POLITECNICO DI TORINO
Repository ISTITUZIONALE

Energy Sustainability Analysis of Deep Neural Network

Original
Energy Sustainability Analysis of Deep Neural Network / Yin, J., Chen, J., Vallero, G., Meo, M.. - (2025), pp. 340-349. (2025 International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) Barcellona (Spa) 27-31Ottobre 2025) [10.1109/MSWiM67937.2025.11309062].

Availability:
This version is available at: 11583/3010213 since: 2026-04-23T13:57:46Z

Publisher:
IEEE

Published
DOI:10.1109/MSWiM67937.2025.11309062

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

31 October 2026

Energy Sustainability Analysis of Deep Neural Network

Jun Yin
Politecnico di Torino
Turin, Italy
jun.yin@polito.it

Jingsi Chen
Politecnico di Torino
Turin, Italy
jingsi.chen@polito.it

Greta Vallero
Politecnico di Torino
Turin, Italy
greta.vallero@polito.it

Michela Meo
Politecnico di Torino
Turin, Italy
michela.meo@polito.it

***Abstract*—Artificial Intelligence (AI) applications are becoming more widespread, raising significant environmental concerns due to the high energy use required to train large Deep Neural Networks (DNNs). To address this issue, we conduct a detailed quantitative analysis of the energy consumption of various models across different training steps and datasets. Our study measures energy use at each training stage with a comprehensive review of selected models. By tracking energy consumption during each training step, we find that the backpropagation phase is the most energy-intensive. Additionally, we evaluate the power limits and performance features of Graphics Processing Units (GPUs), collecting empirical data on their behavior under different GPU power settings. Based on these insights, we explore the integration of locally installed renewable energy sources, such as solar power and battery systems, with the electrical grid to enhance the energy sustainability of GPU operations. We introduce and test an innovative approach for managing energy and computing resources, aiming to optimize energy use and reduce operational costs. Our results demonstrate that this method can reduce energy consumption by more than 40% and operational costs by almost 25%, paving the way for greener AI solutions.**

***Index Terms*—Energy Consumption, Deep Neural Networks, Green Energy, Energy Dispatch Model**

I. Introduction

Artificial Intelligence (AI) applications are widespread across various sectors such as healthcare, finance, transportation, education, and entertainment, with network management being no exception. On one hand, AI helps solve complex problems, like predictive maintenance in telecommunications networks, enabling proactive fault management and reducing downtime, as well as image recognition in medical diagnostics, fraud detection in financial systems, and autonomous driving. On the other hand, the advanced capabilities and complexity of AI models increase demand for computational resources and energy, posing significant challenges for real-world applications [1]. Efficiently managing these resources is particularly difficult, especially in environments with limited capacity, such as edge networks. These networks, located close to data sources, include computational capabilities at the edge devices to improve latency, privacy, reliability, and bandwidth for real-time applications, but face obstacles due to limited energy and memory resources hosted by these simple devices. For this reason, optimizing energy efficiency is critical, not only to address the constraints of available computational capabilities but also to reduce operational costs for mobile operators and align with strategic policies such as the recommendations of the European Commission to decrease CO_2 emissions from software assets set by the European Commission [2].

To address these challenges, similar to the sustainable practices in Radio Access Networks proposed decades ago, such as in [3]–[5], we integrate renewable energy sources into the supply system of the AI model training. In particular, the first part of the paper investigates the energy consumption of image classification AI models, and analyzes GPU performance under varying power limits. Then, an optimization problem is formalized, using real weather data and electricity pricing, to minimize operational costs during training, where the energy is provided by a solar energy system, an energy battery, and the power grid.

The main contributions of this paper are as follows:

- Through measurements of energy consumption during training processes using 20 deep learning models, we analyze the relationships between:
  - training energy consumption and hardware characteristics;
  - training energy consumption and dataset properties;
  - energy consumption across different phases of the training process, with a specific focus on the energy profile of forward propagation in deep learning models.
- Evaluation of GPU performance under varying power limit settings to identify trade-offs between model accuracy and energy efficiency.
- Construction of an Energy Dispatch Model to enhance the energy sustainability of GPUs during the training of deep learning models.

II. Related Works

The work in [6] defines the concept of GreenAI and introduces various methods to reduce carbon emissions while training AI models. In terms of energy consumption calculation, the authors in [7] propose a method for estimating the carbon footprint of Generative AI throughout its lifecycle. The objective in [8] and [9] is the estimation of the energy consumption and carbon emissions for different AI models and hardware facilities in local and cloud computing clusters, respectively. In [10], the authors analyze the performance of different GPUs

and Central Processing Units (CPUs) in deep model training scenarios, verifying that GPUs outperform CPUs. The work presented in [11] analyzes the operational characteristics of models on different hardware devices, while the authors in [12] model the energy consumption for GPUs based on an analysis of GPU architecture at a low level. The work discussed in [13] investigates the DVFS (Dynamic Voltage and Frequency Scaling) technology for GPUs and identifies the performance of GPUs under different operating conditions through DVFS adjustments. In [14], energy consumption data for different models on a single dataset are discussed, in addition to a method for predicting energy consumption based on different model structures.

Regarding the characteristics of GPUs during deep learning model training, [15] identifies the operating speeds of GPUs under different power limits.

Based on this, an optimization algorithm is proposed to adjust the power limits and batch size settings in subsequent model training to ensure that overall energy consumption remains as low as possible.

Considering all the referenced literature, this work involves modeling multiple deep learning models and selecting appropriate measurement tools to collect energy consumption data for various parts of the training process. It provided energy consumption characteristics for each part and overall model energy data. Subsequently, the collected energy data is analyzed for model structure, dataset size, and training parameter settings. Multiple potential factors influencing the energy consumption of deep learning model training are identified and analyzed using the collected data.

III. Data Collection

For monitoring the energy consumption of the training, we rent two GPUs from the AutoDL platform [16], specifically the RTX 3080 and RTX 4090. To collect timestamped measurements of power consumption, we utilize Nvidia's NVML management library [17] to record the power at each sampling point in time, with the sampling rate set to 2 ms. This study evaluates several deep learning image classification models: the pioneering AlexNet [18] model and the parameter-heavy VGG models [19], constructing three variants: VGG11, VGG13, VGG16 and VGG19. Additionally, we use three variants of the ResNet model [20]: ResNet18 ResNet34, and ResNet50.

Furthermore, based on the Inception module of the GoogleNet model [21], we employ the original GoogleNet model along with 9 modified versions by making adjustments to its structure. Lastly, to gain deeper insights into the energy consumption characteristics of the Transformer architecture [23], we develop a modified Vision Transformer model for further investigation.

We train the models we describe above using Fashion-MNIST and CIFAR-100 data-sets, see [22] and [23], respectively. To ensure that the image size does not affect the training process, all images are uniformly resized to 224×224 before the training to maintain consistency.

TABLE I
MACs of the models on different datasets (Fashion-MNIST and CIFAR-100) and Number of Samples for Training Process; Hardware H1 is RTX 3080, H2 is RTX4090

Model	SL_F(H1)	SL_C(H1)	SL_F(H2)	SL_C(H2)
AlexNet	99451	53119	115131	214768
VGG11	294067	289341	734732	968211
VGG13	452899	412981	1264659	1577272
VGG16	558496	474812	1564824	1814464
VGG19	598316	530221	547357	484773
ResNet18	124217	120549	282327	391117
ResNet34	225560	197381	441854	594548
ResNet50	234788	208876	532983	709445
GoogleNet	164526	157499	274024	512358
GoogleNet_mod1	130507	125718	160164	327206
GoogleNet_mod2	97025	92248	126834	233123
GoogleNet_mod3	132375	129061	166711	310538
GoogleNet_mod4	117001	109334	156172	265789
GoogleNet_mod5	85822	95144	119159	225984
GoogleNet_mod6	130055	124952	166220	280968
GoogleNet_mod7	180410	171590	223602	407980
GoogleNet_mod8	231119	221213	354898	521293
GoogleNet_mod9	58156	68584	98535	188343
ViT	178978	170594	141900	138601

IV. Analysis of the Training Energy Consumption

To assess the computing effort of model training, we consider the multiply–accumulate operations (MACs), a fundamental metric for computational complexity in deep learning models. It quantifies the amount of computation required by the model, serving as a key indicator of its computational workload. We measure the MACs for different models across various datasets, and we report them in Table I, where H1 stands for RTX 3080, and H2 stands for RTX4090. The MACs for the models are monitored using the *ptflops* python library [24]. In addition to MAC, Table I also reports the number of energy consumption samples collected at every sampling interval of 2 ms, providing an indication of the duration of the training. In the table, SL_F represents the length of the recorded sample data when the model is trained on the Fashion-MNIST dataset and SL_C represents the length of the recorded sample data when the model is trained on the CIFAR-100 dataset. The number of recorded energy consumption samples directly depends on the duration required to train the model with the given dataset.

1) Consumption with Different Hardware: Fig. 1 presents the power during the training of each of the considered models, using one of the mentioned datasets, on two different hardware platforms. In the figure, the x-axis represents time in seconds with each sampling interval set to 2 ms, while the y-axis represents the power at the sampling point moment. Considering the training cost and to avoid GPU memory overflow during the training process, we set each model to train for 5 epochs, with a batch size of 128 during training.

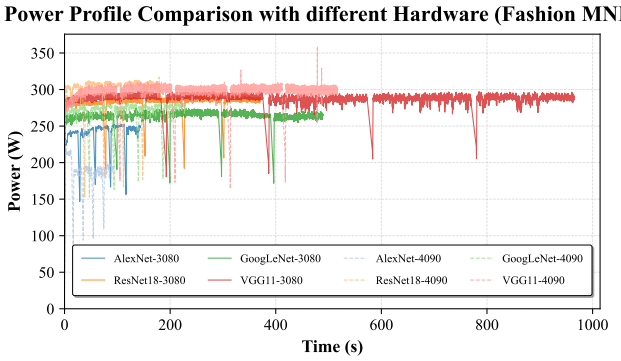


Fig. 1. Average power consumption with Different Hardwares

As shown in the figure, training on the RTX 4090 yields higher sampling power and shorter training time than on the RTX 3080. This is because the RTX 4090 features a more advanced manufacturing process and more computation units, which contribute to more efficient and faster model training than RTX 3080.

2) Consumption with Different Datasets: Fig. 1 also compares the power consumption of identical models trained on the same hardware using different datasets. Each curve shows the energy used by a model during training. Specifically, data is provided for the AlexNet, ResNet18, GoogleNet, and VGG11 models trained on an RTX 3080 GPU. Among the datasets used in this study, CIFAR-100 has 50,000 training samples with 3 channels each, while Fashion-MNIST has 60,000 training samples with 1 channel each. When the number of MACs in a model is small, as in AlexNet, the energy required for training on CIFAR-100 is higher than that for Fashion-MNIST. However, as the MACs increase, training energy is affected more by the number of images than by the number of channels per image.

3) Consumption with Different Models: From Fig. 1, it can also be observed that among the models shown, AlexNet, ResNet18, GoogleNet, and VGG11 exhibit significant differences in energy consumption when training the Fashion-MNIST dataset on the RTX 3080 GPU.

Figs. 2 and 3 show the relationship between the number of MACs of each model and its total energy consumption during training on different devices. The results demonstrate a linear relationship between the number of MACs and energy consumption across various combinations of hardware and data sets. When different models are trained on the same GPU with the same dataset, the MACs of the models and their needed training energy use also display a linear relationship.

This enables energy demand to be inferred from computational complexity (e.g., MACs), since algorithmic choices (e.g., model depth, input size) directly affect their energy footprint.

4) Energy Consumption with Different Steps in Training Process: Fig. 4 illustrates the energy consumption of different models at various steps at each iteration of the training, when using the Fashion-MNIST dataset on the RTX 3080 GPU,

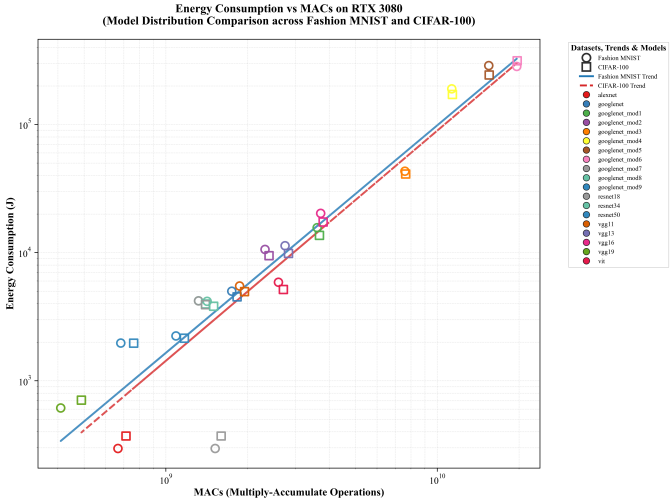


Fig. 2. Relationship between Energy Consumption and Model MACs of Different Models

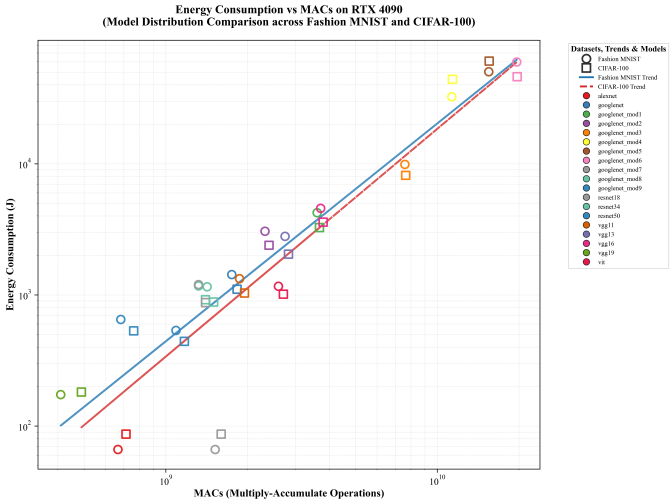


Fig. 3. Relationship between Energy Consumption and Model MACs of Different Models

using the two different datasets presented above. Each iteration of the training is divided into five steps:

- The *To Device* step loads the training dataset into the GPU in batches.
- The *Forward* step passes each batch of training data through the model to obtain the predicted output results.
- The *Loss* step calculates the difference between the true values and the predicted values using the loss function.
- The *Backward* computes the partial derivatives of the loss concerning the model parameters and accumulates the gradients.
- The *Optimizer* step updates the model parameters using the gradient descent algorithm based on the computed gradients.

While the energy consumption for loss calculation and optimization steps is negligible, the backward step emerges as the

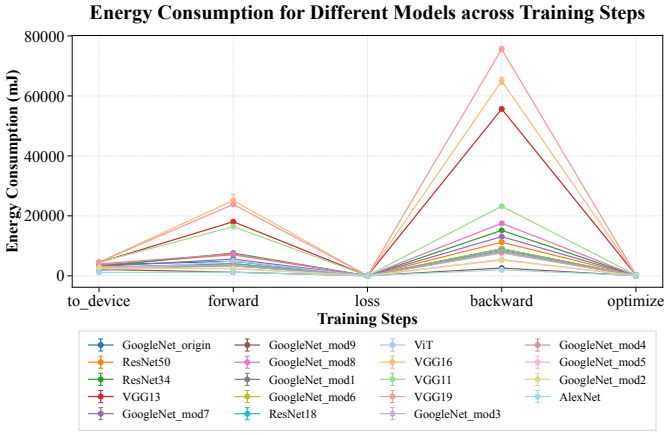


Fig. 4. Consumption of Different Steps of Training Process

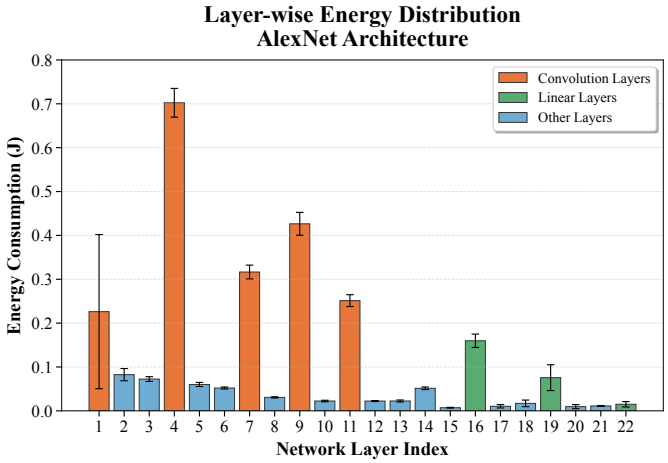


Fig. 5. Energy Consumption in each Layer in AlexNet

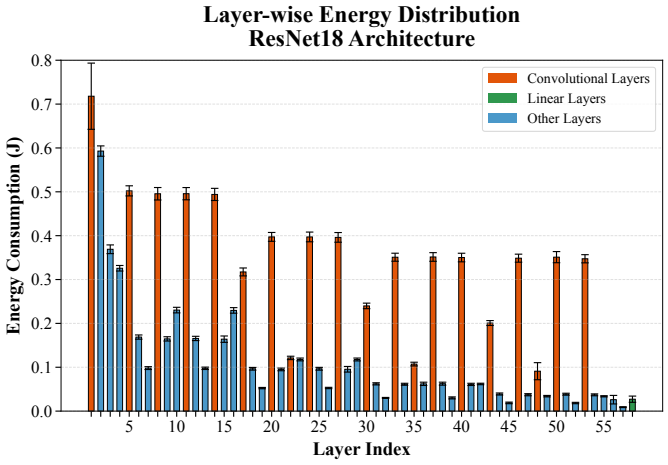


Fig. 6. Energy Consumption in each Layer in ResNet18

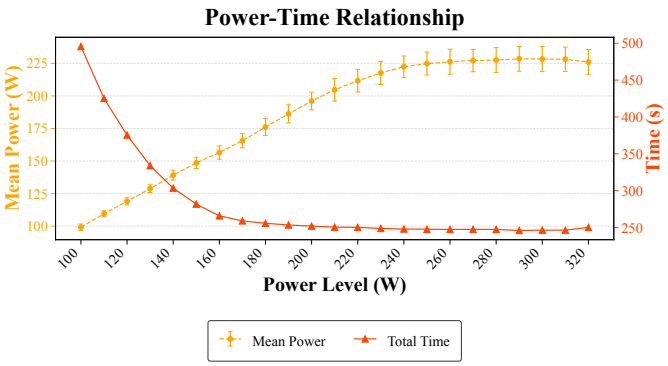


Fig. 7. Relationship between Power Level and Performance of GPU

most energy-consuming step, requiring up to more than double the energy of the forward one during training, depending on the model and dataset used. This highlights that backward computation, primarily responsible for gradient calculation, is the most energy-intensive stage of the training pipeline.

Fig. 5 and Fig. 6 illustrate the energy consumption of each layer during the forward propagation process for the AlexNet and ResNet18 models. It can be observed that during forward propagation, convolutional layers exhibit the highest energy consumption. This is primarily because convolutional layers require the largest number of MACs, which makes them the most energy-intensive components in the training process.

V. Relationship between Power Limits and Performance of GPU

In this section, we analyze the data collected during model training under different GPU power limits. Specifically, we consider 23 power levels ranging from 100 W to 320 W, using an RTX 4070 GPU. To enforce these limits, we use Nvidia's NVML module, which allows us to cap the GPU's maximum power level during training. For the experiment, we train the ResNet18 model on the Fashion-MNIST dataset, conducting 23 training rounds, one for each power level. In each round, the GPU's power limit is set to a different level, and the model is trained for 5 epochs with a batch size of 128. To ensure consistency, the model parameters are identically initialized before each round.

Fig. 7 illustrates the mean power and the time required to finish the training process under different power level. The yellow and red curves represent the average power during the training process, in watts, and the total training time, in seconds. We observe from the figure that as the GPU's power limit increases, the time required to complete the training task decreases (orange curve in the figure), while the mean power requirement for training the model increases (yellow curve in the figure). However, the rate of reduction for the time slows down as the power limit continues to rise, and no further changes occur once the power limit reaches 230 W. Regarding overall mean power needs (yellow curve), it initially decreases as the GPU power limit increases, reaching a minimum at 170 W. Beyond this point, the average power during training begins to rise and stabilizes once the power limit exceeds 230 W. This behavior occurs because, at power limits above 230 W, the GPU's capacity surpasses the actual power demanded during

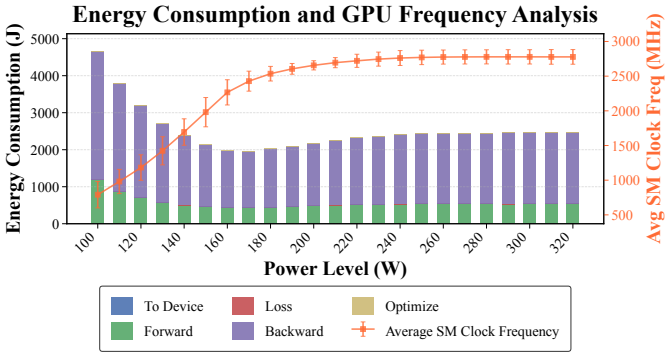


Fig. 8. Relationship between Power Level and Performance of GPU

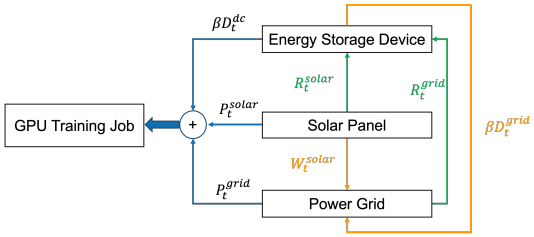


Fig. 9. Relationship between Power Level and Performance of GPU

training. In other words, increasing the power limit beyond 230 W has no further effect, as the training process does not utilize the additional available power.

The bars in different colors in Fig. 8 show the contribution of each step in each iteration during training on the left y-axis, corresponding to different GPU power levels. The orange curve on the right y-axis represents the average frequency of the Streaming Multiprocessor (SM) during training under different power limits. It can be observed that the total energy consumption during the training process initially decreases as the GPU power limit increases to 170 W. After that, it subsequently rises as the GPU power limit goes up to 320 W. This occurs because the SM core runs at a lower frequency under lower power limits, resulting in a lower efficiency ratio between computational output and power consumption. As the power limit increases to 170 W, the SM cores' operating frequency increases approximately linearly with power, and the energy consumption of the GPU during training continuously decreases. This suggests that the efficiency of the SM improves as the power limit is initially raised. However, as the GPU power limit continues to rise until reaching the maximum, the rate of SM frequency increase gradually slows and plateaus after surpassing 230 W. This is because, at this point, the GPU's capacity can already meet the model's power demands under the set parameters, and it continues to increase with the power limit but stabilizes after 230 W. As subsequent energy consumption rises, the proportion of increase in computational output becomes smaller than the proportion of energy consumption increase, leading to a decline in the efficiency ratio. Additionally, setting the GPU to its maximum power during training does not yield any additional benefits. To minimize energy consumption, the GPU power limit can be set to 170 W, resulting in a training time of 259 seconds. If the goal is to complete training as quickly as possible, the minimum training time achieved is 245 seconds, only 5% faster than at 170 W. However, this speed gain comes at a significant energy cost: power consumption increases from 1 964 J at 170 W to 2 452 J, which is about 25% higher than the optimal value.

The simulations show that adjusting GPU power limits within hardware capacity produces distinct performance–energy profiles, enabling training to align with renewable availability. These insights form the basis for our subsequent analysis, in which we consider edge computing nodes powered by renewables and storage, making energy management a central component of sustainable AI deployment.

VI. RES Energy Supply

To investigate the potential of power supply based on Renewable Energy Sources (RES) for reducing the energy cost for the training on edge computing nodes, in this part of the work, we consider a GPU-equipped with a hybrid power system, composed of PV panels, energy batteries, and the power grid, as depicted in Fig. 9.

We formalize an energy management for the training phase, which starts at T_s and must finish within the T_e, which represents the deadline by which the training must be completed. At each timestamp t, the energy management sets the GPU power level and allocates the available energy to minimize the cost. In particular, it chooses one of the 23 possible GPU power level, ranging between 100 W and 320 W, as described in the previous section. Each GPU power level corresponds to a completion time for the training task (see Fig. 7). Increasing the GPU power limit means speed up the time required for the training. In addition, the energy management determines whatever to use totally or partially the energy produced by the PV panel, for the training supply, or totally or partially inject it in the energy battery, if possible, or sell it back to the power grid. In addition, the system can buy some energy from the power grid, to directly supply the training, or to store it in the battery, if enough capacity is available, for later usage. Alternatively, some energy can be taken from the battery. We formalize the optimization problem, whose input and variables are summarized in Table II, and whose optimization function is as follows:

TABLE II
PARAMETERS AND THEIR TYPES

Parameter	Description	Type
T_s (h)	Start time of total jobs	Input
T_e (h)	End time of total jobs	Input
ESD_{max} (W)	Maximum ESD capacity	Input
J_t (h)	Model training time per slot	Input
$Q_t^{\text{grid}'}$ (€/kWh)	Unit price for sold energy to grid	Input
G_t^{solar} (W)	Power generated by solar panels	Input
Q_t^{solar} (€/kWh)	Unit price of solar generation	Input
G_t^{grid} (W)	Power available from grid	Input
Q_t^{grid} (€/kWh)	Unit price of grid purchase	Input
P_t (W)	Supplied power to GPU	Decision
W_t (W)	Power sold back to grid	Decision
C_t (W)	Net power consumption in time-slot	Decision
P_t^{solar} (W)	Solar power allocated to GPU	Decision
R_t^{solar} (W)	Solar power charged into ESD	Decision
W_t^{solar} (W)	Solar power sold back to grid	Decision
P_t^{grid} (W)	Grid power allocated to GPU	Decision
R_t^{grid} (W)	Grid power charged into ESD	Decision
R_t (W)	Total power charged into ESD	Decision
W_t^{grid} (W)	Grid power sold from ESD to grid	Decision
D_t^{grid} (W)	ESD discharge power to grid	Decision
D_t^{dc} (W)	ESD discharge power to GPU	Decision
ESD_t (W)	ESD energy level at each time slot	Decision

TABLE III
ESD EFFICIENCY PARAMETERS

Charge α	Discharge β	Self-discharge θ
0.9	0.9	0.005

$$\min \sum_{t=T_s}^{T_e} C_t \tag{1}$$

where C_t represents the total cost at time t, calculated as the cost of purchasing energy from the power grid minus the revenue earned from selling energy back to it:

$$C_t = G_t^{\text{grid}} \cdot Q_t^{\text{grid}} - W_t^{\text{grid}} \cdot Q_t^{\text{grid}'} \tag{2}$$

where G_t^{grid} (in kWh) and Q_t^{grid} (in €/kWh) represent the amount of energy purchased from the grid and its unit cost, respectively; W_t^{grid} (in kWh) and $Q_t^{\text{grid}'}$ (in €/kWh) denote the amount of energy sold to the grid and the corresponding unit revenue at time t.

We use the decision variable $y_{t,k} \in \{0, 1\}$, set to 1, if, at time t the power level of GPU is set to level k, given $k \in \{1, 2, ..., 23\}$. The corresponding GPU power requirement is $P_k \in \{100, 110, ..., 320\}$. Since at each time slot, a single level can be selected, we have the following constraint:

$$\sum_{k=1}^{23} y_{t,k} = 1 \quad \forall t \in [T_s, T_e] \tag{3}$$

Consequently, the power requirement for time t is

$$P_t = \sum_{k=1}^{23} y_{t,k} \cdot P_k \quad \forall t \in [T_s, T_e] \tag{4}$$

In addition, we impose that the energy generated by the PV panel, G_t^{solar}, can not be wasted: it can be used for training supply, or stored in the battery, or sold back to the grid:

$$G_t^{\text{solar}} = P_t^{\text{solar}} + R_t^{\text{solar}} + W_t^{\text{solar}} \quad \forall t \in [T_s, T_e] \tag{5}$$

where P_t^{solar}, R_t^{solar}, and W_t^{solar} are the solar energy used for the training supply, injected in the battery, and sold to the grid at time t, respectively.

Then, at each time stamp t, the energy purchased from the power grid, G_t^{grid}, is used for the supply of the training or stored in the battery:

$$G_t^{\text{grid}} = P_t^{\text{grid}} + R_t^{\text{grid}} \quad \forall t \in [T_s, T_e] \tag{6}$$

where P_t^{grid} and R_t^{grid} are the energy used for the supply of the training and stored in the battery at time t.

The energy injected into the battery at time t, is given by the energy produced by the PV panel or bought from the grid, but not used for the training supply. This happens, for example, when the PV panel produces more energy than needed or when the energy price is low, making it convenient to buy energy for later use. This is formalized as follows:

$$R_t = R_t^{\text{solar}} + R_t^{\text{grid}} \quad \forall t \in [T_s, T_e] \tag{7}$$

where R_t^{solar} is the energy produced by the PV panel and injected into the battery, and R_t^{grid} is the energy purchased from the grid and stored in the battery, at time t.

At each time stamp t, part of the energy generated by the PV panel, W_t^{solar}, or stored in the battery, βD_t^{grid}, given β the discharging battery efficiency, can be sold to the power grid:

$$W_t^{\text{grid}} = W_t^{\text{solar}} + \beta D_t^{\text{grid}} \quad \forall t \in [T_s, T_e] \tag{8}$$

In addition, given D_{t-1}^{dc} and D_{t-1}^{grid} the energy in the battery that is used for the training supply at time t and sold to the grid, respectively, the battery status, ESD_t, is updated as follows:

$$\text{ESD}_t = (1-\theta)\cdot[\text{ESD}_{t-1} - D_{t-1}^{\text{dc}} - D_{t-1}^{\text{grid}} + \alpha R_{t-1}] \quad \forall t \in [T_s, T_e] \tag{9}$$

$$0 \leq D_t^{\text{dc}} + D_t^{\text{grid}} \leq \text{ESD}_t \quad \forall t \in [T_s, T_e] \tag{10}$$

where θ and α are the self-discharging and the charging efficiency, respectively.

The battery status is constrained by the maximum battery capacity, ESD_{max}:

$$0 \leq \text{ESD}_t \leq \text{ESD}_{\text{max}} \quad \forall t \in [T_s, T_e] \tag{11}$$

At the beginning, it is set to zero:

$$\text{ESD}_{T_s} = 0, \quad D_{T_s}^{\text{dc}} = 0, \quad D_{T_s}^{\text{grid}} = 0 \tag{12}$$

The maximum amount of energy that can be injected into the battery at time t, R_t, is equal to the residual available capacity:

$$0 \leq R_t \leq \text{ESD}_{\text{max}} - \text{ESD}_t \quad \forall t \in [T_s, T_e] \tag{13}$$

TABLE IV
GPU POWER LIMIT AND COMPLETION TIME

Max.GPU power P(W)	Appr.training time J(h)
100	27.52
110	23.62
120	20.85
130	18.54
140	16.86
150	15.65
160	14.77
170	14.39
180	14.21
190	14.08
200	13.99
210	13.91
220	13.90
230	13.81
240	13.77
250	13.76
260	23.75
270	13.75
280	13.66
290	13.69
300	13.69
310	13.69
320	13.69

The total energy used for the training supply, P_t, is the sum of all power sources supplying the GPU, i.e., the solar energy, the energy taken from the energy battery, and from the grid:

$$P_t = P_t^{\text{solar}} + P_t^{\text{grid}} + \beta D_t^{\text{dc}} \quad \forall t \in [T_s, T_e] \tag{14}$$

Power consumption in each time slot should be less than or equal to the selected GPU power:

$$C_t \leq P_t \quad \forall t \in [T_s, T_e] \tag{15}$$

Finally, the progress of the model training must satisfy the following equation:

$$\sum_{t=1}^{T} \frac{1}{J_t} \geq 1 \quad \forall t \in [T_s, T_e] \tag{16}$$

A. GPU power requirement and execution epochs

The GPU operating data is based on the performance analysis of the GPU under different power limits in the previous section. The training model is set as ResNet18, the training dataset is Fashion-MNIST, and the GPU used is RTX 4070. The total number of epochs for the entire training task is set to 1000, and the relationship between the set power limit and the corresponding completion time is shown in Table IV.

B. Solar Panel Generation

In this work, we use data on the energy production of a solar panel system, obtained from the PVWatts website [25], selecting Turin as the location. We consider 10 different sizes for the solar panels' DC system, ranging from 100 W to 1000 W. PVWatts provides hourly energy production data for the solar panel throughout the year.

TABLE V
TARIFF PRICE

Tariff	Time slot	Q_t^{grid} €/kWh	$Q_t^{\text{grid}'}$ €/kWh
F1	08:00 – 20:00	0.45	0.3
F2	20:00 – 23:00	0.35	0.3
F3	23:00 – 08:00	0.25	0.3

C. Storage Battery

The battery model is as in [26] [27], and we set the charge/discharge efficiency, self-discharge rate, and overall capacity of the batteries, as reported in Table III. The battery rated capacity is set to 250 Wh.

D. Grid Data

In our work, we adopt the *time of use* pricing model for residential electricity, following the scheme proposed in [28]. Electricity prices vary between different periods of the day, as summarized in Table V. The selling price of electricity reflects the marginal cost of avoided energy purchases and the associated value of carbon emission reductions. We set this selling price to a fixed value of 0.3 €/kWh.

E. Simulation Scenarios

We considered two different simulation scenarios:

1) *On Grid*: In this case, the available energy is managed by solving the optimization problem formalized above, using Gurobi's optimizer (see [29]), to provide the needed energy for the GPU training task.
2) *Off the Grid*: Similar to the previous scenario, the available energy is managed by solving the same cost-minimization optimization problem. However, in this case, the GPU training task relies solely on solar panels for power. If the available energy is insufficient to support training, the task is temporarily interrupted.

F. Simulation Results

1) Off the Grid: In this scenario, we assume that the GPU training task is powered solely by solar energy, without relying on the power grid or ESD, and that there are no deadlines for completing the training. Since no power grid is available and, therefore, no energy can be purchased from it, the cost is zero. Because of this, we analyze the time required for training to reflect the impact of supplying the entire process with renewable energy sources. Two operating modes are configured:

- The GPU operates at its minimum power level.
- The GPU operates at the maximum power level within the range allowed by the solar panel's output energy.

In the first mode, the GPU training task runs entirely at the minimum power level of 100 W throughout the training process. Fig.10 shows the time required to complete the training task when using solar panels with different output power levels across various months. The figure illustrates that training time varies depending on the month when the

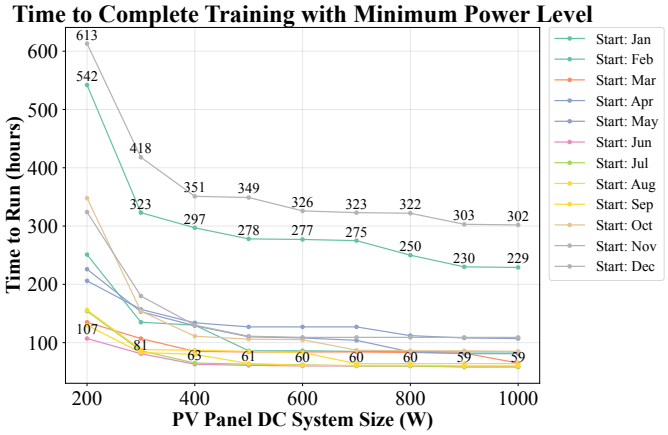


Fig. 10. Time need for Finishing Training Job under Different PV Panel Size and Different Month with Minimum Power Level

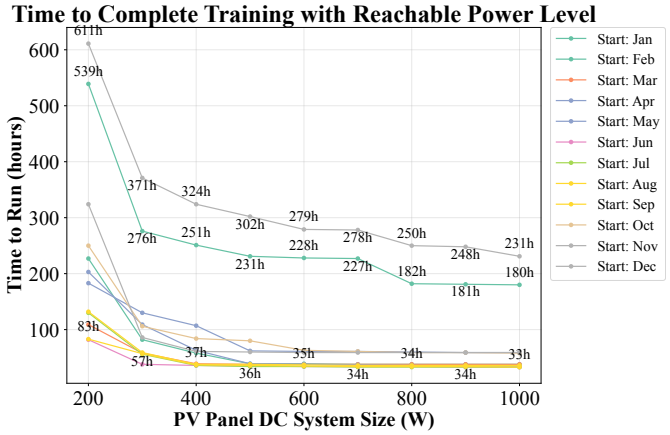


Fig. 11. Average GPU Power Level under Different PV Panel Size in Different Month

process begins, emphasizing the significant impact of seasonal irradiance and weather variability on solar-powered training. Specifically, when training starts in July, the shortest training duration is achieved with any PV system size. Additionally, increasing the panel's capacity reduces training time. When the GPU power is dynamically adjusted, the adjustment logic ensures that the GPU operates at the maximum power level achievable without exceeding the energy output provided by the solar panels. The results of the time required to complete the training with different PV sizes, starting in different months, are shown in Fig. 11. As for the previous case, we notice that as the size of the PV increases, the time needed to complete the training task varies by month. Specifically, s. Specifically, when training begins in July, the shortest training duration is achieved with any PV system size. Each curve in Fig. 12 shows the mean GPU power level during training in a given month, with the PV panel capacity increasing on the x-axis. It rises almost linearly until reaching 700 W, where the GPU power level reaches 230 W. Beyond this, the curve turns slightly downward because the GPU's average power reaches its maximum operating power.

By comparing Figs.10 and 11, it can be seen that significant differences are observed when different GPU power levels are enabled, compared to when they are not enabled, in cases where the PV panel capacity is no larger than 200 W. The situation changes with larger PV panel capacities: enabling the use of different GPU power levels significantly reduces the time needed for training, as a large PV panel allows for the activation of high GPU power levels, thereby speeding up the training process.

2) On Grid: In this part of the work, we consider the *On Grid* scenario, where the PV panel, the ESD, and the power grid power the training. Figs. 13, 14, and 15 show the total training cost, varying the maximum training duration (between 14 and 28 hours), the training start time (between 00:00 and 20:00), assuming that it begins on January 1^{st}, and the PV panel capacity (from 100 Wp to 1000 Wp), when the scenario *On Grid* is considered.

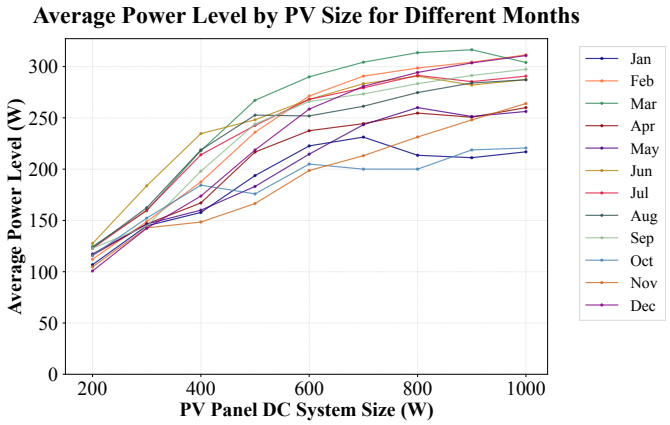


Fig. 12. Average Power Level by PV Size for Different Month

It can be observed that the cost of GPU training tasks initially increases significantly as the training duration limit is extended. However, the cost gradually decreases with further increases in the time limit, reaching its lowest value of 0.72 € when the training duration limit is 19 hours. Beyond this point, the cost fluctuates between 0.7 € and 0.9 € as the training duration continues to increase. This is because, when the GPU training duration is extended, the model can utilize the excess energy generated by the solar panels beyond what is required for GPU training by storing it in an ESD or selling it back to the grid.

The start time of training tasks also significantly impacts the overall cost. When training begins at 8 a.m., the cost hits its highest point, whereas starting at 8 p.m. results in the lowest cost. This difference is due to the pricing of electricity grids. Specifically, starting training at 8 a.m. coincides with higher electricity rates during that time. Additionally, the GPU can only access one instance of cheap grid electricity to charge the ESD, which limits potential savings. Furthermore, as shown in Fig. 16, the energy generated by solar panels during the daytime on the first day isn't sufficient to fully power the GPU training task. As a result, extra energy must be drawn from the

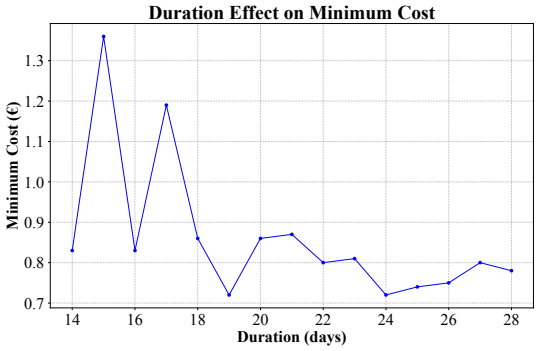


Fig. 13. Effect of Duration

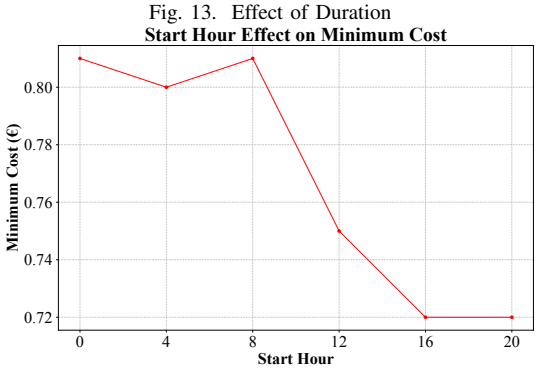


Fig. 14. Effect of Start Hour

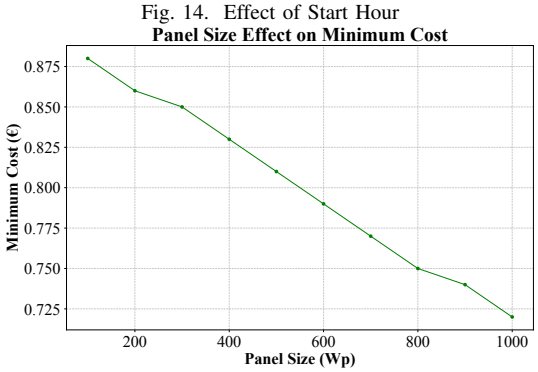


Fig. 15. Effect of Panel Size

grid, increasing the total cost. Conversely, beginning training at 8 p.m. allows it to continue into the next day when solar panels produce more energy. This enables the GPU to rely more on solar energy rather than grid electricity, substantially lowering training costs. This highlights the importance of scheduling training during periods of peak solar energy to maximize energy efficiency and minimize costs.

We report the results in Table VI for the case where the solar panel size is 1 000 W, the training starts at 8 p.m., and the training duration limit is 19 hours. The table reports the cost and the amount of energy purchased from the grid in cases where we use our optimization problem to manage the available energy for the training supply (*our solution* in the table), as well as when the training is supplied solely by the power grid (*current solution* in the table). In the former case, the training task achieves the cost of 0.72 €, with a power

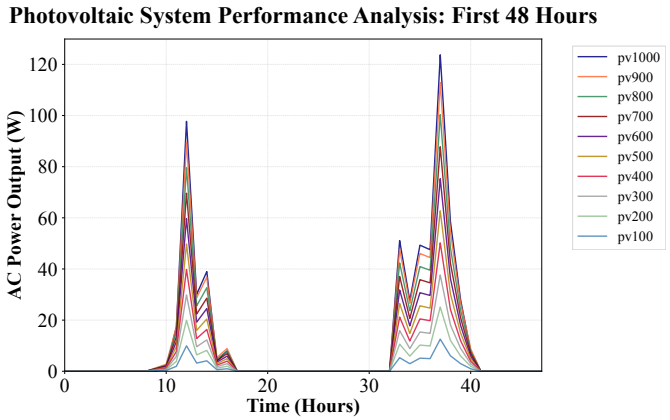


Fig. 16. Solar Panel Output Over First 48 Hours in Different Size

TABLE VI
DIFFERENT ENERGY USAGE

Method	Grid Usage (J)	Cost (€)
Current solution	3220	1.33
Our proposal	2304.37	0.72

of 2 431.71 J taken from the grid. In the latter, the cost of completing the training task rises to 1.33 €, accounting for 3 220 J bought from the power grid.

This demonstrates that effective energy management reduces training costs by 46.03% and grid energy consumption by 24.48%, highlighting its efficiency in optimizing energy usage and minimizing operational costs.

VII. LESSON LEARNT

In this section, we discuss the main aspects that have emerged in our work. This paper investigates the energy requirements of image classification AI models by assessing their performance and power usage across various neural network architectures, datasets, and GPU configurations, to analyze the most impactful aspects. In particular, we collect energy consumption data during the training process of 20 image classification DNN models trained on two different datasets, using two different hardware devices. The work also evaluates the impact of the GPU efficiency under different power limits on the total energy consumption and needed training time. To tackle the high operational costs associated with increasing AI adoption and its energy demands, the work suggests using renewable energy sources as a promising solution. However, because renewable energy is intermittent, the available energy must be managed effectively. To accomplish this, the paper formulates an optimization problem that leverages real weather data and electricity prices to lower training costs, incorporating multiple energy sources such as solar power, battery storage, and the power grid.

First, we observe that the most energy-intensive operations during training are the forward pass and backpropagation, with the latter consuming more than twice the energy of the former.

This highlights the importance of optimizing backpropagation when aiming to reduce training energy demands.

We also find a strong linear relationship between a model's energy consumption and its number of MACs, suggesting that computational complexity is a reliable predictor of energy use.

Focusing on hardware behavior, we examine the training of ResNet18 on an RTX 4070 GPU. Interestingly, energy consumption initially decreases as the GPU's maximum power setting is increased but begins to rise again once this threshold exceeds the model's actual computational demand. Beyond a certain point, further increasing the power limit has no effect, indicating a saturation in energy efficiency.

Finally, we investigate the role of renewable energy. Since solar panel output varies significantly with seasonal conditions, we introduce an optimal energy management approach to intelligently coordinate energy sources. This method enables a 28.44% reduction in energy drawn from the power grid and a substantial 47.53% decrease in overall training costs.

VIII. Conclusion

This paper examines the energy requirements of image classification AI models across various architectures, datasets, and GPU configurations, while also assessing GPU efficiency under different power constraints. To combat the rising energy costs of AI, it considers using renewable sources despite their intermittent nature. An optimization problem is formulated to minimize training expenses by utilizing real weather data and electricity prices, while coordinating energy from solar panels, battery storage, and the power grid. The proposed energy management reduces training costs by 46.03% and grid energy consumption by 24.48%.

Future work will address two directions. First, measuring the energy footprint of large-scale AI training (e.g., LLMs on GPUs such as NVIDIA H100) to benchmark against edge tasks. Second, refine the Energy Dispatch Model by multi-regional renewable integration to reduce grid dependence.

Acknowledgment

This work has been supported by the project "National Center for HPC, Big Data and Quantum Computing, CN00000013 (Bando M42C – Investimento 1.4 – Avviso Centri Nazionali" – D.D. n. 3138 of 16.12.2021, funded with MUR Decree n. 1031 of 17.06.2022).

References

[1] A. Khoshsirat, G. Perin, and M. Rossi, "Decentralized llm inference over edge networks with energy harvesting," *arXiv preprint arXiv:2408.15907*, 2024.

[2] L. C. Vieira, M. Longo, and M. Mura, "From carbon dependence to renewables: The european oil majors' strategies to face climate change," *Business Strategy and the Environment*, vol. 32, no. 4, pp. 1248–1259, 2023.

[3] G. Jansen, Z. Dehouche, and H. Corrigan, "Cost-effective sizing of a hybrid regenerative hydrogen fuel cell energy storage system for remote & off-grid telecom towers," *International Journal of Hydrogen Energy*, vol. 46, no. 35, pp. 18 153–18 166, 2021.

[4] F. Odoi-Yorke and A. Woenagnon, "Techno-economic assessment of solar pv/fuel cell hybrid power system for telecom base stations in ghana," *Cogent Engineering*, vol. 8, no. 1, p. 1911285, 2021.

[5] G. Vallero, M. Deruyck, M. Meo, and W. Joseph, "Base station switching and edge caching optimisation in high energy-efficiency wireless access network," *Computer Networks*, vol. 192, p. 108100, 2021.

[6] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[7] A. Berthelot, E. Caron, M. Jay, and L. Lefèvre, "Estimating the environmental impact of generative-ai services using an lca-based methodology," *Procedia CIRP*, vol. 122, pp. 707–712, 2024.

[8] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: quantifying the carbon footprint of computation," *Advanced science*, vol. 8, no. 12, p. 2100707, 2021.

[9] J. Dodge, T. Prewitt, R. Tachet des Combes, E. Odmark, R. Schwartz, E. Strubell, A. S. Luccioni, N. A. Smith, N. DeCario, and W. Buchanan, "Measuring the carbon intensity of ai in cloud instances," in *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, 2022, pp. 1877–1894.

[10] NVIDIA, "Inference: The next step in gpu-accelerated deep learning," https://developer.nvidia.com/blog/inference-next-step-gpu-accelerated-deep-learning/, 2016, [Accessed: 03-Apr-2024].

[11] Y. Wang, Q. Wang, S. Shi, X. He, Z. Tang, K. Zhao, and X. Chu, "Benchmarking the performance and energy efficiency of ai accelerators for ai training," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020, pp. 744–751.

[12] S. Hong and H. Kim, "An integrated gpu power and performance model," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 280–289.

[13] Z. Tang, Y. Wang, Q. Wang, and X. Chu, "The impact of gpu dvfs on the energy and performance of deep learning: An empirical study," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, 2019, pp. 315–325.

[14] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "Neuralpower: Predict and deploy energy-efficient convolutional neural networks," in *Asian Conference on Machine Learning*. PMLR, 2017, pp. 622–637.

[15] J. You, J.-W. Chung, and M. Chowdhury, "Zeus: Understanding and optimizing gpu energy consumption of dnn training," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 119–139.

[16] AutoDL, "Autodl platform," https://www.autodl.com/, [Accessed: 03-Apr-2024].

[17] NVIDIA, "Nvidia management library (nvml)," https://developer.nvidia.com/management-library-nvml, [Accessed: 03-Apr-2024].

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[22] Zalando Research, "Fashion-mnist dataset," https://huggingface.co/datasets/zalando-datasets/fashion_mnist, 2025, accessed: 2025-04-03.

[23] University of Toronto Computer Science, "Cifar-100 dataset," https://huggingface.co/datasets/uoft-cs/cifar100, 2025, accessed: 2025-04-03.

[24] V. Sovrasov, "ptflops: a flops counting tool for neural networks in pytorch framework," https://github.com/sovrasov/flops-counter.pytorch, 2018, accessed: 2024-04-03.

[25] "Pvwatts calculator," https://pvwatts.nrel.gov/pvwatts.php, 2025, accessed: 2025-04-03.

[26] "Pvwatts calculator," https://pvwatts.nrel.gov/pvwatts.php, 2025, accessed: 2025-04-03.

[27] "Today in energy: Battery storage increasing rapidly," https://www.eia.gov/todayinenergy/detail.php?id=46756, 2025, accessed: 2025-04-03.

[28] "Tou price," https://www.arera.it/consumatori/offerte-standard-per-i-clienti-finali-placet, 2025, accessed: 2025-04-03.

[29] "Gurobi optimization," https://www.gurobi.com/, 2025, accessed: 2025-04-03.