo CPU+GPU puede cambiar poco o incluso aumentar según el régimen. Esto refuerza una idea clave desde la perspectiva ambiental, terminar antes reduce el tiempo de exposición del sistema, pero el impacto climático depende del producto entre duración y potencia, no del tiempo por sí solo.

En segundo lugar, se derivan criterios prácticos para mejorar eficiencia. AMP desplaza el entrenamiento hacia una región más eficiente reduciendo simultáneamente tiempo y energía, lo que lo convierte en una decisión robusta cuando el objetivo es disminuir consumo y emisiones sin sacrificar productividad. El power capping permite afinar el balance, en este sistema, 250W por GPU ofrece un compromiso favorable reduciendo consistentemente la energía con penalización temporal mínima, mientras que 200W tiende a penalizar el tiempo de forma desproporcionada para ahorros energéticos marginales, siendo más apropiado cuando se prioriza estrictamente reducir consumo aun si aumenta la duración.

En tercer lugar, el análisis mediante fronteras tiempo–energía proporciona una lectura clara y comparable de las configuraciones, separa extremos (mínimo tiempo vs. mínima energía) e identifica una región operativa intermedia donde el balance entre aceleración y sostenibilidad resulta más práctico. Esta representación es especialmente útil para guiar decisiones reales, porque explicita que no existe una configuración mejor universal, sino elecciones óptimas condicionadas por el objetivo (velocidad, energía o equilibrio).

Finalmente, respecto a emisiones, este trabajo reporta CO_2e como una función determinista de la energía medida, lo que permite interpretar el impacto ambiental de manera transparente, la energía es el puente entre la ejecución y el clima, y el factor que traduce kWh a CO_2e es la intensidad de carbono (IC) de la red eléctrica. Bajo esta formulación, cambiar CI no altera el ranking de configuraciones (solo escala las emisiones), pero sí afecta el impacto absoluto, subrayando que la sostenibilidad del entrenamiento depende tanto de la eficiencia del sistema como del contexto energético regional. En ese marco, CodeCarbon resulta útil para reproducir tendencias y facilitar reportes comparables, pero en este experimento mostró una subestimación sistemática frente a la medición directa (error relativo mediano $\sim 4\%$). Por ello, para conclusiones ambientales con diferencias finas entre configuraciones, es recomendable acompañar estimaciones tipo CodeCarbon con medición directa o, al menos, reportar explícitamente su error y sesgo, de modo que el CO_2e estimado no se interprete como un valor absoluto incuestionable sino como una aproximación con incertidumbre cuantificada. En consecuencia, los resultados deben interpretarse dentro de este protocolo y no como una caracterización general de entrenamientos hasta convergencia o de sistemas distribuidos de mayor escala.

Principio del formulario

REFERENCIAS

[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.

FINANCIACIÓN

Ninguna

CONFLICTOS DE INTERÉS

No existen

CONTRIBUCIÓN DE AUTORÍA

Redacción - borrador original: Carlos Nelson Henríquez Miranda, Malak Andrés Sánchez Cataño, German Sánchez Torres.

Redacción - revisión y edición: Carlos Nelson Henríquez Miranda, Malak Andrés Sánchez Cataño, German Sánchez Torres.