Computer Standards & Interfaces 99 (2027) 104170

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Estimating Deep Learning energy consumption based on model architecture and training environment☆

Santiago del Rey [a], Luís Cruz [b], Xavier Franch [a], Silverio Martínez-Fernández [a,*]

[a] Universitat Politècnica de Catalunya, Barcelona, Spain
[b] Delft University of Technology, Delft, Netherlands

ARTICLE INFO

Dataset link: https://zenodo.org/records/17041051

Keywords:
Green in AI
Energy consumption estimation
Neural networks
Empirical software engineering
Sustainable computing

ABSTRACT

To raise awareness of the environmental impact of deep learning (DL), numerous studies have estimated the energy consumption of DL systems. However, energy estimates during DL training often rely on unverified assumptions. This work addresses that gap by investigating how model architecture and training environment affect energy consumption. We train a variety of computer vision models and collect energy consumption and accuracy metrics to analyze their trade-offs across configurations. Our results show that selecting the right *model–training environment* combination can reduce training energy consumption by up to 80.68% with less than 2% loss in F_1 score. We find a significant interaction effect between model and training environment: energy efficiency improves when GPU computational power scales with model complexity. Moreover, we demonstrate that common estimation practices, such as using FLOPs or GPU TDP, fail to capture these dynamics and can lead to substantial errors. To address these shortcomings, we propose the Stable Training Epoch Projection (STEP) and the Pre-training Regression-based Estimation (PRE) methods. Our evaluation demonstrates that STEP and PRE achieve reductions in Root Mean Squared Error (RMSE) up to 97% and 84%, respectively, when compared to existing estimation tools.

1. Introduction

We are witnessing an unprecedented rise in the adoption of Artificial Intelligence (AI) technologies across domains [1,2]. This growth is driven by recent advances in Deep Learning (DL), including large language models and generative AI, which have revolutionized the landscape of AI applications [3]. However, this progress comes at a cost: the growing energy footprint of DL model training [4,5]. For instance, training Gemini Ultra is estimated to have emitted 37,620 tonnes of CO_2eq [6], an amount comparable to the annual electricity-related emissions of 7840 U.S. households.[1] As DL systems become integral parts of mainstream software products, enhancing the energy efficiency of training pipelines has become an urgent concern, not only for environmental sustainability but also for economic viability and scalability.

In response to this challenge, the research community has increasingly embraced the concept of *Green AI*, which advocates for the systematic consideration of energy and environmental costs in AI research and development [7]. Gartner further reinforces this trend by identifying "energy-efficient computing" as a top strategic technology trend shaping responsible innovation in 2025.[2] Within this context, Gutiérrez et al. [8] distinguish between two complementary research directions: *Green BY AI*, which uses AI to improve sustainability in external domains, and *Green in AI*, which focuses on reducing the energy consumption of AI systems themselves. This work firmly belongs to the *Green in AI* research agenda.

Substantial progress has already been made toward improving the energy efficiency of DL training. Prior studies have proposed lightweight architectures [9], optimization techniques such as pruning and quantization [10], and training strategies that reduce computational overhead. These contributions provide data scientists with an

☆ This work was supported by Grant PID2024-156019OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by ERDF, EU. Additionally, we thank the GAISSA-Optimizer research project under the AGAUR IdC program (2025 PROD 00236). This work is also partially supported by the Joan Oró pre-doctoral support program (2023 FI-1 00374), co-funded by the European Union.

* Corresponding author.
E-mail addresses: santiago.del.rey@upc.edu (S. del Rey), L.Cruz@tudelft.nl (L. Cruz), xavier.franch@upc.edu (X. Franch), silverio.martinez@upc.edu (S. Martínez-Fernández).

[1] The equivalence has been obtained from the U.S.EnvironmentalProtectionAgency.
[2] https://www.gartner.com/en/articles/top-technology-trends-2025

https://doi.org/10.1016/j.csi.2026.104170
Received 5 September 2025; Received in revised form 2 March 2026; Accepted 15 May 2026
Available online 21 May 2026

increasingly rich set of tools to reduce energy consumption at the *model level*. As a result, the role of model architecture selection in energy-aware DL development is relatively well understood.

In contrast, considerably less attention has been paid to *system-level design decisions* surrounding DL training. In practice, software engineers are responsible for selecting and configuring the training environment in which models are executed, ranging from consumer-grade desktops to cloud servers or high-performance computing (HPC) infrastructure. Yet, existing literature provides limited empirical guidance on how such training environments influence energy consumption, or how they interact with different model architectures. As a result, fundamental questions remain unanswered: Should a given model be trained on a desktop or a server? Does more powerful hardware always lead to more energy-efficient training? And how should practitioners reason about these trade-offs in real-world software systems?

Recent standardization efforts aim to improve transparency and consistency in reporting AI energy consumption. For instance, ISO/IEC DTR 20226 [11] outlines sustainability aspects of AI systems across their life cycle, while the Green Software Foundation's Software Carbon Intensity (SCI) specification [12] defines a methodology for estimating carbon emissions from software systems. National initiatives, such as those by UNE [13] and DIN [14], pursue similar goals. However, these standards are still emerging and have seen limited adoption in practice. Consequently, practitioners continue to rely heavily on software-based estimation tools and simplified metrics when assessing training energy consumption.

Although numerous studies estimate the energy consumption or carbon emissions of DL training [15,16], most rely on simplifying assumptions, such as treating Thermal Design Power (TDP) or FLOPs as reliable proxies for energy usage, that do not account for the complex interplay between model architecture, hardware characteristics, and training configuration. Moreover, prior empirical studies typically analyze either architectural choices or execution environments in isolation. As a result, it remains unclear how these factors jointly shape energy consumption and whether commonly used estimation practices remain valid across heterogeneous training environments.

This paper addresses these limitations through a comprehensive empirical study of the energy consumption of DL training that explicitly considers the interaction between model architecture and the training environment. We systematically train a diverse set of computer vision models across multiple realistic environments, collecting fine-grained measurements of energy consumption, GPU usage, and accuracy. By statistically analyzing these data, we show that energy efficiency is not an inherent property of either the model or the hardware alone, but rather emerges from their specific pairing. Furthermore, we empirically demonstrate that widely used energy estimation practices based on FLOPs and TDP fail to capture these interaction effects and can lead to substantial estimation errors.

Building on these empirical insights, we introduce two data-driven energy estimation methods: Stable Training Epoch Projection (STEP) and Pre-training Regression-based Estimation (PRE). In contrast to existing tools that approximate energy consumption using static proxies such as FLOPs or GPU TDP, our methods explicitly model observed training dynamics and system context. As a result, STEP and PRE achieve substantially lower estimation error than state-of-the-art approaches, reducing RMSE by up to 97% and 84%, respectively, in our experimental setting. These improvements demonstrate that accurate energy estimation requires moving beyond proxy-based assumptions toward empirically grounded, context-aware models.

To ensure the relevance of our findings, we focus on computer vision (CV) workloads. CV models are widely used in safety-critical and resource-intensive applications, including autonomous driving, medical image analysis, and industrial inspection systems. Moreover, convolutional architectures are commonly used as benchmarks in energy efficiency research [17,18], allowing our results to be compared with a broad body of existing work and to inform energy-aware training decisions beyond the CV domain.

Our results provide actionable guidance for both data scientists and software engineers. Data scientists can use our findings to select architectures that offer favorable energy–accuracy trade-offs, while software engineers can make informed decisions about matching training environments to model complexity. In addition, cloud and infrastructure providers may leverage these insights to dynamically allocate computational resources more energy efficiently.

The rest of the document is structured as follows. Section 2 reviews related work. Section 3 outlines the study design and methodology. Section 4 presents the results of the study, and Section 5 discusses related points. Section 6 highlights the implications of our findings. Section 7 outlines threats to the validity of the study and how we mitigate them. Finally, Section 8 concludes with the main contributions of our work and directions for future work.

2. Related work

This section reviews prior work on the energy consumption of DL training. We first discuss studies analyzing how model architectures and system-level design decisions affect training energy efficiency. We then summarize existing methods and tools for measuring and estimating energy consumption and carbon emissions in DL training.

2.1. Energy consumption of DL training

Energy efficiency has become a central concern in the life cycle of DL systems [19]. Most existing work focuses on comparing the energy consumption of different models or optimization techniques, while fewer studies examine how architectural choices and training environments jointly influence energy usage.

Several studies analyze the impact of hardware and execution configurations. Luckow et al. [20] compare local and cloud-based training, showing that cloud execution increases training time by approximately 1.5× and that multi-GPU scaling yields diminishing efficiency gains. Similarly, Caspart et al. [21] study heterogeneous compute nodes and report that image classification workloads achieve lower power draw and shorter runtimes on single GPUs than on multi-core Central Processing Units (CPUs), while also identifying non-negligible energy consumption during GPU idle periods. These results highlight the importance of execution context beyond raw performance metrics.

Other work examines energy consumption at finer architectural granularity. Li et al. [22] analyze power behavior across popular DL models, frameworks, and hardware platforms, identifying convolutional layers as the dominant contributors to energy usage. Complementarily, Yarally et al. [23] confirm the high energy cost of convolutional layers during CNN training and show that increased model complexity often yields diminishing accuracy returns. In the computer vision domain, Jha et al. [24] demonstrate that reducing computational complexity, measured in Multiply-accumulate operations (MACs), does not necessarily translate into proportional energy savings or throughput improvements, underscoring the limits of proxy-based efficiency metrics.

More recent studies explicitly explore energy–accuracy trade-offs at the model level. Aquino-Brítez et al. [25] propose the *Kappa-Energy Index* to compare the energy efficiency of different architectures trained on similar GPUs, showing that architecture choice significantly affects energy consumption even when training times are comparable. Likewise, Gowda et al. [26] conduct a large-scale empirical analysis across convolutional networks and vision transformers, demonstrating that marginal accuracy improvements often incur disproportionate energy costs. While these studies provide valuable insights into architecture-level efficiency, they largely rely on homogeneous or normalized hardware settings and treat the training environment as a secondary factor.

Table 1
Summary of related work.

Study	Training Env.	Model–environment interaction	Accuracy–energy trade-off	Energy estimation method
[20]	Local vs. Cloud	✗	✗	✗(Focus on training time)
[21]	Single environment (CPU vs. GPU)	✗	✗	Real-time estimation
[22]	Single environment (CPU vs. GPU)	✗	✗	Real-time estimation
[23]	Single environment	✗	✓	Real-time estimation
[24]	Single environment	✗	✗	Proxy-based (MACs/Joule)
[25]	Single environment (two GPUs)	✗	✓	Hardware-based & Real-time estimation
[26]	Multiple GPUs (unknown environments)	✗	✓	Real-time measurement
[27]	Single environment	✗	✓	Real-time auto-regressive prediction
[28]	Single environment	✗	✗	TDP-based
[29]	Single environment	✗	✗	Scaled TDP
[30]	Single environment	✗	✗	FLOP-based statistical model
[31]	Single environment	✗	✗	Hardware- and software-based measurement
This work	**Diverse (Desktop N & ML, Server)**	✓	✓	**Data-driven, context-aware regression models**

Design-time optimization strategies are explored by Reguero et al. [27], who evaluate layer freezing, quantization, and early stopping on VGG-16 and propose a predictive model to identify the optimal epoch to apply these decisions.

Taken together, prior work shows that both model architecture and system configuration influence training energy consumption, yet their interaction remains underexplored. As summarized in Table 1, existing studies typically analyze either architectural or environmental factors in isolation. However, as Cruz et al. [32] emphasize, sustainable DL requires holistic, lifecycle-aware system design rather than isolated model-level optimizations. Our work addresses this gap by empirically and statistically analyzing how model architecture and training environment jointly determine energy consumption, demonstrating that energy efficiency emerges from their combined pairing rather than from either factor alone.

2.2. Measuring energy consumption and carbon footprint in DL training

The growing environmental impact of DL has driven increased attention toward methods for measuring and estimating energy consumption and carbon emissions. Community initiatives such as Hugging Face model cards[3] and the Green Software Foundation[4] promote standardized carbon reporting, while benchmark efforts such as MLPerf Power provide reproducible methodologies for system-level energy efficiency evaluation across diverse hardware platforms and training scales [18]. Additionally, the ML.ENERGY benchmark targets inference energy consumption in realistic deployment scenarios [33].

Energy assessment approaches can be broadly classified into hardware-based and software-based methods [34,35]. We focus on software-based tools due to their accessibility and ease of integration. These tools typically operate either as real-time estimators that monitor hardware utilization during execution or as post-training estimators that infer energy consumption from training parameters and assumed power models.

Among real-time estimators, CodeCarbon[5] is one of the most widely adopted tools, providing a Python interface for tracking energy consumption and carbon emissions. Alternatives such as PowerJoular [36] and EnergyBridge [37] offer similar functionality without requiring Python integration. While these tools are primarily designed for continuous monitoring, we also investigate their potential for projecting training-time energy consumption, thereby reducing unnecessary experimentation and computational overhead.

Post-training estimators include the ML CO_2 Impact Calculator (MLCO2) calculator [28] and the Green Algorithms (GA) tool [29]. Both estimate carbon emissions by first approximating energy consumption. MLCO2 assumes that the GPU's TDP represents average power consumption and multiplies it by execution time. GA extends this model by scaling TDP based on hardware utilization, incorporating memory power draw, accounting for Power Usage Effectiveness (PUE), and using the Pragmatic Scaling Factor (PSF) to represent repeated experiments. Despite their widespread adoption, subsequent studies continue to rely on these foundational assumptions without systematic validation [6,16,17].

Recent empirical work by Newkirk et al. [30] demonstrates that TDP-based estimates can incur errors of 27%–37% on modern H100 nodes, motivating empirically calibrated, architecture-specific models based on FLOPs. Beyond estimation accuracy, Tripp et al. [31] highlight methodological challenges in direct energy measurement, showing that background processes, power management, and I/O behavior significantly influence observed energy consumption. Their findings underscore that energy usage depends not only on model execution but also on the surrounding training environment.

Motivated by these limitations, our work adopts a data-driven approach that integrates computational characteristics with training environment properties, such as compute capability and available system memory. By explicitly modeling interactions among hardware, software, and the execution context, we enable more accurate, context-aware predictions of training energy consumption.

3. Study design and execution

This section presents the context of our study, as well as our research goal and research questions. It then describes the study variables, the experimental procedures, and the subsequent data analysis. Fig. 1 provides an overview of the main aspects of the study.

3.1. Context: image classification

We conduct our study in the context of a computer vision task: image classification. The datasets were selected according to four main criteria: (i) diversity of tasks and visual characteristics, (ii) dataset size, to balance experimentation time and result stability, (iii) relevance as established benchmarks, and (iv) applicability to real-world scenarios.

[3] https://huggingface.co/docs/hub/model-cards-co2
[4] https://if.greensoftware.foundation/
[5] https://github.com/mlco2/codecarbon

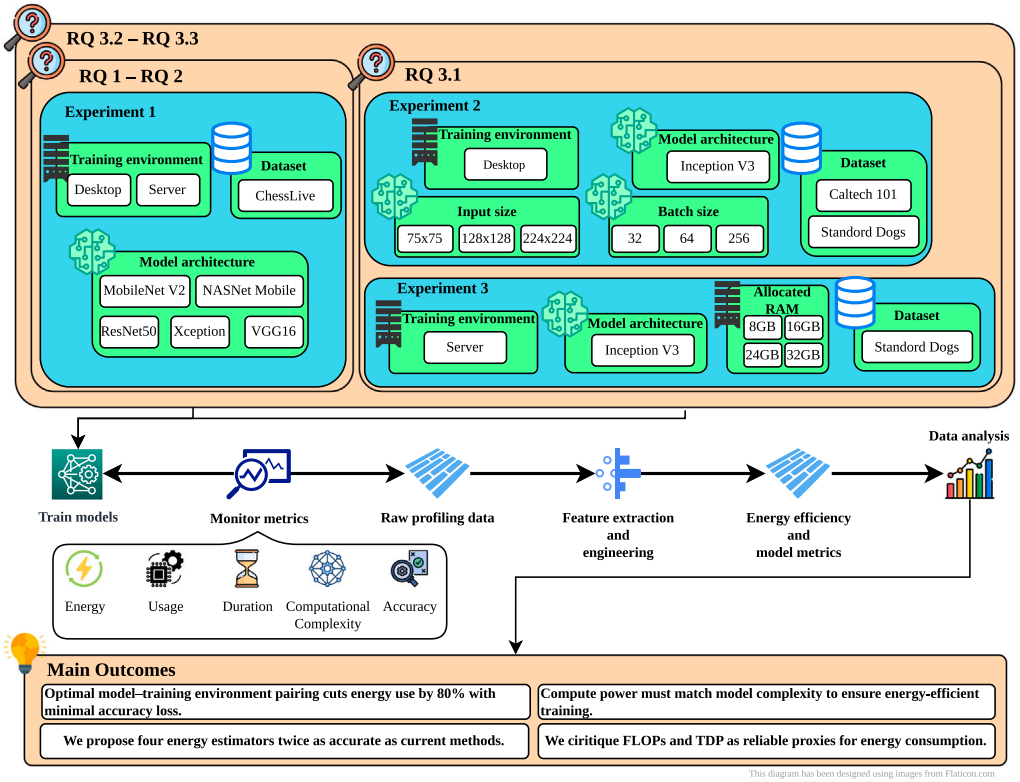

Fig. 1. Structure of the study and main outcomes.

A wide range of image classification datasets exists in the literature. Large-scale benchmarks such as ImageNet [38] have played a central role in advancing deep learning methods. Smaller-scale datasets frequently used for benchmarking include CIFAR-10 and CIFAR-100 [39] and the Oxford-IIIT Pets dataset [40]. More recent datasets, such as iNaturalist [41] and Food-101 [42], focus on fine-grained or domain-specific recognition tasks.

However, not all available datasets are suitable for our experimental objectives. Extremely large datasets, such as ImageNet, introduce substantial computational overhead and are typically used for large-scale pretraining rather than for controlled comparative experimentation. Conversely, very small datasets (e.g., CIFAR-10/100) consist of low-resolution images (32 × 32), which differ significantly from realistic deployment scenarios and may not adequately reflect modern classification challenges. Domain-specific datasets such as Oxford-IIIT Pets provide fine-grained recognition tasks but are structurally and computationally similar to Stanford Dogs, offering limited additional diversity for our study.

Furthermore, while newer datasets such as iNaturalist enable large-scale, fine-grained classification, they are less frequently used in controlled experimental studies due to their size and complexity. Their scale often shifts the experimental focus toward large-scale representation learning rather than architectural or training-variable analysis.

Based on these considerations, we selected two well-known public benchmarks, Caltech-101 and Stanford Dogs. These datasets provide complementary levels of visual and task complexity: Caltech-101 addresses general object classification with moderate intra-class variation, while Stanford Dogs focuses on fine-grained classification with high inter-class similarity. Their moderate dataset sizes allow repeated experimentation while keeping computational costs manageable.

Additionally, we include the Chesslive dataset, a custom dataset created by our research group to support real-world applicability. This dataset targets a binary classification task, detecting empty versus occupied chessboard squares, and is designed to closely reflect the conditions under which the model would be deployed in practice.

Together, these datasets provide a diverse and balanced experimental setting, allowing us to study the variables of interest across different task complexities, visual domains, and dataset scales, while remaining computationally feasible and aligned with practical constraints.

3.2. Research goal and questions

Following the Goal-Question-Metric guidelines [43], we define our goal as:

Analyze *DL model architectures and training environments*
For the purpose of *estimating energy consumption*
With respect to *the combined energy consumption of the GPU and RAM, GPU usage, model accuracy, and their trade-off*
From the point of view of *data scientists and software engineers*
In the context of *image classification model training.*

From this goal, we derive the following Research Questions (RQs):

> **Research Question 1**
>
> What is the impact of model architecture and training environment on image classification training concerning energy consumption and GPU usage?

As we have shown, prior work has explored how model architecture affects the energy consumption of training DL models. However, there remains little evidence regarding the effects of the training environment and its interaction with the model architecture. With RQ_1, we aim to investigate this interaction and its impact on the combined training energy consumption of the GPU and RAM, as well as GPU usage. Our goal is to determine whether the choice of model architecture affects the selection of the training environment when aiming to reduce training energy consumption.

Moreover, seeing how previous work uses the GPU usage as a scaling factor when estimating the energy consumption [29], in RQ_1 we

also study whether this assumption holds when changing the training environment. Additionally, we analyze whether there is a relationship between GPU usage and training energy consumption. To achieve these goals, we divide RQ_1 into two subquestions:

$RQ_{1.1}$: How do model architecture and training environment impact training energy consumption and GPU usage?

$RQ_{1.2}$: Is there a relationship between GPU usage and energy consumption when training DL models?

To answer RQ_1, we measure the combined energy consumption of the GPU and RAM. Specifically, we measure their energy consumption, in Joules, while training multiple model architectures in two different training environments (i.e., Server and Desktop, see below), and sum them to obtain their combined energy consumption for training, which we abbreviate in the remainder of this document as *energy consumption*. The GPU usage is measured as the percentage of time during which at least one core of the GPU is active.

We refer to *Server* environments as those present in High-Performance Computing (HPC) centers or similar facilities, which contain clusters of highly specialized hardware, such as very powerful CPUs and GPUs, and powerful refrigeration systems. For *Desktop* environments, we refer to the hardware components found in a consumer-class laptop or desktop computer (e.g., medium to low-end CPUs and GPUs).

Research Question 2

Do potential gains in accuracy achieved by energy-greedy DL model architectures justify their increased energy consumption in image classification tasks?

Previous work has raised concerns about the trend of increasing the computational demands of DL models to improve their accuracy [7,44,45]. In RQ_2, we analyze the trade-off between energy consumption and model accuracy across five model architectures with varying computational complexity. We aim to examine whether model architectures with higher energy consumption achieve higher accuracy and whether the increase in accuracy is significant, so that their higher energy demands can be justified.

To answer this RQ, we measure the energy consumption, as defined in RQ_1, and the model accuracy in terms of F_1 score.

Research Question 3

Can we accurately estimate energy consumption before training DL models for image classification?

With RQ_3, we aim to develop a method that reliably estimates energy consumption. To this end, we first investigate how different parameters (i.e., batch size, input size, and allocated RAM) affect variables such as energy consumption and RAM usage. Then, we look for patterns in the power consumption of training runs to determine whether it can be used to estimate energy consumption. Finally, we evaluate our proposed method by comparing its performance to two popular estimation methods (i.e., MLCO2 and GA). The final goal of this RQ is to provide data scientists with a way to observe the energetic implications of building their DL models without actually training them. This way, they can compare different model architectures and training environments with respect to energy consumption much faster, without consuming as much energy as they would otherwise. To achieve this goal, we divide RQ_3 into three subquestions:

$RQ_{3.1}$: Which parameters can help estimate the energy consumption?

$RQ_{3.2}$: Are there recurring patterns between epochs within the same training run? If so, can we use them to predict energy consumption?

$RQ_{3.3}$: How reliable is the proposed energy estimation method?

3.3. Variables

This section presents and motivates the study's variables. Table 2 summarizes all variables and their operationalization.

Independent Variables. We select five independent variables that reflect common decisions made by data scientists and software engineers when designing and executing deep learning training pipelines.

Model architecture refers to a set of DL models widely adopted by the community that vary in size, layer types, and connectivity patterns. This variable captures architectural design choices that are known to influence both computational demands and model accuracy.

Training environment models the hardware context in which training is performed, a factor that has received comparatively little attention in prior energy-related studies. We consider three environments that reflect realistic options available to practitioners: one Server environment and two Desktop environments. The "Normal User" Desktop (*Desktop N*) represents consumer-grade hardware, while the "Machine Learning (ML) Engineer" Desktop (*Desktop ML*) models a more computation-oriented setup. This variable enables analysis of how hardware context interacts with model architecture with respect to energy consumption and GPU usage.

Allocated RAM represents scenarios commonly encountered in shared or clustered computing environments, where users must explicitly specify memory requirements when submitting training jobs. This variable enables us to examine how memory allocation affects RAM usage and energy consumption.

Batch size and *input size* are fundamental hyperparameters of deep learning training pipelines. Batch size has been shown to influence training efficiency and energy consumption [46,47], while input size directly affects the number of parameters and computational complexity of the model, increasing its energy demands [48]. Together, these variables capture common optimization choices available to practitioners.

Dependent Variables. To evaluate the impact of the independent variables on energy efficiency and model performance, we consider six dependent variables. These are divided into core outcome metrics and supporting explanatory variables.

The primary metric for energy efficiency is *energy consumption*, defined as the total energy used by the GPU and RAM during training. GPUs are the dominant source of power draw in modern deep learning workloads [22], while RAM can contribute non-negligibly to overall energy consumption [49]. Measuring both components provides a more representative estimate of the training energy footprint than GPU-only approaches.

GPU usage is included as a core metric because it is used in prior work as a proxy or scaling factor for estimating energy consumption [29]. However, GPU usage reflects utilization rather than actual power draw and may vary across hardware and software configurations. Measuring GPU usage alongside real energy consumption enables us to assess the validity of this assumption across different model architectures and training environments. It is important to note that the GPU usage reported by `nvidia-smi`[6] represents the percentage of time the GPU is actively performing computations, not the percentage of GPU cores in use.

Model accuracy, measured using the F_1 score, captures the effectiveness of the trained models. Energy efficiency cannot be meaningfully assessed in isolation from model performance, as reductions in energy consumption may come at the cost of degraded predictive quality. The F_1 score is particularly suitable for image classification tasks where class imbalance may be present, as it jointly accounts for precision and recall. The combined analysis of energy consumption and model accuracy enables the study of energy–accuracy trade-offs, which are central to practical model and environment selection decisions.

[6] https://developer.nvidia.com/system-management-interface

Table 2
The variables of the study.

Name	Scale	Operationalization
Independent variables:		
Model architecture	nominal	Base model: MobileNet V2, NASNet Mobile, ResNet-50, Xception, VGG-16, Inception V3
Training environment	nominal	Three environments: Desktop "Normal User", Desktop "ML Engineer", and Server
Allocated RAM	ordinal	Amount of RAM allocated for the training process (8GB, 16GB, 24GB, 32GB)
Batch size	ordinal	Number of images processed in a training step (32, 64, 256)
Input size	ordinal	Width, and height in pixels of the input images (75×75, 128×128, 224×224)
Dependent variables:		
Energy consumption	ratio	Profiled with nvidia-smi and psutil
GPU usage	ratio	Profiled with nvidia-smi
RAM used	ratio	Profiled with the psutil Python library
Training duration	ratio	Profiled with the datetime Python library
Number of FLOPs	ratio	Measured once through the TensorFlow API
F_1 score	ratio	Measured with the test set after the training phase
Other variables:		
Dataset	nominal	Three datasets: Chesslive, Caltech101, Stanford Dogs

In addition to these core metrics, we measure *RAM usage*, *training duration*, and *model computational complexity* expressed as the number of FLOPs required for a single forward pass. These variables are not treated as primary outcome metrics but are used to support the interpretation of energy consumption and GPU usage. In particular, they help explain observed differences in energy efficiency across model architectures, training environments, and hyperparameter configurations.

Other Variables. To align the experimental setup with real-world use cases, we include the dataset as an additional variable. Datasets represent common image classification tasks on which deep learning models are trained and may influence training dynamics and resource utilization. We use two widely adopted datasets (Caltech101 and Stanford Dogs) and a third dataset developed for other research purposes. Details regarding their composition and characteristics are provided in the following section.

3.4. Datasets

3.4.1. Caltech101

For the Caltech101 dataset [50], we use the TensorFlow Datasets version.[7] The dataset consists of pictures of objects belonging to 101 classes, plus one background clutter class. Each image is labeled with a single object. Each class contains roughly 40 to 800 images, totaling around 9k images. Images are of variable sizes, with typical edge lengths of 200–300 pixels. We use the default train-test splits provided by the TensorFlow API. There are 9,144 images, out of which 3,060 are used for training and 6,084 for testing.

3.4.2. Stanford Dogs

For the Stanford Dogs dataset [51], we use the TensorFlow Datasets version.[8] The dataset contains images of 120 breeds of dogs from around the world. This dataset was constructed from ImageNet images and annotations for the task of fine-grained image categorization. There are 20,580 images, with a default partitioning of 12,000 for training and 8,580 for testing.

3.4.3. Chesslive dataset

Our research group has created the Chesslive dataset for multiple research purposes, since high-quality labeled datasets for chess piece recognition are relatively scarce in the literature. Existing resources are predominantly synthetic or limited in diversity, with the synthetic dataset proposed by Wölflein and Arandjelović [52] and the image collection from Roboflow [53] being among the most commonly used in related work. To ensure robust generalization and realistic evaluation conditions, we constructed the Chesslive dataset by combining synthetic data with real-world imagery.

The dataset builds upon the synthetic chessboard images provided by Wölflein and Arandjelović [52], which include detailed metadata such as Forsyth–Edwards Notation (FEN)[9] strings and board corner coordinates. To reduce the domain gap between synthetic renders and real tournament conditions, we extended this base with real-world images from two sources: the Roboflow image collection [53], and a custom set of images captured during several local chess tournaments. The latter sources were included to reflect realistic lighting conditions, viewpoints, and piece appearances expected in deployment scenarios.

As real-world images consist of full-board images rather than individual square crops, a dedicated preprocessing pipeline was required. This pipeline comprised two main stages. First, labeling was performed using FEN strings, which enabled efficient annotation while providing precise ground truth for each board position. Second, we applied the automated cropping process, proposed by Wölflein and Arandjelović [52], consisting of (i) board corner detection, (ii) perspective rectification, and (iii) extraction of individual board squares based on a regular grid.

To ensure dataset reliability, a two-step validation procedure was conducted. First, during the initial labeling phase, images containing severe occlusions (e.g., hands covering significant parts of the board or pieces) were removed from the dataset to prevent ambiguous or unreliable annotations. Second, following the automated cropping stage, a comprehensive manual inspection of all cropped square images was performed. Whenever a cropped image was found in an incorrect label directory, we traced the error back to the corresponding original board

[7] https://www.tensorflow.org/datasets/catalog/caltech101
[8] https://www.tensorflow.org/datasets/catalog/stanford_dogs

[9] A standard notation for describing a particular board position in a chess game using a single line of text composed of six fields separated by spaces.

Table 3
Number of parameters, GFLOPs, and size of each model architecture used in the study. All data have been extracted using an input size of 128 × 128.

Model architecture	Total parameters	GFLOPs	Size (MB)
MobileNet V2	2,914,369	0.20	11.12
NASNet Mobile	4,811,413	0.37	18.35
ResNet-50	26,211,201	2.53	99.99
Xception	22,960,681	2.96	87.59
VGG-16	14,715,201	10.53	56.13
Inception V3	40,797,062	1.50	155.63

image and its associated FEN string. The annotation was then corrected to ensure consistency among the square image, its class label, and the board-state representation.

After preprocessing and validation, the Chesslive dataset consists of 34,010 color images of size 50 × 50 pixels, evenly split between occupied and empty squares. As this is a custom dataset, we use a 70%/30% train/test split, yielding 23,807 training images and 10,203 test images.

In all three cases, we do not set aside a portion of the dataset for validation, as model optimization via hyperparameter tuning is beyond the scope of this analysis.

3.5. Experiment setting

To implement the model architectures, we use the TensorFlow[10] implementations for six base models: MobileNet V2, NASNet Mobile, ResNet-50, Xception, VGG-16, and Inception V3. Each model considerably differs in the number of parameters, FLOPs, and storage size (see Table 3). We apply transfer learning to train our models. As such, we use the base models without their top layers (i.e., the fully connected and output layers) and replace them with a new top layer. We follow the same training process for the first five models. We start by fixing the same hyperparameters for the five models, then we train each model on the Chesslive dataset in two consecutive steps: (i) train the models with all the layers frozen except for the top; and (ii) unfreeze all the layers and fine-tune the whole network. The Inception V3 model is trained on the Caltech101 and Stanford Dogs datasets. Its training process consists of a single step in which we attach a new top to the base model and train the model while the base model's layers are frozen. To speed up experimentation, the first five models are excluded from $RQ_{3.1}$. Also, the Inception V3 model is only used during RQ_3.

Regarding the training environment, we use an Nvidia GTX 750 Ti (2 GB) in Desktop *N* and an Nvidia RTX 3070 (8 GB) in Desktop ML. Both settings use a 32 GB 3,600 MT/s DDR4 memory. For the Server environment, we use the Universitat Politècnica de Catalunya's rdlab[11] HPC service, including an Nvidia RTX 3090-24GB and 16 GB 2,666 MT/s DDR3 of memory for RQ_1 and RQ_2, and 8 GB to 32 GB for $RQ_{3.1}$.

We use nvidia-smi to collect the following GPU metrics: power draw, utilization, and temperature. We use the psutil library to measure the amount of RAM used during the training process. We set the sampling interval to 1 s. The GPU energy consumption is computed as the integral of the power over the training time. To compute the RAM energy consumption, we first transform the measurements of used RAM to power using the ratio 0.375W/GB of RAM used [34]. We then compute the RAM energy consumption as the integral of power over training time. Finally, we sum the GPU and RAM energy consumption to obtain the combined energy consumption.

The complete experiment and data analysis are implemented in Python 3.9.13.[12] The code used to compute FLOPs and analyze the data is available in our **replication package** [54].

[10] https://www.tensorflow.org/versions/r2.10/api_docs
[11] https://rdlab.cs.upc.edu/
[12] Python3.9.13release

3.6. Experiment execution

To answer RQ_1 and RQ_2, we define a 5 × 3 factorial design where the model architecture factor has five levels and the training environment factor has three levels (see Table 2). For each *model architecture*× *training environment*, we run the training process 30 times to provide more reliable measurements of the dependent variables. This design allows us to measure the energy consumption, GPU usage, and model accuracy under different combinations of model architecture and training environment, allowing us to study their effects and interactions on all dependent variables. Additionally, conducting a factorial design provides all the required data to address the RQs. We only need to select the relevant factors and levels for each RQ (see Section 3.7).

To answer RQ_3, we first need to understand which parameters influence energy consumption so that we can use them to estimate it. This process corresponds to $RQ_{3.1}$, which we answer through two experiments using the Inception V3 model architecture.

The first experiment is conducted in the Desktop ML environment and employs a 3 × 3 factorial design with batch size and input size as factors. Each of the nine combinations is trained 30 times while measuring the energy consumption. The experiment uses two datasets, Caltech-101 and Stanford Dogs, to account for dataset-specific variation. This design enables robust analysis of the main and interaction effects of batch and input sizes on energy consumption.

In the second experiment, we train the Inception V3 model on the Stanford Dogs dataset in the Server environment while changing the allocated RAM to 8 GB, 16 GB (default in previous experiments), 24 GB, and 32 GB. The batch size is fixed to 256, and the input size to 128 × 128. In this experiment, we aim to verify whether allocating additional RAM can reduce energy consumption, suggesting that it is a relevant factor in estimating the final consumption.

To ensure accurate measurement of energy consumption, we follow the guide by Cruz [55]. We perform a dummy GPU-intensive warm-up task before all experiments, in which we train one of the DL models for approximately 10 min. Additionally, we schedule 5-minute pauses between the 30 runs of each experiment to prevent thermal throttling.

$RQ_{3.2}$ regards the study of patterns in energy consumption within the training process, and the development of an energy estimation method. To answer this RQ, we decided not to execute more experiments, but to use all the data we have already collected during the executions of all the experiments previously described.

In $RQ_{3.3}$ we evaluate the proposed energy estimation methods and compare their performance with widely used energy estimation methods (i.e., MLCO2 and GA).

3.7. Data analysis

This section outlines the preprocessing steps and statistical methods used to analyze the experimental data.

3.7.1. Data processing

We first cleaned the dataset by removing failed runs, empty values, and runs where the sampling rate exceeded the epoch duration, reducing the original 1630 runs to 1365 valid entries. Outliers were then removed separately for each dependent variable to preserve valid data for variables with non-extreme values. Detailed steps are available in the replication package.

For RQ_1 and RQ_2, we excluded runs from the Desktop *N* environment, where only MobileNet V2 and NASNet Mobile could be trained due to GPU RAM constraints. These runs were retained to support pattern analysis in RQ_3. Finally, we normalized total energy consumption by the number of images processed during training to enable fair comparisons across models and environments, accounting for differences in training time and model complexity, inspired by the approach of Fischer et al. [56].

3.7.2. Statistical analysis

For each research question, we evaluated several candidate regression models reflecting different distributional assumptions and link functions. Model selection followed a multi-criteria procedure. Candidate models were first required to satisfy standard residual diagnostic checks, including assessments of normality, homoscedasticity, independence, and influential observations, as appropriate for each modeling framework. When multiple models met these diagnostic requirements, we compared their performance using standard goodness-of-fit and information-theoretic metrics, including the coefficient of determination (R^2), the Akaike Information Criterion (AIC), and the Bayesian Information Criterion (BIC), as implemented in the `performance` R package. The simplest model providing an adequate fit to the data was selected. In the following, we report and discuss only the final selected models used to answer each research question.

For $RQ_{1.1}$, we fitted two linear regression models: one predicting normalized energy consumption and another predicting GPU usage. Linear models were chosen because both outcomes are continuous and because the primary objective was to estimate and compare mean differences across experimental conditions. The predictors included model architecture, training environment, and their interaction, allowing us to assess both main effects and potential moderation effects. Since GPU usage is naturally bounded between 0 and 100 and exhibited pronounced left-skewness, we applied a square-root transformation to the reflected response, $\sqrt{100 - \text{gpu\_usage}}$, to stabilize variance and improve residual normality. Statistical significance of main effects and interactions was evaluated using Type III ANOVA with sum-to-zero contrasts, which is appropriate for unbalanced designs and facilitates interpretation of interaction terms. Post-hoc pairwise comparisons were conducted using Tukey-adjusted estimated marginal means to control the family-wise error rate.

To answer $RQ_{1.2}$, we employed two additional linear regression models predicting normalized energy consumption and average power draw, respectively. Linear modeling was again appropriate given the continuous nature of both outcomes. The models included GPU usage, model architecture, training environment, and their interactions to capture both direct effects and joint dependencies among system- and model-level factors. Model fit and statistical significance were assessed using the same Type III ANOVA framework as above.

For RQ_2, we examined the trade-off between energy consumption and model accuracy using a two-stage approach. First, we modeled F_1 scores using a Generalized Additive Models for Location, Scale, and Shape (GAMLSS) with a Beta distribution. This modeling choice was motivated by the continuous nature of the F_1 score and its strict confinement to the interval $[0, 1]$, for which the Beta distribution provides a flexible and theoretically appropriate likelihood. Model computational complexity (FLOPs) was included as the primary explanatory variable. The training environment was initially included but removed from the final model due to insufficient statistical significance. Statistical significance was assessed using Type III ANOVA, and Tukey-adjusted estimated marginal means were used for pairwise comparisons. In the second stage, we computed the Pareto front to identify models that achieve an optimal balance between energy consumption and predictive performance, reflecting the multi-objective nature of this research question.

For $RQ_{3.1}$, we fitted a Generalized Linear Model (GLM) with a Gamma distribution to analyze the effects of batch size, input size, and dataset on normalized energy consumption. The Gamma family was selected because the normalized energy consumption is strictly positive and right-skewed. All interaction terms were included to capture potential non-additive effects among experimental factors. Statistical significance was assessed using Type III ANOVA. In addition, we evaluated the influence of allocated RAM on energy consumption using a separate model with a single predictor. Because no interactions were present in this analysis, Type I ANOVA was employed, which is appropriate for sequential testing in simple model structures.

For $RQ_{3.2}$, we analyzed power usage time series to identify recurring consumption patterns. We employed the FLUSS segmentation algorithm proposed by Gharghabi et al. [57], as implemented in the `stumpy` Python library [58]. This method was selected for its ability to detect regime changes and structural motifs in time series data without requiring prior specification of segment boundaries.

Finally, for $RQ_{3.3}$, we trained multiple regression models corresponding to two proposed energy estimation methods. Model training and hyperparameter selection were performed using 5-fold cross-validation on 70% of the data to mitigate overfitting and ensure robust performance estimates. The remaining 30% of the data were held out for a final comparison against two existing baseline approaches. Model accuracy was primarily evaluated using the RMSE, as it penalizes large prediction errors more heavily, which is particularly critical in the context of energy estimation. Additionally, R^2 and Mean Absolute Error (MAE) were computed during cross-validation to provide complementary perspectives on model fit and predictive reliability.

All statistical tests were conducted at the 95% confidence level. Complete modeling details, including preprocessing scripts, diagnostic plots, and replication materials, are provided in the accompanying replication package.

4. Results

This section presents the experimental findings addressing the RQs guiding this study.

4.1. RQ1: Impact of DL model architecture and training environment in energy consumption and GPU usage

(1) Results $RQ_{1.1}$: As shown in Fig. 2, both energy consumption and GPU usage during image classification training were significantly influenced by model architecture and training environment.

MobileNet V2 consistently showed the lowest energy consumption across both environments, while VGG-16 exhibited the highest. Although some trends held across environments, notable deviations emerged. In the Desktop ML environment, energy usage increased consistently with model computational complexity. However, this pattern was disrupted in the Server environment, where ResNet-50 consumed less energy than NASNet Mobile, whereas NASNet Mobile and Xception exhibited comparable energy use.

Linear model analysis revealed significant main effects and interaction effects on per-image energy consumption ($p < .001$). Tukey-adjusted post hoc comparisons showed that MobileNet V2 trained in the Desktop ML environment was the most energy-efficient configuration, exhibiting significantly lower energy usage than all others (all $p < .0001$), achieving an 80.68% reduction compared to the most demanding setup (VGG-16–Desktop ML). Architecture was the dominant explanatory factor ($\eta^2 = 0.851$), although it overlapped with the interaction effect ($\eta^2 = 0.065$) due to collinearity. Training environment explained much less variance on its own ($\eta^2 = 0.083$), but meaningfully modified architecture effects (especially for VGG16). Thus, its importance lies in its interaction with architecture rather than as a dominant main effect.

Interestingly, the Server environment did not consistently result in lower energy consumption. Except for VGG-16, all models consumed more energy when trained on the Server. For instance, NASNet Mobile's energy consumption was significantly higher on the Server than on Desktop ML (estimate = –0.155, $p < .0001$), with similar trends for MobileNet V2, ResNet-50, and Xception. Additionally, the energy consumption of NASNet Mobile and Xception was statistically indistinguishable ($p = 1$), consistent with visual observations. These results indicate that greater computational capacity does not consistently improve energy efficiency.

Regarding GPU usage, Fig. 2 reveals substantial variation across model architecture–training environment combinations. VGG-16,

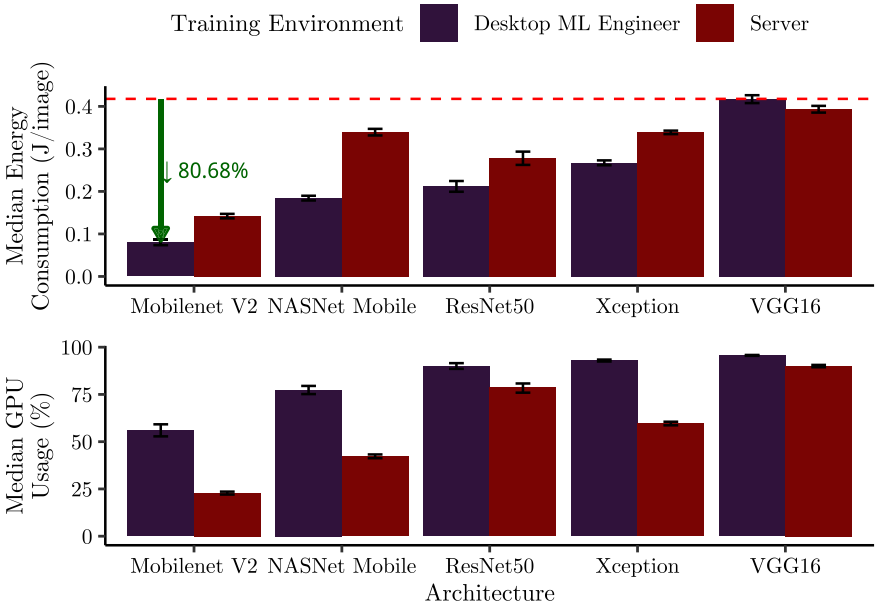

Fig. 2. Comparison of the five model architectures trained in the different environments in terms of median energy consumption per image (top) and median GPU usage (bottom). Architectures are ordered by ascending FLOPs.

trained in Desktop ML, achieved the highest utilization, followed closely by Xception in the same setup. In contrast, MobileNet V2 in the Server environment showed the lowest usage, followed by NASNet Mobile.

The linear model fitted to reflected-transformed GPU usage data revealed significant main and interaction effects ($p < 0.001$ for all). Architecture accounted for most of the variance ($\eta^2 = 0.679$), followed by environment ($\eta^2 = 0.269$) and the interaction ($\eta^2 = 0.051$). Partial η^2 values exceeded 0.98, indicating robust effects.

Tukey-adjusted pairwise comparisons showed that 44 of 45 architecture-environment pairs differed significantly ($p < .0001$), except for ResNet-50 in Desktop ML and VGG-16 in Server, which had similar GPU usage.

Answer to $RQ_{1.1}$

Model architecture and training environment have a substantial, non-uniform effect on energy consumption and GPU usage during training. Lightweight models (e.g., MobileNet V2, NASNet Mobile) exhibit high variability across environments, whereas heavier models (e.g., VGG-16) consistently incur higher computational costs. Efficiency depends on the model architecture–training environment combination. No environment is universally optimal, but in resource-constrained contexts, lightweight models in moderately provisioned environments provide the most favorable trade-offs.

(2) Results $RQ_{1.2}$: Fig. 3 illustrates the relationship between GPU usage, normalized energy consumption (left panel), and average power draw (right panel). In the Desktop ML environment, energy consumption scales linearly with GPU usage up to ≈80%, after which it increases exponentially. In contrast, the Server environment shows no clear linear pattern, though a general positive trend is present. Notably, operating at 40% GPU usage in the Server environment can result in energy consumption comparable to, or even higher than, that at 60% or 80% usage levels.

To further explore these relationships, we applied the linear model introduced in Section 3.7. Type III ANOVA revealed a significant main effect of model architecture ($F = 19.6, p < .001$), along with significant two-way interactions between GPU usage and architecture ($F = 22.88, p < .001$) and between model architecture and training environment ($F = 5.37, p < .001$). A significant three-way interaction was also observed between GPU usage, model architecture, and training environment ($F = 4.58, p = .001$).

Total eta-squared indicated that training environment and model architecture each explained approximately ≈37% of the total energy consumption variance, while GPU usage accounted for ≈22%. Interaction effects accounted for less than 3% of the total variance.

The right panel of Fig. 3 depicts the relationship between GPU usage and average power draw. In the Desktop ML environment, the trend resembles that of energy consumption: a steep rise beyond 80% usage, suggesting diminishing returns in energy efficiency. In the Server environment, power draw increases more linearly with GPU usage. This divergence from the energy trend is likely due to differences in image-processing time across model architectures, which affect total energy despite similar power levels.

Finally, we highlight a key limitation of commonly used power estimation practices. Estimating average power via TDP scaled by GPU utilization leads to substantial inaccuracies: it overestimates power in the Desktop ML setup and underestimates it in the Server. This renders TDP-based methods unreliable for cross-environment comparisons of model energy efficiency.

Answer to $RQ_{1.2}$

The effect of GPU usage on energy consumption is shaped by both training environment and model architecture. High GPU usage can exponentially increase energy costs depending on the environment. TDP-based power estimation methods are inaccurate across environments, underscoring the need for more robust alternatives.

4.2. RQ2: Trade-offs between energy consumption and model accuracy

A preliminary analysis of descriptive statistics (Table 4) revealed minimal differences in mean F_1 scores across model architectures, except for ResNet-50. The largest performance drop — 0.16 — was

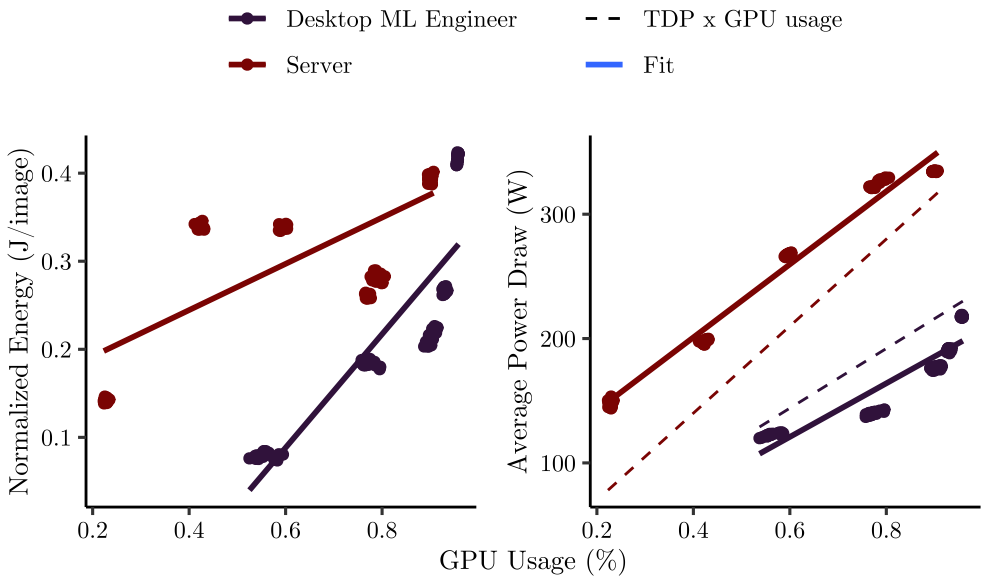

Fig. 3. Relationships between GPU usage, and normalized energy and power draw.

Table 4
Descriptive statistics for F_1 score by model architecture.

Model architecture	GFLOPs	F_1 score		
		Mean	Std.	Median
MobileNet V2	0.20	0.97	0.015	0.97
NASNet Mobile	0.37	0.98	0.001	0.98
ResNet-50	2.53	0.83	0.168	0.93
Xception	2.96	0.98	0.001	0.98
VGG-16	10.53	0.99	0.001	0.99

observed between VGG-16 and ResNet-50. To assess the influence of computational complexity, we fitted a GAMLSS model using GFLOPs to predict both the mean and dispersion of F_1 scores. The inclusion of GFLOPs led to a substantial model improvement over the null $\Delta AIC \approx 1000$, with GFLOPs emerging as a highly significant predictor ($p < .001$). Tukey-adjusted pairwise comparisons confirmed significant differences across all GFLOPs levels ($p < .001$), with ResNet-50 (2.5 GFLOPs) performing significantly worse than both less and more complex models.

Having established a significant effect of model architecture on both energy consumption and F_1 score, we examined the trade-offs between them. Fig. 4 displays the total energy consumption versus F_1 score across all training runs, with the Pareto front highlighted. Inspection of this front revealed that NASNet Mobile does not contribute any optimal instances, suggesting that other models — especially MobileNet V2 — offer more favorable trade-offs. Indeed, nearly half of the Pareto-optimal configurations correspond to MobileNet V2.

Answer to RQ$_2$

Our analysis shows that energy-intensive models yield negligible gains in the F_1 score relative to energy-efficient alternatives in our context. For example, replacing VGG-16 with MobileNet V2 reduces energy consumption by approximately 80%, with only a 0.01 drop in F_1 score.

4.3. RQ3: Energy estimation model

(1) Results RQ$_{3.1}$: Identifying useful parameters for estimating energy consumption. To identify useful predictors of energy consumption during training, we analyzed the effects of batch size and input size. Fig. 5 presents normalized energy consumption across varying configurations for the Inception V3 model, trained on the Desktop ML environment using the Caltech101 and Stanford Dogs datasets.

Batch size demonstrated a consistent energy-saving effect: larger batches significantly reduced energy consumption per image across all input sizes and datasets. This reduction is largely attributed to decreased training time, despite a modest increase in average power usage, as illustrated in Fig. 6.

In contrast, increasing input size substantially elevated energy consumption. For example, input dimensions of 224 × 224 resulted in nearly four times higher energy usage compared to 75 × 75. This increase is driven by the associated rise in computational complexity (FLOPs), reflected in both longer training durations and greater power draw, regardless of batch size or dataset.

The GLM results confirmed that batch size, input size, dataset, and their interactions were all statistically significant ($p < .001$). Effect size analysis revealed that input size was the dominant predictor, accounting for 94.7% of the variance in energy consumption ($\eta^2 = 0.947$). Batch size explained 3.4% ($\eta^2 = 0.034$), while dataset effects were negligible ($\eta^2 = 0.002$). Although absolute interaction effects were small (e.g., $\eta^2 = 0.014$ for batch size × input size), partial effect sizes were large, indicating strong contextual influence (e.g., partial $\eta^2 = 0.993$).

Tukey-adjusted pairwise comparisons by dataset confirmed significant differences between batch–input combinations ($p < 0.001$). Hedges' g analysis further emphasized the magnitude of these effects. Increasing input size led to very large effect sizes—exceeding $g = 200$ when comparing 75 × 75 to 224 × 224. For example, training with a batch size of 32 and an input size of 75 × 75 consumed substantially less energy than training with 224 × 224 inputs: $g = -214.2$ (Caltech101) and $g = -218.4$ (Stanford Dogs). Even moderate increases (e.g., to 128 × 128) produced large effects ($g = -87.5$ to -130.6).

Larger batch sizes consistently reduced energy usage, particularly at lower input sizes. For example, increasing the batch size from 32 to 256 at 75 × 75 resolution yielded large positive effects ($g = 71.1$ for Caltech101, $g = 70.9$ for Stanford Dogs), indicating substantial energy savings per image. In contrast, increasing batch size at high input sizes (e.g., 224 × 224) had smaller but still notable effects (e.g., $g = 30.6$ for Caltech101 and $g = 40.3$ for Stanford Dogs when comparing batch sizes 32 and 256).

The most substantial energy reductions were observed when the batch size increased and the input size decreased. These consistent, large effect sizes across datasets highlight the strong practical impact of these training parameters on energy efficiency in DL.

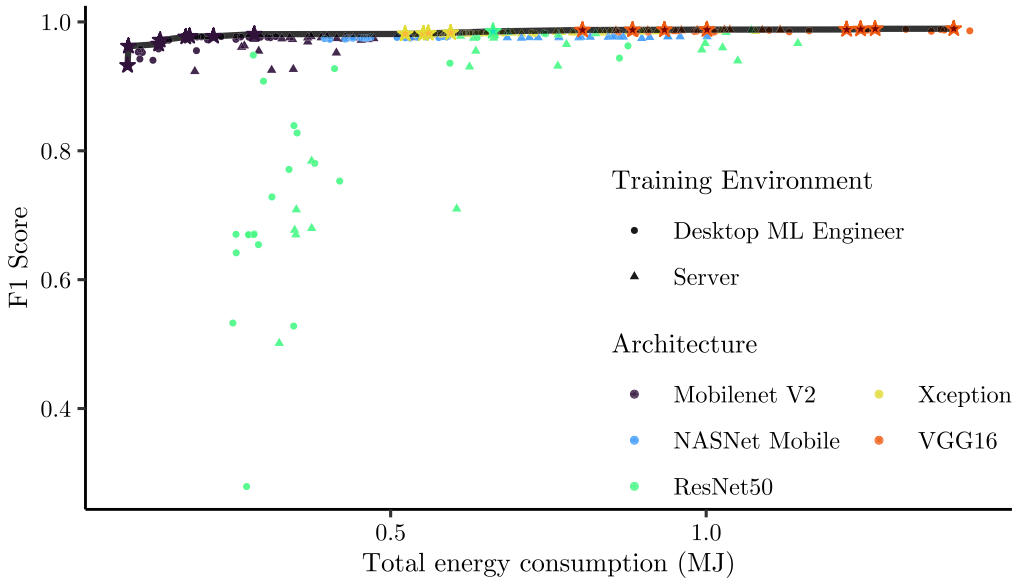

Fig. 4. Distribution of F_1 score vs. energy consumption. Instances belonging to the Pareto-optimal configurations are represented with a star.

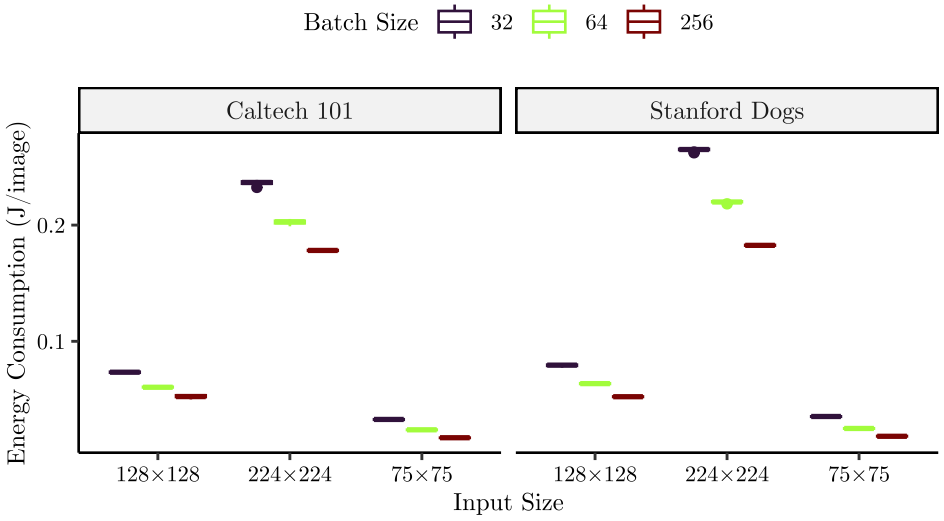

Fig. 5. Impact of batch and input size on normalized energy consumption for Inception V3 in the Desktop ML environment, split by dataset.

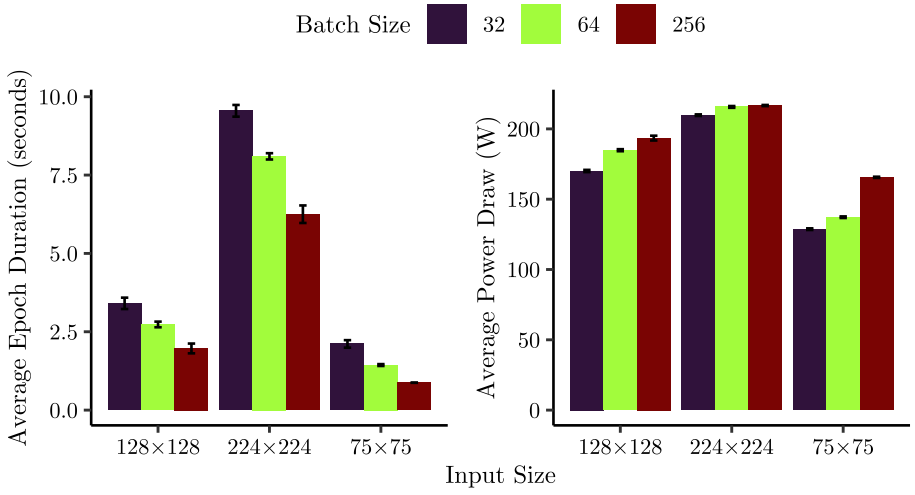

Fig. 6. Impact of batch and input size on average epoch duration and average power consumption for the Desktop ML training environment and Caltech 101 dataset.

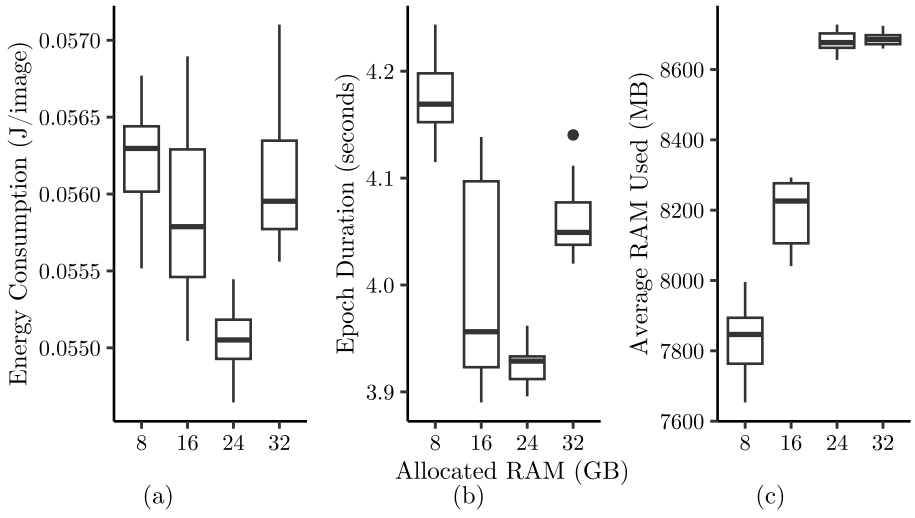

Fig. 7. Effect of allocated RAM in; (a) normalized energy consumed, (b) epoch duration, and (c) average RAM used in the Server environment.

Moving to the effect of allocated RAM, Fig. 7 presents boxplots of normalized training energy consumption (J/image), epoch duration (seconds), and average RAM usage (GB), grouped by total allocated RAM (GB). The visual trends indicate a moderate reduction in both energy consumption and epoch time as allocated RAM increases, although the improvements diminish beyond 24 GB.

Notably, performance gains are non-linear. Increasing allocation beyond a certain point can be negligible or even counterproductive, as shown by the overlapping distributions across groups. For example, the energy consumption distributions for 8 GB, 16 GB, and 32 GB overlap substantially. Similarly, epoch duration for 16 GB closely overlaps with that of 24 GB and 32 GB. In terms of average RAM used, the 24 GB and 32 GB groups are nearly identical.

Statistical analysis supports these visual observations. Allocated RAM had a significant effect on energy consumption per image ($F = 59.41$, $p < .001$), explaining 63% of the variance ($\eta^2 = 0.63$). Tukey-adjusted comparisons revealed a significant reduction in energy consumption from 8 GB to 24 GB ($p < .001$), but no significant benefit from further increasing to 32 GB ($p = .30$).

Epoch duration was also significantly influenced by allocated RAM ($F = 144.98$, $p < .001$), accounting for 81% of the variance ($\eta^2 = 0.81$). While increasing from 8 GB to 24 GB led to significant speedups ($p < .001$), a further increase to 32 GB slightly increased duration, albeit still statistically significant ($p < .001$).

As expected, allocated RAM strongly affected average RAM usage during training ($F = 1,087.59$, $p < .001$, $\eta^2 = 0.97$). However, RAM usage plateaued between 24 GB and 32 GB, with no statistically significant difference between them ($p = .99$).

Answer to $RQ_{3.1}$

Batch size, input size, and allocated RAM are all significant predictors of energy consumption during training. Larger batch sizes consistently reduce the energy per image by decreasing training time, while larger input sizes significantly increase energy consumption due to higher computational demands. Allocated RAM also influences energy use, with improvements observed up to a threshold. Beyond this point, additional RAM yields diminishing or even adverse effects. Among the parameters, input size is the most influential factor in energy modeling, followed by batch size and allocated RAM.

(2) Results $RQ_{3.2}$: Predicting training energy consumption. To develop effective models for predicting energy consumption during training, we first investigate temporal and power-related stability within training runs.

We begin by assessing whether epoch duration is consistent across executions. As shown in Fig. 6, the mean and standard deviation of epoch durations for the Inception V3 model on Caltech101 indicate very low variability. Narrow error bars across batch and input size combinations suggest strong temporal stability. This observation generalizes across datasets, models, and environments, indicating that epoch duration remains highly consistent under fixed conditions.

Next, we explore power usage patterns across training runs. Assuming that power stabilizes after an initial warm-up period, we apply semantic segmentation to detect when steady-state behavior emerges. Fig. 8 reveals that in at least 90% of the runs, power consumption stabilizes by epoch ten. Thus, we define epoch ten as the "stabilizing epoch" for subsequent analysis.

Given the stability of both epoch duration and power consumption, we propose the Stable Training Epoch Projection (STEP) methods:

1. Stable Training Epoch Power Projection (STEP-P):

$$E = \overline{P}_w \times t;\ 1 \le w \le s \tag{1}$$

where $\overline{P}_w$ is the average power over a window of w stable epochs, t is the total training time, and s is the total number of stable epochs.

2. Stable Training Epoch Energy Projection (STEP-E):

$$E = E_w \times \frac{s}{w} + E_u;\ 1 \le w \le s \tag{2}$$

where $\overline{E}_w$ is the total energy of a window of w stable epochs, and E_u is the energy from unstable epochs.

Fig. 9 (top) shows the impact of window size w on estimation accuracy, measured using RMSE (in kilojoules). A window size of five already achieves strong performance, with RMSE values of 7.34 for STEP-P and 7.88 for STEP-E.

We also examine the influence of the stabilizing epoch (bottom chart in Fig. 9). Results confirm that both methods achieve low, stable RMSE values after epoch 10. This aligns with the previously identified stabilization point.

Finally, using a randomly selected window after the stabilizing epoch (rather than starting precisely at epoch 10) slightly improves estimation accuracy: RMSE drops to 6.96 for STEP-P and 5.13 for STEP-E—representing modest improvements of 0.34 and 2.75 kJ, respectively.

While the STEP methods demonstrate strong predictive accuracy, they rely on runtime measurements and are therefore unsuitable for estimating energy consumption before training. To address this limitation, we introduce the Pre-training Regression-based Estimation (PRE)

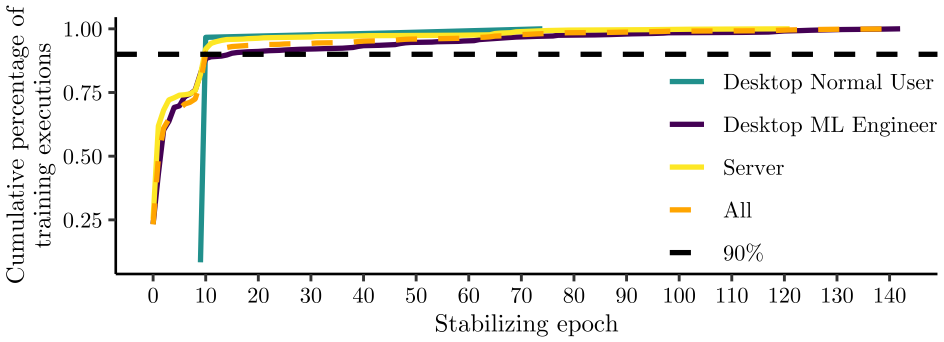

Fig. 8. Cumulative percentage of training executions that stabilize after the nth epoch in the different training environments.

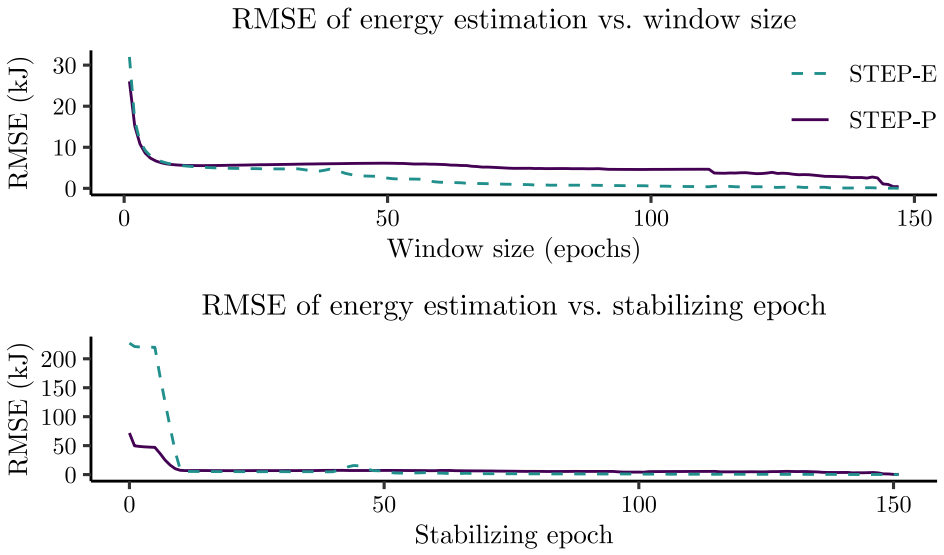

Fig. 9. Effect of changing w and the stabilizing epoch on the RMSE of the STEP methods. The top chart shows the progression of the RMSE when increasing w from the stabilizing epoch. The bottom chart shows the same progression when $w = 5$ and the stabilizing epoch is varied.

methods, which do not require in-process measurements and can be used in advance. One predicts the average power draw over the entire training execution (PRE-P), while the other estimates the average energy consumption per stable epoch (PRE-E).

These models take as input a set of parameters typically available to practitioners before training begins. These include the number of CUDA cores on the GPU, total GPU RAM, the GPU's thermal design power (TDP), total system RAM, the model's GFLOPs, the number of training and validation samples, image width and height, and the batch size.

This approach differs from existing estimation tools such as MLCO2, which estimates energy as:

$$E = t \times P_g \tag{3}$$

where t is the runtime in hours and P_g is the GPU's TDP. Similarly, GA estimates energy as:

$$E = t \times (n_c \times P_c \times u_c + n_g \times P_g \times u_g + n_m \times P_m) \times PUE \times k \tag{4}$$

where n_c, n_g, and n_m are the counts of CPU cores, GPU cores, and available memory (GB), respectively; P_c and P_g are the TDPs of the CPU and GPU; u_c and u_g are their usage factors (between 0 and 1); P_m is RAM power draw, set to 0.3725 W per GB; PUE is the power usage effectiveness of the data center; and k accounts for repeated runs.

Tables 5 and 6 summarize the performance of four regression models — Linear, Ridge, Kernel Ridge, and Support Vector Machines (SVM) — evaluated using 5-fold cross-validation on 70% of the dataset. For PRE-E, we consider two variants: one that includes energy from the initial (unstable) epochs and one that excludes it. All models perform well; however, Kernel Ridge regression consistently yields the best results across all evaluation metrics. Accordingly, we select it as our final model. For PRE-E, we select the variant trained with the initial epochs included, as it achieves slightly superior performance on two of three metrics.

Table 5
Performance of the PRE-P models in the 5-fold cross-validation.

Regression method	R^2	RMSE	MAE
Linear	0.83	28.54	21.25
Ridge	0.82	28.60	21.27
Kernel Ridge	**0.99**	**7.85**	**4.38**
SVM	0.96	13.21	6.03

Table 6
Performance of the PRE-E models in the 5-fold cross-validation.

Regression method	R^2	RMSE	MAE	Uses initial epochs?
Linear	0.83	1.62	0.97	True
Ridge	0.84	1.60	0.95	True
Kernel Ridge	0.96	**0.74**	**0.31**	True
SVM	0.95	0.86	0.38	True
Linear	0.84	1.76	1.03	False
Ridge	0.84	1.74	1.02	False
Kernel Ridge	**0.98**	0.78	0.33	False
SVM	0.95	0.93	0.40	False

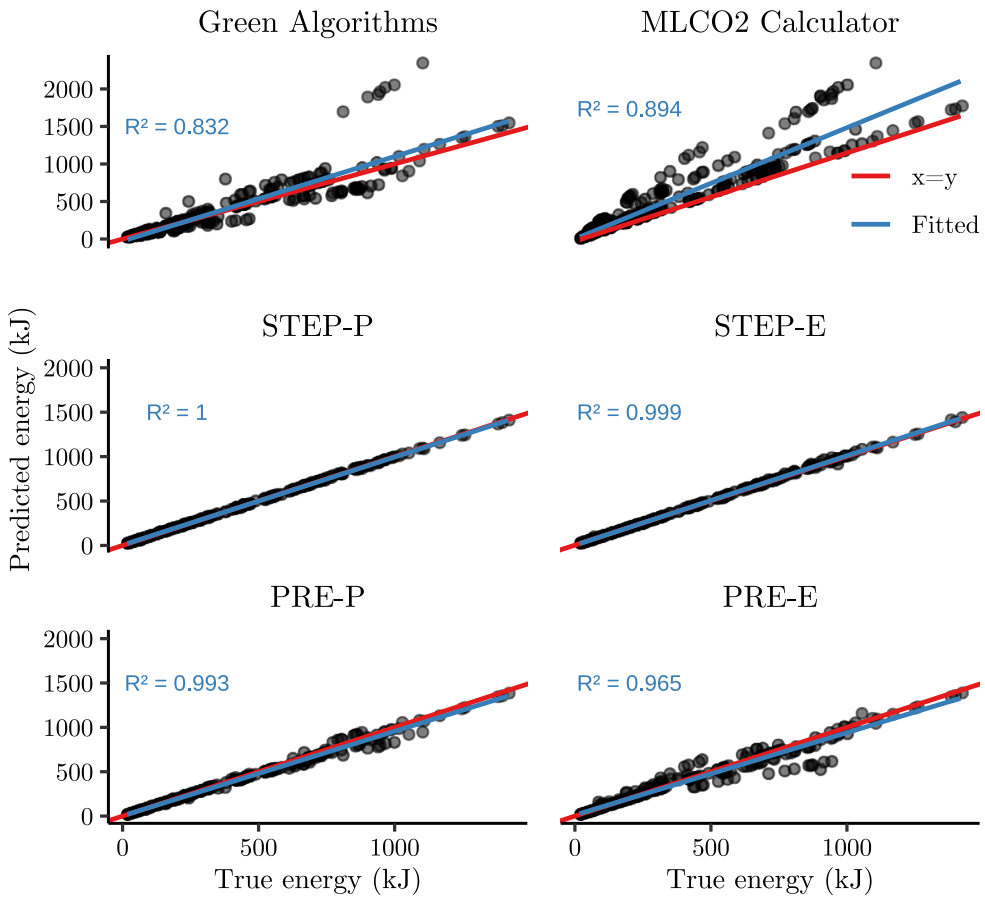

Fig. 10. Predicted energy vs. real energy using six different energy prediction methods. The blue line shows the fitted estimator. The red line is the perfect fit for the true energy. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The proposed PRE methods need an additional step to make their predictions actionable. For PRE-P, the predicted average power must be multiplied by the expected total training duration. Likewise, the output of PRE-E must be multiplied by the anticipated number of training epochs. This step transforms per-unit predictions into complete energy-consumption estimates for the entire training process.

Answer to $RQ_{3.2}$

Training epochs exhibit highly stable behavior in both duration and power under fixed settings. We identify the 10th epoch as a reliable stabilization point and introduce the STEP methods, which can accurately predict total training energy using data from only a few stable epochs. Additionally, we propose the PRE methods, based on regression models, that estimate energy consumption prior to training using readily available hardware and model parameters.

(3) Results $RQ_{3.3}$: Evaluation of the estimation methods.

To evaluate the quality of the online and offline estimation methods, we test them on 30% of the reserved evaluation data and compare them with MLCO2 and GA.

The STEP methods yield the lowest RMSE, with STEP-P reporting 5.50 kJ and STEP-E 7.97 kJ. PRE methods follow with PRE-P reporting an RMSE of 30.36 kJ, and PRE-E 58.98 kJ. The two methods previously proposed in the literature yield RMSE values of 159.83 kJ (GA) and 185.39 kJ (MLCO2). These two methods exceed the RMSE of the worst-performing method in our proposed set.

To further analyze the performance of the different methods, we compare the predicted energy with the true energy. In Fig. 10, we see how the GA, MLCO2, and PRE-E have a noticeable spread from the ideal output. MLCO2 is biased to overestimate energy consumption consistently. Similarly, GA is slightly skewed toward overestimation. However, we see that it frequently reports underestimations as well. Instead, PRE-E is slightly skewed to underestimate energy consumption. Nevertheless, it also reports overestimations that deviate substantially from the true energy.

Regarding the three remaining methods, we observe minimal deviations from the actual values in their reported energy consumption. Indeed, the STEP methods nearly match the ideal values. This is unsurprising, as they benefit from estimating energy from the actual measurements obtained during training.

Answer to $RQ_{3.3}$

The proposed methods achieve two- to threefold lower error rates, even for the worst-performing approach. Moreover, both STEP and PRE methods yield fairly accurate estimates, offering data scientists accessible methods to obtain reliable energy estimates both before and during training.

5. Discussion

5.1. Model architecture and training environment: fiends or foes?

A central outcome of this work is the empirical confirmation that energy efficiency in deep learning training is fundamentally an interaction effect between model architecture and training environment. While previous studies have often evaluated architectures or hardware

platforms in isolation, our results show that such approaches overlook critical dependencies.

Lightweight architectures, such as MobileNet V2, achieve high energy efficiency when trained on moderately provisioned environments, but become less efficient on server-class hardware due to underutilization. Conversely, computationally intensive architectures, such as VGG-16, only benefit from powerful hardware under specific conditions and otherwise incur disproportionately high energy costs. These patterns explain why no single training environment consistently outperforms others across architectures.

Importantly, our findings challenge the common assumption that scaling up hardware inherently leads to more energy-efficient training. In most configurations, the Server environment consumed more energy than the Desktop ML for the same model, despite offering greater computational capacity. This behavior appears to stem from mismatches between model complexity and available hardware resources, in which surplus GPU capacity remains idle while still contributing to baseline power consumption. This observation aligns with the interaction effects reported in our statistical analysis and is consistent with findings from prior work, such as Husom et al. [59].

These insights underline the need for collaboration between data scientists and software engineers when designing DL training pipelines. Data scientists may select models based on accuracy or transferability, while engineers can ensure that computational resources align with model requirements. This multidisciplinary cooperation is especially important when considering green AI practices.

The interaction between model architecture and training environment observed in our results has particularly strong implications for cloud-based and distributed training paradigms. In cloud settings, lightweight models trained on highly provisioned instances may lead to systematic underutilization and unnecessary energy consumption, whereas heavier architectures may benefit only under specific conditions. These effects are likely amplified in environments where resources are dynamically allocated, where runtime reduction and throughput are prioritized. The interaction effects observed in this study indicate that these strategies may inadvertently increase energy consumption when scaling does not align with model computational demands. A similar claim is given by Tschand et al. [18], where they observe that horizontal scaling offers diminishing returns after a certain threshold. Similarly, in federated and edge-based learning scenarios, where training occurs across heterogeneous devices, a single, fixed model architecture is unlikely to be energy-efficient across all participants. FL frameworks often select clients based on availability, connectivity, or data volume. Our results suggest that incorporating hardware-aware and energy-aware criteria into client selection could substantially reduce overall energy consumption. Together, these observations suggest that static provisioning and one-size-fits-all model designs are fundamentally misaligned with energy-efficient training. Following the line of thought of Abad et al. [60], we suggest that serverless or adaptive computing infrastructures may offer energy-efficient alternatives. These systems can dynamically allocate hardware based on a model's real-time requirements, reducing reliance on manual configuration and domain expertise.

Our results align with the findings of Georgiou et al. [61], who observed differences in energy and performance costs across DL frameworks. Our results also show that allocating more RAM does not guarantee improvements in training speed or energy efficiency. Over-allocation wastes energy and resources—especially in shared Server environments, where unused RAM could benefit other users. However, determining the precise RAM requirements for a model in advance remains challenging. There is a clear need for intelligent tooling to guide users in selecting optimal resource configurations.

Ultimately, the ability to model and understand the relationship between training environments and model architectures has significant implications for reducing carbon emissions and optimizing resource usage. This calls for the development of standardized AI energy benchmarks, as also proposed by Shi et al. [62]. Such benchmarks would enable consistent evaluation of the energy profiles of DL models across environments and assist practitioners in making informed deployment decisions.

Efforts like the reporting guidelines from Castaño et al. [63] for Hugging Face, which include carbon impact metrics, are steps in the right direction. Future pre-trained models should recommend optimal hardware configurations, and DL frameworks should go further by integrating energy monitoring and transparently reporting actual resource utilization.

Take-away 1

The most energy-efficient training environment is one that closely matches the model's computational requirements. To reduce the carbon footprint of DL projects, training environments should be carefully selected or dynamically adapted to fit the resource profile of the model being trained.

5.2. Server vs. Desktop, which environment to choose?

While Desktop environments require the upfront purchase of a GPU, they can offer substantial energy savings, provided subsequent model architectures are carefully chosen to match the hardware. Our results show that, for certain models, Desktop environments consistently consume less energy than Server environments. This reduced energy footprint can translate into lower operational costs, particularly when electricity prices are high. When accounting for the long-term cost of ownership — including the avoided expenses of server maintenance, infrastructure, and storage — Desktop environments can become economically competitive, with the added benefit of promoting accessibility and resource efficiency for smaller organizations and individual researchers.

However, the energy efficiency of training depends significantly on the alignment between models computational requirements and the available hardware. For large, complex models or workflows that demand substantial computational flexibility, Server environments offer a clear advantage. These environments typically support a wider range of GPUs and configurations, making it easier to match model complexity to available compute power without requiring new hardware purchases. This adaptability is especially valuable in organizational settings where computational workloads vary across projects.

From an environmental perspective, while Desktop setups can offer lower operational energy consumption, one must also consider the embodied carbon footprint of GPUs. Server environments can mitigate this impact by supporting shared, sustained use of hardware resources, thereby improving the overall utilization of energy-intensive components. Moreover, server-class infrastructures can often be hosted in geographically optimized data centers powered by cleaner energy sources [5], a flexibility not achievable with local Desktop systems.

Take-away 2

The decision between Server and Desktop should be guided by model complexity, training frequency, and both economic and environmental resource amortization. The most sustainable strategy may involve a hybrid approach that combines the agility of Desktop environments for lightweight, frequent training tasks with the scalability and efficiency of Server infrastructures for heavier workloads.

5.3. Are we using the correct estimators for energy consumption?

Our study reveals that widely adopted estimation methods and assumptions often fail to deliver accurate predictions of energy consumption in DL. One such assumption is that the number of FLOPs required by a model is a reliable proxy for its energy usage. While increased FLOPs generally correlate with higher energy demands, our results reveal substantial variation across training environments. For instance, although Xception requires nearly eight times as many FLOPs as NASNet Mobile, both consume almost the same energy in the Server environment—despite Xception being more energy-intensive in the Desktop ML environment. This discrepancy highlights a key limitation of FLOPs as a standalone estimator: it does not account for characteristics of the training environment.

These findings align with prior critiques in the literature. Schwartz et al. [7] and Lacoste et al. [28] suggest FLOPs as a proxy for energy, while others caution against hardware-agnostic metrics [64–66]. Our results reinforce the latter view, emphasizing the tight and complex interplay between model architecture, training environment, and energy behavior.

Another common estimation method assumes the TDP of a GPU approximates its average power draw during training [17,28,29]. Although we find this method tends to overestimate energy use, making it conservative, it still introduces significant uncertainty. Specifically, it hampers fair comparisons across models, as one could easily mistake an overestimated energy footprint for a more energy-intensive model, when in reality, it may be more efficient.

This problem intensifies when TDP values are scaled by GPU utilization—an increasingly common practice for incorporating GPU load into estimates. While this approach intends to improve accuracy, it can be more problematic than using unscaled TDP, as it introduces the risk of both over- and underestimation. Our analysis shows that power consumption does not consistently scale linearly with GPU usage, and the nature of this relationship varies across environments. For example, scaling TDP by GPU usage overestimated power in the Desktop ML environment, but underestimated it in the Server environment. These inconsistencies highlight the unreliability of TDP-based methods for estimating energy consumption, especially in cross-system comparisons.

These limitations have significant implications for evaluating the environmental sustainability of DL models. Platforms such as Hugging Face encourage researchers to report CO_2 emissions and recommend tools like CodeCarbon or MLCO2. While real-time estimators, such as CodeCarbon, can provide accurate measurements, their adoption often requires modifying training scripts or relying on specific platforms (e.g., AutoTrain), barriers that many practitioners do not overcome. Moreover, practitioners may also wish to report the CO_2 emissions of older models for which no real-time measurements exist. In practice, this leads many to rely on offline tools such as MLCO2, which, as our results demonstrate, frequently misestimate energy use and distort comparisons.

This challenge is evident in the Hugging Face model repository: as of March 2023, fewer than 1% of the 170,000 hosted models had reported emissions [63]. If creators were to update their models retroactively, roughly 168,000 would depend on tools such as MLCO2 or GA—underscoring the risk of widespread reliance on biased methods.

Instead, we advocate for a data-driven approach. Our findings indicate that such methods yield more accurate predictions by capturing the complex interactions among the model, the environment, and energy use. This perspective aligns with broader software engineering efforts to identify robust proxies for energy consumption [67], a challenge that remains largely unresolved.

Take-away 3

Estimating energy consumption using FLOPs or scaled TDP is unreliable because these metrics ignore the complex interplay between model architecture and the training environment. Our findings show that accurate estimation requires data-driven, context-aware approaches that reflect both system and model characteristics.

5.4. Data-driven energy estimation

In this study, we introduced several data-driven methods for estimating energy consumption that can improve the reliability and reproducibility of future work in Green in AI.

We first proposed the STEP methods that require only a fraction of the total training process to yield accurate results. While one might argue that it is preferable to measure the entire training run using real-time energy estimators such as pynvml, RAPL, or CodeCarbon — especially since these tools impose minimal overhead — we believe the value of our STEP methods lies elsewhere. Specifically, they are particularly useful in cases where retraining is infeasible, such as with large-scale models like GPT-4 or BLOOM. For instance, BLOOM's training spanned more than 1.08 million GPU hours, making a second run solely for energy estimation impractical. Yet, as Luccioni et al. [16] demonstrated, such models are often evaluated using the inaccurate TDP-based assumptions. Instead, using STEP-P, one can estimate energy consumption by measuring just a small, representative segment of the training process, offering a much more reliable alternative.

Following the idea of Reguero et al. [27], an interesting extension of the STEP approach is energy-aware early stopping. Much like data scientists monitor training loss or accuracy to determine when to halt training, energy consumption could be incorporated as a real-time constraint. By observing both energy usage and evaluation metrics during initial epochs, one could estimate the projected energy cost of continuing training relative to the model's expected improvement. This opens the door to developing smarter, energy-aware training pipelines.

Inspired by the STEP methods, we proposed the PRE methods. While slightly less accurate, these methods offer practical advantages: they require no model retraining and depend only on a small set of readily available parameters (e.g., model architecture, batch size, input resolution, GPU type). Compared with methods such as MLCO2, which rely on coarse assumptions, our PRE methods provide more reliable, context-sensitive estimates with minimal user effort.

Importantly, both STEP and PRE can support energy-aware task scheduling and resource allocation in modern training infrastructures. In shared clusters, cloud platforms, or adaptive computing environments, these estimators could be incorporated into scheduling heuristics to guide decisions about where and when training jobs should be executed. For example, PRE could be used to predict a job's energy footprint across different hardware configurations prior to execution, enabling schedulers to select the most energy-efficient placement. Similarly, STEP-based estimates obtained early during execution could inform dynamic rescheduling, throttling, or early termination of jobs whose projected energy costs exceed acceptable thresholds. In this sense, our methods provide not only estimation capabilities but also building blocks for energy-aware orchestration policies.

PRE methods could also contribute meaningfully to initiatives like Hugging Face's AI Energy Score [68]. At present, the score focuses exclusively on inference energy due to the lack of standardized, verifiable training consumption data. Our methods could help close this gap by providing a common framework for estimating training energy from transparent, repeatable inputs—without requiring developers to share proprietary logs or rerun models.

However, a key limitation of the PRE approach is generalizability, which depends heavily on the availability of diverse, high-quality training energy data. To address this, we advocate for greater transparency

and accessibility in profiling practices. Ideally, publicly released models should be accompanied by their energy and power profiling logs. Lowering the technical barrier to profiling — e.g., through DL frameworks that natively support energy tracking or profilers requiring minimal code changes — could accelerate this process. Additionally, existing tools, such as CodeCarbon, could be improved by providing finer-grained data. Currently, CodeCarbon provides only cumulative energy values, thereby limiting insight into temporal patterns and model behavior over the training period.

To mitigate data scarcity, our findings suggest generating synthetic profiling data as a promising direction. One could measure a representative segment of the stable power draw, observable after a few training epochs, and then extrapolate the full energy profile using noise-injected replication. This would reduce the need for exhaustive training while still producing valuable input for pre-training energy estimators, saving time, computational resources, and energy.

6. Implications

The findings discussed above have direct implications for both the research community and practitioners involved in designing, evaluating, and operating deep learning training pipelines.

6.1. Implications for researchers

For researchers working on Green in AI and energy-aware machine learning, this study highlights the need to reconsider how training energy consumption is modeled and evaluated. First, future empirical studies should explicitly account for interactions between model architecture and the training environment. Benchmarking architectures under a single hardware setup risks drawing conclusions that do not generalize to realistic deployment scenarios.

Second, the demonstrated shortcomings of FLOPs- and TDP-based proxies call for greater methodological rigor in energy-related research. Researchers proposing new architectures, optimization techniques, or benchmarks should carefully justify their energy metrics and, where possible, complement proxy-based estimates with empirical measurements or validated estimation models.

Third, the strong performance of the STEP and PRE methods opens new directions for research on hybrid estimation approaches that combine early runtime measurements with static configuration features. Integrating such methods into experimental workflows can reduce unnecessary long training runs, aligning research practice more closely with the goals of Green in AI.

Finally, our results emphasize the importance of evaluating energy efficiency across heterogeneous and realistic environments. Expanding empirical studies beyond homogeneous GPU configurations will enhance external validity and yield findings that are more actionable for real-world systems.

6.2. Implications for practitioners

For practitioners, the primary implication is that energy-efficient training is a system-level design challenge rather than a purely model-level concern. Selecting the most accurate model or the most powerful hardware in isolation is insufficient. Instead, practitioners should aim to align model complexity with the computational capacity of the training environment.

Our results show that substantial energy savings, up to 80%, can be achieved with negligible losses in predictive performance by choosing architectures that lie on the energy–accuracy Pareto front. This insight is particularly relevant in settings where models are retrained frequently, such as continuous integration pipelines, large-scale experimentation, or hyperparameter optimization workflows.

The PRE estimation methods provide practitioners with a practical mechanism for estimating training energy consumption prior to execution, using information typically available at design time. This enables early comparison of architectural and configuration choices without incurring the cost of exploratory runs, supporting more sustainable experimentation practices.

From an operational perspective, the findings also suggest that static resource allocation strategies may be suboptimal. Adaptive provisioning and scheduling mechanisms that account for expected model complexity could reduce idle power consumption and improve energy efficiency in shared clusters and cloud-based infrastructures, thereby operationalizing Green in AI principles in practice.

7. Threats to validity

We identified and addressed several threats to validity in the design and execution of this study. Below, we outline the primary concerns and our mitigation strategies.

A key threat to conclusion validity lies in the potential low statistical power of our analyses. To mitigate this, we followed a rigorous data collection and execution protocol (see Section 3.6) and ensured a substantial sample size. In total, 1225 experimental runs were conducted, covering various combinations of factors, treatments, and repetitions, thereby increasing the reliability of our statistical inferences.

External factors, such as hardware instability or background processes, may introduce noise into energy measurements, thereby compromising their reliability. To reduce these effects, we executed each sub-experiment 30 times and ensured a clean execution environment with only essential processes running. The inherent randomness of deep learning training was addressed by fixing the seeds of all pseudo-random number generators, as recommended by the Keras documentation.[13] We also randomized the execution order of runs to minimize the impact of any temporal anomalies.

A mono-method bias could affect construct validity, as we used a single metric for both computational complexity and model accuracy. For complexity, we relied on FLOPs, which is widely used in efficient DL research [69,70]. For accuracy, we used the F_1 score because it balances precision and recall in a single, interpretable metric. Other performance metrics, such as accuracy, precision, recall, and AUC, were also recorded and are available in our **replication package** [54].

Concerning internal validity, successive experiment runs may introduce a history bias, such as increased hardware temperature influencing results. To mitigate this, we introduced a 5-minute cooldown period between runs and performed an initial warm-up task to ensure consistent hardware conditions across all executions.

Finally, for external validity, our findings are based on a limited set of five DL architectures and three training environments. While these choices reflect a range of real-world settings, from entry-level to advanced, we acknowledge that the generalizability of our results may be constrained. Nonetheless, the selected models are widely used and representative, providing a solid foundation for future work.

Another external validity threat concerns the generalizability of the results of the energy estimation methods. Both the STEP and PRE methods were trained and evaluated on a limited set of models, datasets, and training environments. While our findings demonstrate strong predictive performance within this scope, their accuracy may vary when applied to different architectures, hardware configurations, or larger-scale models. Although we employed multiple regression algorithms and cross-validation to enhance robustness, generalization to unseen settings is not guaranteed. Further studies using diverse profiling data are needed to confirm the broader applicability of these estimators.

Lastly, the specific datasets and contexts used may also impact generalizability. Results could vary with different datasets or data volumes. To address this, we report normalized energy consumption metrics (see Section 3.7). However, we recognize that metrics like GPU usage and F_1 score are inherently problem-dependent and should be interpreted within the dataset context.

[13] https://keras.io/getting_started/faq

8. Conclusion and future work

The rapid adoption of Deep Learning (DL) in computer vision has raised growing concerns about its environmental impact. In this work, we conducted a comprehensive empirical study to understand how two major design factors [71] — model architecture and training environment — jointly influence training energy consumption. By systematically varying these factors across multiple architectures and training environments, we provide evidence that energy efficiency in DL training is not an inherent property of models or hardware in isolation, but emerges from their interaction.

Beyond characterizing these interaction effects, this study contributes a set of data-driven energy estimation methods that address key limitations of existing proxy-based approaches. The proposed STEP and PRE methods significantly reduce estimation error while remaining practical for both real-time and pre-training use cases. Together, these contributions advance the methodological foundations of Green in AI by enabling more reliable, context-aware reasoning about training energy consumption without relying on oversimplified assumptions such as FLOPs or TDP.

This work positions energy consumption as a first-class system property that must be considered alongside traditional performance metrics. Our findings suggest that energy-aware decision-making should be integrated directly into model development, experimentation, and execution workflows, particularly in environments where resources are dynamically provisioned or shared.

Several directions for future work naturally follow from this perspective. First, an important next step is to integrate energy estimation with standard performance and efficiency metrics used in machine learning practice, such as training throughput, convergence speed, or cost-based objectives. Jointly analyzing accuracy, runtime, cost, and energy would enable multi-objective optimization frameworks that better reflect real-world constraints.

Second, the interaction effects identified in this study motivate tighter integration with cloud-based and adaptive computing frameworks. Incorporating STEP- and PRE-style estimators into schedulers, autoscalers, or serverless training platforms could enable energy-aware task placement, dynamic resource allocation, and early termination policies. Evaluating such integrations in realistic cloud settings remains an open and promising research avenue.

Third, while this work focused primarily on the training phase, the inference phase represents a substantial and often dominant share of the life cycle energy consumption of deployed models. Extending the proposed estimation methods to inference workloads, particularly under variable batch sizes, latency constraints, and deployment environments, would provide a more complete picture of model sustainability.

Finally, future studies should broaden the empirical scope of this work by considering additional tasks, such as object detection or segmentation, and by examining larger and more heterogeneous execution environments. Expanding publicly available datasets of fine-grained energy profiling data will be essential to improve the generalizability of data-driven estimators and to support reproducible research in sustainable AI.

Overall, this work takes a step toward bridging model-centric and system-centric views of energy efficiency, providing both empirical evidence and practical tools to support more sustainable deep learning development.

Acronyms

AI	Artificial Intelligence
CPU	Central Processing Unit
DL	Deep Learning
FEN	Forsyth Edwards Notation
FLOPs	Floating-point Operations
GA	Green Algorithms
GAMLSS	Generalized Additive Models for Location, Scale, and Shape
GLM	Generalized Linear Model
GPU	Graphical Processing Unit
HPC	High-Performance Computing
MAC	Multiply-accumulate operation
MAE	Mean Absolute Error
ML	Machine Learning
MLCO2	ML CO_2 Impact Calculator
PSF	Pragmatic Scaling Factor
PUE	Power Usage Effectiveness
RAM	Random Access Memory
RMSE	Root Mean Squared Error
RQ	Research Question
TDP	Thermal Design Power

CRediT authorship contribution statement

Santiago del Rey: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Luís Cruz:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Xavier Franch:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization. **Silverio Martínez-Fernández:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We provide all the data and code with the replication package cited in the paper: https://zenodo.org/records/17041051.

References

[1] A. Bick, A. Blandin, D.J. Deming, The rapid adoption of generative AI, in: Working Paper Series, 2024, http://dx.doi.org/10.3386/w32966, Working Paper 32966, National Bureau of Economic Research.

[2] M. Simaremare, H. Edison, The state of generative AI adoption from software practitioners' perspective: An empirical study, in: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, 2024, pp. 106–113, http://dx.doi.org/10.1109/SEAA64295.2024.00024.

[3] M. Alenezi, M. Akour, AI-driven innovations in software engineering: A review of current practices and future directions, Appl. Sci. 15 (3) (2025) http://dx.doi.org/10.3390/app15031344.

[4] A. Dario, H. Danny, S. Girish, C. Jack, B. Greg, S. Ilya, AI and compute, 2018, URL https://openai.com/blog/ai-and-compute/#addendum.

[5] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D.R. So, M. Texier, J. Dean, The carbon footprint of machine learning training will plateau, then shrink, Computer 55 (7) (2022) 18–28, http://dx.doi.org/10.1109/MC.2022.3148714.

[6] Y. Yu, J. Wang, Y. Liu, P. Yu, D. Wang, P. Zheng, M. Zhang, Revisit the environmental impact of artificial intelligence: The overlooked carbon emission source? Front. Environ. Sci. Eng. 18 (12) (2024) 1–5, http://dx.doi.org/10.1007/s11783-024-1918-y.

[7] R. Schwartz, J. Dodge, N.A. Smith, O. Etzioni, Green AI, Commun. ACM 63 (2020) 54–63, http://dx.doi.org/10.1145/3381831.

[8] M. Gutiérrez, M.Á. Moraga, F. García, C. Calero, Green IN artificial intelligence from a software perspective: State-of-the-art and green decalogue, ACM Comput. Surv. 57 (3) (2024) 64:1–64:30, http://dx.doi.org/10.1145/3698111, URL https://dl.acm.org/doi/10.1145/3698111.

[9] F. Chen, S. Li, J. Han, F. Ren, Z. Yang, Review of lightweight deep convolutional neural networks, Arch. Comput. Methods Eng. 31 (4) (2024) 1915–1937, http://dx.doi.org/10.1007/s11831-023-10032-z.

[10] S. Cong, Y. Zhou, A review of convolutional neural network architectures and their optimizations, Artif. Intell. Rev. 56 (3) (2023) 1905–1969, http://dx.doi.org/10.1007/s10462-022-10213-5.

[11] ISO, ISO/IEC TR 20226:2025, 2025, URL https://www.iso.org/standard/86177.html. (Accessed 25 August 2025).

[12] Green Software Foundation, Software carbon intensity (SCI) specification, 2025, URL https://sci.greensoftware.foundation/. (Accessed 29 August 2025).

[13] Spanish Association for Standardization, Comité CTN 71/SC 42 inteligencia artificial y big data, 2025, URL https://www.en.une.org/encuentra-tu-norma/comites-tecnicos-de-normalizacion/comite?c=CTN+71/SC+42. (Accessed 25 August 2025).

[14] DIN, Home - Resource-efficient Software, 2025, URL https://din.one/site/rs. (Accessed 25 August 2025).

[15] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, J. Dean, Carbon emissions and large neural network training, 2021, arXiv:2104.10350.

[16] A.S. Luccioni, S. Viguier, A.-L. Ligozat, Estimating the carbon footprint of BLOOM, a 176B parameter language model, J. Mach. Learn. Res. 24 (253) (2023) 1–15.

[17] A.S. Luccioni, A. Hernandez-Garcia, Counting carbon: A survey of factors influencing the emissions of machine learning, 2023, arXiv:2302.08476.

[18] A. Tschand, A.T.R. Rajan, S. Idgunji, A. Ghosh, J. Holleman, C. Kiraly, P. Ambalkar, R. Borkar, R. Chukka, T. Cockrell, O. Curtis, G. Fursin, M. Hodak, H. Kassa, A. Lokhmotov, D. Miskovic, Y. Pan, M.P. Manmathan, L. Raymond, T.S. John, A. Suresh, R. Taubitz, S. Zhan, S. Wasson, D. Kanter, V.J. Reddi, MLPerf power: Benchmarking the energy efficiency of machine learning systems from μWatts to MWatts for sustainable AI, in: 2025 IEEE International Symposium on High Performance Computer Architecture, HPCA, 2025, pp. 1201–1216, http://dx.doi.org/10.1109/HPCA61900.2025.00092.

[19] R. Verdecchia, J. Sallou, L. Cruz, A systematic review of Green AI, WIREs Data Min. Knowl. Discov. 13 (4) (2023) e1507, http://dx.doi.org/10.1002/widm.1507.

[20] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, B. Vorster, Deep learning in the automotive industry: Applications and tools, in: IEEE Big Data, IEEE, 2016, pp. 3759–3768, http://dx.doi.org/10.1109/BigData.2016.7841045.

[21] R. Caspart, S. Ziegler, A. Weyrauch, H. Obermaier, S. Raffeiner, L.P. Schuhmacher, J. Scholtyssek, D. Trofimova, M. Nolden, I. Reinartz, F. Isensee, M. Götz, C. Debus, Precise energy consumption measurements of heterogeneous artificial intelligence workloads, in: H. Anzt, A. Bienz, P. Luszczek, M. Baboulin (Eds.), High Performance Computing. ISC High Performance 2022 International Workshops, Springer International Publishing, 2022, pp. 108–121.

[22] D. Li, X. Chen, M. Becchi, Z. Zong, Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs, in: BDCloud-SocialCom-SustainCom, IEEE, 2016, pp. 477–484, http://dx.doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.76.

[23] T. Yarally, L. Cruz, D. Feitosa, J. Sallou, A. van Deursen, Uncovering energy-efficient practices in deep learning training: Preliminary steps towards green AI, in: 2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI, CAIN, 2023, pp. 25–36, http://dx.doi.org/10.1109/CAIN58948.2023.00012.

[24] N.K. Jha, S. Mittal, G. Mattela, The ramifications of making deep neural networks compact, in: 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems, VLSID, (ISSN: 2380-6923) 2019, pp. 215–220, http://dx.doi.org/10.1109/VLSID.2019.00056, URL https://ieeexplore.ieee.org/document/8711062.

[25] S. Aquino-Brítez, P. García-Sánchez, A. Ortiz, D. Aquino-Brítez, S. Aquino-Brítez, P. García-Sánchez, A. Ortiz, D. Aquino-Brítez, Towards an energy consumption index for deep learning models: A comparative analysis of architectures, GPUs, and measurement tools, Sensors 25 (3) (2025) http://dx.doi.org/10.3390/s25030846.

[26] S.N. Gowda, X. Hao, G. Li, S.N. Gowda, X. Jin, L. Sevilla-Lara, Watt for what: Rethinking deep learning's energy-performance relationship, in: A. Del Bue, C. Canton, J. Pont-Tuset, T. Tommasi (Eds.), Computer Vision – ECCV 2024 Workshops, Springer Nature Switzerland, Cham, 2025, pp. 388–405, http://dx.doi.org/10.1007/978-3-031-92089-9_24.

[27] Á.D. Reguero, S. Martínez-Fernández, R. Verdecchia, Energy-efficient neural network training through runtime layer freezing, model quantization, and early stopping, Comput. Stand. Interfaces 92 (2025) 103906, http://dx.doi.org/10.1016/j.csi.2024.103906.

[28] A. Lacoste, A. Luccioni, V. Schmidt, T. Dandres, Quantifying the carbon emissions of machine learning, 2019, arXiv:1910.09700.

[29] L. Lannelongue, J. Grealey, M. Inouye, Green algorithms: quantifying the carbon footprint of computation, Adv. Sci. 8 (12) (2021) 2100707, http://dx.doi.org/10.1002/advs.202100707.

[30] A.C. Newkirk, J. Fernandez, J. Koomey, I. Latif, E. Strubell, A. Shehabi, C. Samaras, Empirically-calibrated H100 node power models for accurate AI training energy estimation, Environ. Res.: Energy 2 (4) (2025) 045016, http://dx.doi.org/10.1088/2753-3751/ae2486.

[31] C.E. Tripp, J. Perr-Sauer, J. Gafur, A. Nag, A. Purkayastha, S. Zisman, E.A. Bensen, Measuring the energy consumption and efficiency of deep neural networks: An empirical analysis and design recommendations, 2024, http://dx.doi.org/10.48550/arXiv.2403.08151, arXiv:2403.08151.

[32] L. Cruz, J.P. Fernandes, M.H. Kirkeby, S. Martínez-Fernández, J. Sallou, H. Anwar, E. Barba Roque, J. Bogner, J. Castaño, F. Castor, A. Chasmawala, S. Cunha, D. Feitosa, A. González, A. Jedlitschka, P. Lago, H. Muccini, A. Oprescu, P. Rani, J. Saraiva, F. Sarro, R. Selvan, K. Vaidhyanathan, R. Verdecchia, I.P. Yamshchikov, Greening AI-enabled systems with software engineering: A research agenda for environmentally sustainable AI practices, SIGSOFT Softw. Eng. Notes 50 (3) (2025) 14–23, http://dx.doi.org/10.1145/3743095.3743099.

[33] J.-W. Chung, J.J. Ma, R. Wu, J. Liu, O.J. Kweon, Y. Xia, Z. Wu, M. Chowdhury, The ML.ENERGY benchmark: Toward automated inference energy measurement and optimization, 2025, http://dx.doi.org/10.48550/arXiv.2505.06371, URL http://arxiv.org/abs/2505.06371.

[34] L. Bouza, A. Bugeau, L. Lannelongue, How to estimate carbon footprint when training deep learning models? A guide and review, Environ. Res. Commun. 5 (11) (2023) 115014, http://dx.doi.org/10.1088/2515-7620/acf81b.

[35] A. Guldner, R. Bender, C. Calero, G.S. Fernando, M. Funke, J. Gröger, L.M. Hilty, J. Hörnschemeyer, G.-D. Hoffmann, D. Junger, T. Kennes, S. Kreten, P. Lago, F. Mai, I. Malavolta, J. Murach, K. Obergöker, B. Schmidt, A. Tarara, J.P. De Veaugh-Geiss, S. Weber, M. Westing, V. Wohlgemuth, S. Naumann, Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components—Green Software Measurement Model (GSMM), Future Gener. Comput. Syst. 155 (2024) 402–418, http://dx.doi.org/10.1016/j.future.2024.01.033.

[36] A. Noureddine, PowerJoular and joularjx: Multi-platform software power monitoring tools, in: 2022 18th International Conference on Intelligent Environments, IE, 2022, pp. 1–4, http://dx.doi.org/10.1109/IE54923.2022.9826760.

[37] J. Sallou, L. Cruz, T. Durieux, EnergiBridge: Empowering software sustainability through cross-platform energy measurement, 2023, arXiv:2312.13897.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255, http://dx.doi.org/10.1109/CVPR.2009.5206848.

[39] A. Krizhevsky, G. Hinton, et al., Learning Multiple Layers of Features from Tiny Images (Master's thesis), University of Toronto, 2009.

[40] O.M. Parkhi, A. Vedaldi, A. Zisserman, C.V. Jawahar, Cats and dogs, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3498–3505, http://dx.doi.org/10.1109/CVPR.2012.6248092.

[41] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, S. Belongie, The inaturalist species classification and detection dataset, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2018, pp. 8769–8778.

[42] L. Bossard, M. Guillaumin, L. Van Gool, Food-101 – mining discriminative components with random forests, in: Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 446–461.

[43] V.R. Basili, G. Caldiera, H.D. Rombach, The goal question metric approach, Encycl. Softw. Eng. (1994) 528–532.

[44] D. Zhang, N. Maslej, E. Brynjolfsson, J. Etchemendy, T. Lyons, J. Manyika, H. Ngo, J.C. Niebles, M. Sellitto, E. Sakhaee, Y. Shoham, J. Clark, R. Perrault, The AI index 2022 annual report, 2022, arXiv:2205.03468.

[45] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, P. Villalobos, Compute trends across three eras of machine learning, in: 2022 International Joint Conference on Neural Networks, IJCNN, (ISSN: 2161-4407) 2022, pp. 1–8, http://dx.doi.org/10.1109/IJCNN55064.2022.9891914.

[46] D. Edelman, S. Samsi, J. McDonald, A. Michaleas, V. Gadepally, An analysis of energy requirement for computer vision algorithms, in: 2023 IEEE High Performance Extreme Computing Conference, HPEC, IEEE, 2023, pp. 1–7, http://dx.doi.org/10.1109/HPEC58863.2023.10363596.

[47] N. Alizadeh, F. Castor, Green AI: A preliminary empirical study on energy consumption in DL models across different runtime infrastructures, in: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, in: CAIN '24, Association for Computing Machinery, 2024, pp. 134–139, http://dx.doi.org/10.1145/3644815.3644967.

[48] S. Rajput, T. Widmayer, Z. Shang, M. Kechagia, F. Sarro, T. Sharma, Enhancing energy-awareness in deep learning through fine-grained energy measurement, ACM Trans. Softw. Eng. Methodol. 33 (8) (2024) 211:1–211:34, http://dx.doi.org/10.1145/3680470.

[49] M. Hodak, A. Dholakia, Recent efficiency gains in deep learning: Performance, power, and sustainability, in: 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 2040–2045, http://dx.doi.org/10.1109/BigData52589.2021.9671762.

[50] F.-F. Li, M. Andreeto, M. Ranzato, P. Perona, Caltech 101, 2022, http://dx.doi.org/10.22002/D1.20086.

[51] A. Khosla, N. Jayadevaprakash, B. Yao, L. Fei-Fei, Novel dataset for fine-grained image categorization, in: First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition, 2011, p. 2.

[52] G. Wölflein, O. Arandjelović, Determining chess game state from an image, J. Imaging 7 (2021) 94, http://dx.doi.org/10.3390/jimaging7060094.

[53] Roboflow, Chess pieces dataset, 2021, URL https://public.roboflow.com/object-detection/chess-full. (Accessed 11 July 2023).

[54] S. del Rey, Estimating Deep Learning energy consumption based on model architecture and training environment - Replication package, 2024, http://dx.doi.org/10.5281/zenodo.17041051.

[55] L. Cruz, Green software engineering done right: a scientific guide to set up energy efficiency experiments, 2021, http://dx.doi.org/10.6084/m9.figshare.22067846.v1.

[56] R. Fischer, M. Jakobs, S. Mücke, K. Morik, A unified framework for assessing energy efficiency of machine learning, in: Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Springer Nature Switzerland, 2023, pp. 39–54.

[57] S. Gharghabi, Y. Ding, C.-C.M. Yeh, K. Kamgar, L. Ulanova, E. Keogh, Matrix profile VIII: Domain agnostic online semantic segmentation at superhuman performance levels, in: 2017 IEEE International Conference on Data Mining, ICDM, IEEE, 2017, pp. 117–126, http://dx.doi.org/10.1109/ICDM.2017.21.

[58] S.M. Law, STUMPY: A powerful and scalable python library for time series data mining, J. Open Source Softw. 4 (39) (2019) 1504.

[59] E.J. Husom, S. Sen, A. Goknil, Engineering carbon emission-aware machine learning pipelines, in: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, in: CAIN '24, Association for Computing Machinery, 2024, pp. 118–128, http://dx.doi.org/10.1145/3644815.3644943.

[60] C. Abad, I.T. Foster, N. Herbst, A. Iosup, Serverless computing (Dagstuhl Seminar 21201), Dagstuhl Rep. 11 (4) (2021) 34–93, http://dx.doi.org/10.4230/DagRep.11.4.34.

[61] S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, Y. Zou, Green AI: do deep learning frameworks have different costs? in: Proceedings of the 44th International Conference on Software Engineering, ICSE '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1082–1094, http://dx.doi.org/10.1145/3510003.3510221.

[62] J. Shi, Z. Yang, D. Lo, Efficient and green large language models for software engineering: Literature review, vision, and the road ahead, ACM Trans. Softw. Eng. Methodol. 34 (5) (2025) 137:1–137:22, http://dx.doi.org/10.1145/3708525.

[63] J. Castaño, S. Martínez-Fernández, X. Franch, J. Bogner, Exploring the carbon footprint of hugging face's ML models: A repository mining study, in: 2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM, IEEE, 2023, pp. 1–12, http://dx.doi.org/10.1109/ESEM56168.2023.10304801.

[64] Q. Cao, A. Balasubramanian, N. Balasubramanian, Towards accurate and reliable energy measurement of NLP models, 2020, arXiv:2010.05248.

[65] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, J. Pineau, Towards the systematic reporting of the energy and carbon footprints of machine learning, J. Mach. Learn. Res. 21 (248) (2020) 1–43.

[66] A. Asperti, D. Evangelista, E. Loli Piccolomini, A survey on variational autoencoders from a green AI perspective, SN Comput. Sci. 2 (4) (2021) 301, http://dx.doi.org/10.1007/s42979-021-00702-9.

[67] M. Weber, C. Kaltenecker, F. Sattler, S. Apel, N. Siegmund, Twins or false friends? A study on energy consumption and performance of configurable software, in: 2023 IEEE/ACM 45th International Conference on Software Engineering, ICSE, IEEE, 2023, pp. 2098–2110, http://dx.doi.org/10.1109/ICSE48619.2023.00177.

[68] S. Luccioni, B. Gamazaychikov, E. Strubell, S. Hooker, Y. Jernite, M. Mitchell, S. Chamberlin, AI Energy Score Leaderboard - December 2025, 2025, URL https://huggingface.co/spaces/AIEnergyScore/Leaderboard. (Accessed 19 December 2025).

[69] K. Neshatpour, F. Behnia, H. Homayoun, A. Sasan, ICNN: An iterative implementation of convolutional neural networks to enable energy and computational complexity aware dynamic approximation, in: DATE, IEEE, 2018, pp. 551–556, http://dx.doi.org/10.23919/DATE.2018.8342068.

[70] F. Boutros, N. Damer, M. Fang, F. Kirchbuchner, A. Kuijper, MixFaceNets: Extremely efficient face recognition networks, in: IJCB, IEEE, 2021, pp. 1–8, http://dx.doi.org/10.1109/IJCB52358.2021.9484374.

[71] S. del Rey, Software design decisions for greener machine learning-based systems, in: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, in: CAIN 2024, Association for Computing Machinery, 2024, pp. 256–258, http://dx.doi.org/10.1145/3644815.3644972.