How to estimate carbon footprint when training deep learning models? A guide and review

Lucia Bouza Heguerte, Aurélie Bugeau, Loïc Lannelongue

How to estimate carbon footprint when training deep learning models? A guide and review

Lucía Bouza Heguerte,
Université Paris Cité, CNRS, MAP5 UMR 8145, 75006, Paris, France

Aurélie Bugeau
Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, Talence, France
IUF
aurelie.bugeau@u-bordeaux.fr

Loïc Lannelongue
Cambridge Baker Systems Genomics Initiative, Department of Public Health and Primary Care
British Heart Foundation Cardiovascular Epidemiology Unit, Department of Public Health and Primary Care
Victor Phillip Dahdaleh Heart and Lung Research Institute
Health Data Research UK Cambridge, Wellcome Genome Campus
University of Cambridge, Cambridge, United Kingdom

Abstract

Machine learning and deep learning models have become essential in the recent fast development of artificial intelligence in many sectors of the society. It is now widely acknowledge that the development of these models has an environmental cost that has been analyzed in many studies. Several online and software tools have been developed to track energy consumption while training machine learning models. In this paper, we propose a comprehensive introduction and comparison of these tools for AI practitioners wishing to start estimating the environmental impact of their work. We review the specific vocabulary, the technical requirements for each tool. We compare the energy consumption estimated by each tool on two deep neural networks for image processing and on different types of servers. From these experiments, we provide some advice for better choosing the right tool and infrastructure.

1 Introduction

Deep learning has been widely used in every sector of the society for a few years. A search of Scopus shows that it went from about 1,350 research papers in 2015 to more than 85,000 in 2022. Results obtained in every domain are impressive, and AI is a promising tool for tackling environmental challenges in particular [1, 2, 3]. But it is also now widely documented that training and deploying deep learning projects has an impact on the environment [4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. These studies have assessed energy consumption and corresponding amount of greenhouse gas emissions (in CO2 equivalent, denoted as CO2eq) from computer calculations when training a deep learning program, and showed that recent large language models can be responsible for hundreds of tonnes of CO2eq [14], whereas, for context, a limit of 2 tCO2eq/person/year is what is needed to keep global warming under 1.5°C [15]. Some studies have also compared existing estimation tools [10, 9, 16].

Despite these many studies, when AI practitioners wish to start estimating their environmental impact, they may face several difficulties. Depending on their backgrounds, it might be difficult for them to get used to the hardware-related vocabulary, know how to use the estimation tools (locally or on servers), and determine which tool is best suited for their current use-case. This document aims to address these and ease the process of energy consumption measurement for AI practitioners. It can be used as a guide to measure the energy consumption and associated greenhouse gas emission

when training deep learning algorithms and although what will be explained can be applied to other types of algorithms and other infrastructures, we will focus on training deep-learning models in different types of infrastructures.

In this context, this document makes the following contributions:

- We review existing tools for measuring or estimating the energy consumption of computations, and explain the specific notions that are not always known by AI practitioners. It goes further than previous surveys [10, 9, 16] in providing details about what is measured by each tool and on which infrastructure they can be used, the measurement process, how usage factor is being used, default values, and the source of information that are used. These information are crucial to correctly interpreting the data obtained.
- We test and compare these different approaches using wattmeters to assess their accuracy. We also quantify the energy consumption of the estimation tools themselves.
- We run a range of experiments to analyze the influence of key hyperparameters such as batch size, data load, checkpoints and epochs. These lead to a set of recommendations on how and when to use these tools depending on the infrastructure available to train the models. For instance, we show that it seems possible to only measure part of training and extrapolate to avoid the small extra consumption from energy measurement. We also show that batch size can influence energy consumption. The recommendations complete previous works that intended to make machine learning researchers better understand their carbon impact and to take steps to mitigate it [17, 12].

The seven different tools that we study are: Green-Algorithms [18] (GA), CodeCarbon [19] (CC (P) for process, CC (M) for machine), Eco2AI [20] (E2 (P) for process, E2 (M) for machine), CarbonTracker [21] (CT), Experiment-Impact-Tracker [22] (EIT), MLCO2 [23] and Cumulator [24] (CMLTRs).

We use the following infrastructures, all located in France, for training models: Labri servers (institutional server), MAP5 servers (institutional server), Grid5000 distributed cluster and personal computers. Mention will also be made of the Google Colab environment.

In the Labri servers, personal computer and in Grid5000 there are wattmeters (WM), which can provide real information on the consumption of energy of the infrastructure in a given period.

We focus on two machine learning experiments, both for image processing. In the first one, a small neural network is trained for digit classification on the MNIST dataset [25]. This experiment is short, approximately 1 minute. In the second, a DNCNN network is trained for noisy image denoising. The training is carried out with the Imagenet validation dataset [26]. This experiment is longer, approximately 2 h.

Figures 1 and 2 summarize the energy consumption of the different tools in the five tested infrastructure. As we

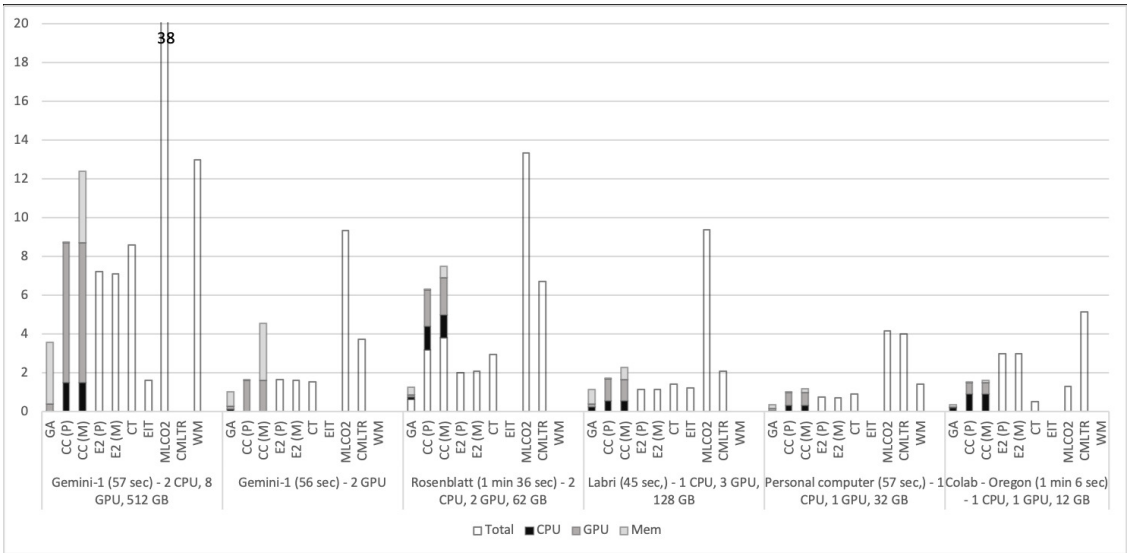

Figure 1: Energy consumption in Wh of the different methods over the 5 different infrastructures for the first experiment. For the tools that do not provide detail for CPU/GPU/Memory consumption, the total energy reported is plotted.

detail in this guide, the high variability comes from the different goals of the different tools, some estimate the power

consumption of the entire machine while others focus on a particular process. The idle power consumption is also accounted for differently, alongside usage factors, CPUs vs GPUs etc.

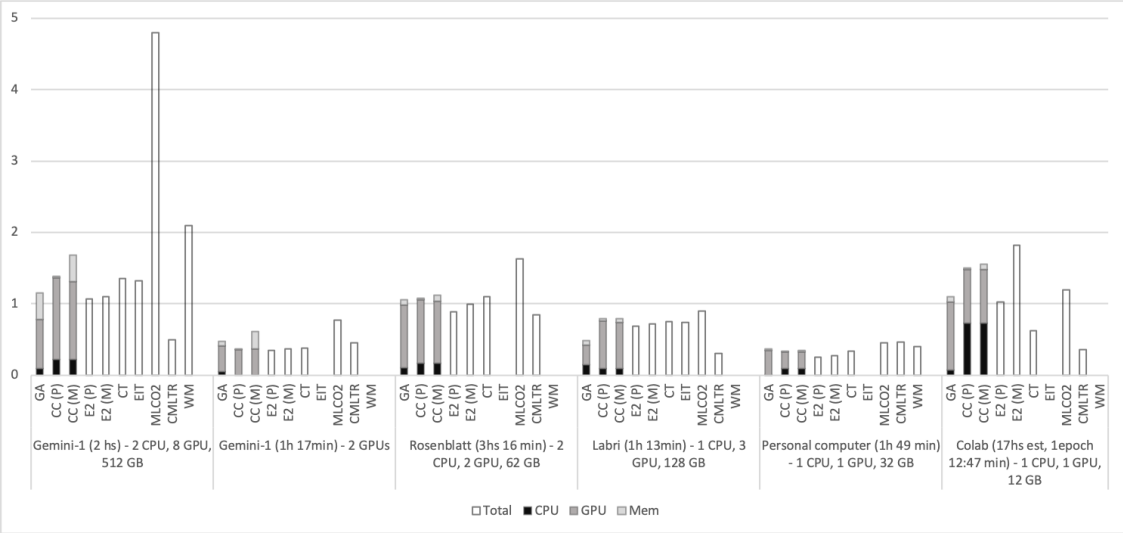

Figure 2: Energy consumption in kWh of the different methods over the 5 different infrastructures for the second experiment. For the tools that do not provide detail for CPU/GPU/Memory consumption, the total energy reported is plotted.

The document is organized as follows. Users already familiar with carbon footprint estimation may directly jump to section 5 for the results. Section 2 reviews previous publications in this field. Section 3 details the specificities of each tool: energy consumption of each hardware components and their communications, power usage effectiveness and emission intensity. Section 4 details the type of infrastructures that are typically used to train AI models and what tools can be used for each. Section 5 presents the experimental setup and an analysis of the results. Discussions on the results presented and recommendations on when and how to estimate all environmental impacts end this guide in section 6. Finally, errors reported and found in the tools are added in the Appendix.

2 Related works

Only recently estimation tools have been made available and consequently, few studies have compared and analyzed existing strategies for measuring energy consumption of deep learning projects.

The authors of [10] reviewed six tools (CarbonTracker, Experiment-impact tracker, Green-Algorithms, MLCO2, energy usage and Cumulator) that are available to measure energy use and CO2eq emissions in the context of natural language processing. They compared the tools according to publication details, technical criteria (availability, online, easy-of-use, documentation etc.), configuration criteria (specification of carbon intensity, PUE, install dependent, etc.) and functional criteria (idle power and communication between hardware). The authors observed a two-fold variation in estimates between tools and concluded that further studies are needed to better understand these tools and estimate broader impacts.

In the same line of research, the authors of [16] compared some tools on server nodes, not all specifically designed for deep learning and therefore not all integrating GPUs. They categorized tools between external and internal node sensors, power profiling software, energy measurement software packages and online energy calculators. They looked at publication year, environment criteria (hardware compatibility, virtualization, etc.), functional criteria (hardware compatibility, software power model, sampling frequency, reporting and profiling), and user-friendliness. They tested each tool on the same server nodes and compared them with external power meters. The authors drew some recommendations from this study: to monitor power consumption in real time, it is better to use power profiling software, but they do not measure GPUs consumption; relationship between energy measurement software tools and power meter is not constant, so software tools are not perfectly accurate.

Finally, [9] provided general guidelines about the strength and weaknesses of different types of estimation tools, namely online calculators, embedded packages and server-side tools. The criteria that are discussed are compatibility with any

hardware, any programming language, research field, some ease of use criteria and scalability with number of jobs and long periods of time.

The different tools discussed above focus on energy consumption during the training phase of AI models, which only constitute part of the broader environmental impacts of AI [5, 6, 7, 8, 9]. In this context, the authors of [14] later included embodied impacts as well as emissions from static infrastructure and deployment when studying BLOOM, a large language model.

3 Estimating greenhouse gas emissions

This section explains how tools measure or estimate energy consumption and CO2eq emissions, from Python libraries integrated into the code (referred to as software tools), to web forms and physical watt measurement devices connected to the infrastructure used. Some of these tools also have a server-side version, to be used in HPC clusters and thus be able to collect information more easily to estimate energy consumption. Online tools and server-side tools can be used without modifying the code, and are independent of the programming language used. Python libraries can only be used in Python code but enable measurements of the consumption of different parts of the programs. Watt-measurement enables measuring the consumption of the whole node but are not always available and can not isolate a paritcular process. Each tool has its own way of estimating the consumption of each component. A summary of the characteristics is shown in the table 7.

The most power consuming devices on a personal machine or a server are the GPUs (if present), CPUs, and memory. There are other resources, such as storage or the network, that are generally not considered in software measurements, since they do not provide a significant load over the duration of an AI task. Indeed, in regular use, storage is typically solicited far less than memory and is mainly used as a more permanent record of the data, independently of the task [18].

When the machine is in a data center, energy usage of all equipment that are necessary to power, cool and maintain the datacenter should be measured as they may account for an important amount of energy consumption. This is done using the efficiency coefficient of the data center called power usage effectiveness (PUE).

3.1 Energy consumption of each component

In this section, we will see the different strategies used by the tools to estimate the energy consumed by the different resources and estimate the consumption of the processes. Green-Algorithms and CodeCarbon are the only Python tools that report the estimate of consumed or emissions, discriminated by each component: memory, CPU and GPU.

A transversal concept to all resources is the usage factor. The usage factor of a resource refers to the percentage of use that can be assigned to the process being measured. For example, if the CPU power is estimated to 2W, but the CPU usage factor of the process was 50%, then the consumption of a one hour process is assumed to be 1 kWh. If the usage factor is unknown, then 100% of the use of the resource is being assigned to the process, when in fact there may be other processes also using said resource.

During the measured period, some tools query sensors or perform calculations to estimate power consumption. Note that Lower measurements frequency mean fewer measurements that may lead to more approximate results. By default, CodeCarbon performs these measurements every 15 seconds. Eco2AI, CarbonTracker and Experiment-Impact-Tracker take measurements every 10 seconds. Cumulator does not query sensors or intermediate measurements to estimate energy consumption.

3.1.1 Energy consumed by CPU

There are two methods used in the tools to estimate energy consumed by CPUs: using CPU thermal design power (TDP) provided by the manufacturer, or using software integrated tools (RAPL files or Power Gadget). Appendix A provides explanations of these two methodologies. Note hat software integrated tools may require privilege permissions as summarized in section 4.1. We review in table 1 how CPU power consumption is measured in AI measurement tools.

Table 1: Estimation of energy consumption for CPUs

Green-Algorithms	
Energy	uses the model of CPU provided by the user to pull the corresponding TDP from a database, or the user can input the TDP manually. If TDP is unknown, GA uses an average of 12W per core, but the paper does not explain this value. In this model, a core power usage is assumed to be equal to the TDP divided by the number of cores (if a chip has 2 cores and a TDP of 50W, then the TDP per core is 25).

Usage factor	uses usage factors if known, and assumes 100% usage if not.
CodeCarbon	
Energy	uses RAPL files or Power Gadget to report CPU energy consumption (only for INTEL CPUs with root access). The consumption reported by RAPL files or Power Gadget represents the consumption of the whole machine, and not only the process. If CodeCarbon cannot find the software to track the CPUs, then the tool uses the model of CPU to search in a list the corresponding TDP. If the model is unknown, it uses a TDP of 85W. The authors do no specify where is this value taken from.
Usage factor	Not computed when using RAPL files or Power Gadget When TDP is used, CodeCarbon assumes that the average usage factor is 50% but this value is not explained and seems arbitrary.
Eco2AI	
Energy	uses the model of the CPU to search in a list the corresponding TDP. If TDP is unknown, it uses an average of 100W [27].
Usage factor	uses *os* and *psutil* python modules to determine usage factor if the tracking mode *current* is set (default).
CarbonTracker	
Energy	uses RAPL files to report CPU energy consumption (only for INTEL CPUs with root access). Without access to the RAPL files, the tool will not measure CPU. CarbonTracker will work only if it can measure at least one component (CPU or NVIDIA GPU).
Usage factor	not computed. The power consumption values of the RAPL files are global to the whole machine.
Experiment-Impact-Tracker (EIT)	
Energy	uses RAPL files to report CPU energy consumption (only for INTEL CPUs with root access and Linux operating system)
Usage factor	uses *psutil* python module to determine usage factor
MLCO2	does not measure CPU utilization.
Cumulator	
Energy	It is not possible to measure GPU and CPU components at the same time but Cumulator measures CPU utilization by default. It uses the model of CPU to search in a list for the corresponding TDP. If TDP is unknown, it uses an average of 250W. This value is the one of Nvidia GeForce GTX Titan X, which is the GPU model in the IC cluster of the EPFL Machine Learning and Optimization Laboratory (MLO). It considers just one CPU.
Usage factor	does not use usage factor.

file:///Users/bugeau/Recherche/Impact/IA/ComparaisonOutils/Guide to measure carbon footprint when training deep learning models in France/bibliography.bib

3.1.2 Energy consumed by GPU

As with CPUs, energy consumption for GPUs are computed either from TDPs provided by manufacturers or from internal tools. The latter is done with the *pynvml* library that only works for Nvidia GPUs. We review in table 2 how GPU power consumption is measured in AI measurement tools.

Table 2: Estimation of energy consumption for GPUs

Green-Algorithms	
Energy	uses the model of GPU to search in a list the corresponding TDP. You can load the TDP of the GPU if the model is not listed. If TDP is unknown, it uses an average of 200W, but the paper does not explain the reason for choosing this value.
Usage factor	GPUs usage factor is considered if known by the user. If not, GA considers 100% of usage.
CodeCarbon	
Energy	uses *pynvml* library (only for NVIDIA GPUs). CodeCarbon does not measure consumption of non-NVIDIA GPUs.
Usage factor	not computed. The consumption reported by pynvml represents the consumption of the whole machine, and not only the process.
Eco2AI	
Energy	uses *pynvml* library (only for NVIDIA GPUs). Eco2AI does not measure consumption of non-NVIDIA GPUs.

Usage factor	not computed. The consumption reported by pynvml represents the consumption of the whole machine, and not only the process.
CarbonTracker	
Energy	uses *pynvml* library (only for NVIDIA GPUs). CarbonTracker does not measure consumption of non-NVIDIA GPUs.
Usage factor	not computed. The consumption reported by pynvml represents the consumption of the whole machine, and not only the process.
EIT	
Energy	uses *nvidia-smi* command line (only for NVIDIA GPUs). EIT does not measure consumption of non-NVIDIA GPUs.
Usage factor	uses `Popen` to open a thread, execute the command `nvidia-smi -q -x`, get the output in a xml, and parse it to get the usage factor of the GPU.
MLCO2	
Energy	uses the model of GPU to search in a list the corresponding TDP. It is not possible to load the TDP of the GPU if the model is not listed. In this case, it is necessary to do a pull request to add the value. It is not possible to choose the quantity of GPUs.
Usage factor	does not use usage factor. The GPU is considered at maximum load and this load is assumed to correspond to the measured process.
Cumulator	
Energy	uses the model of GPU to search in a list the corresponding TDP. If TDP is unknown, it uses an average of 250W. It considers just one GPU.
Usage factor	does not use usage factor. The GPU is considered at maximum load and this load is assumed to correspond to the measured process.

3.1.3 Energy consumed by memory

According to [28] GPUs are responsible for around 70% of power consumption, CPU for 15%, and RAM for 10%.

Some tools like Green-Algorithms consider that power consumption of RAM depends strongly on the available memory, independently of the memory consumed [29, 30], while other tools like Eco2AI considers that it depends on the allocated memory by the process [27]. We review in table 3 how memory power consumption is measured in AI measurement tools.

Table 3: Estimation of energy consumption for memory

Green-Algorithms	Energy consumption by memory is 0.3725W/GB of memory available (If we have all the server memory available, it will account for all the server memory. If we are in an HPC cluster, it will account only for the amount of memory requested, regardless of how much the process consumes). The value 0.3725 was obtained experimentally [1].
CodeCarbon	Energy consumption by memory is 0.375W/GB of memory used[2]. If tracking mode is "process", the memory used by the process is measured via *psutil*.
Eco2AI	Energy consumption of memory is 0.375W/GB of memory used [27]. Memory used by the process is measured via *psutil*.
CarbonTracker	uses RAPL files to report memory energy consumption. It measures the total energy of memory available, not only the one used by the process. Without access to the RAPL files, the tool will not measure memory energy consumption.
EIT	uses RAPL files or Power Gadget to report memory energy consumption. Memory used by the process is measured via *psutil* considering memory used exclusively by the process and the shared memory between processes (weighted by the number of processes). Without access to the RAPL files or Power Gadget, the tool cannot be used.
MLCO2	does not measure memory.
Cumulator	does not measure memory.

3.1.4 Energy consumed by communications

[1] Source: www.tomshardware.com
[2] Source: Crucial

In ICT (Information and Communication Technology), communications refer to the exchange of information or data between two or more nodes. Nodes can be any device that is connected to a network, including computers, routers, servers, and even mobile devices. Machine Learning algorithms typically involve the exchange of data between nodes at various stages, such as during data generation, during training (parameter updates across different nodes in the network), or while the model is in production.

The only tool that estimates the cost of communications is Cumulator. Each time the model sends a data file to another node of the network, Cumulator records the size of the file which is communicated. The cost of communication relies on the "1byte model" of the Shift Project [31]. The value from 2017 is 6.894×10^{-11} kWh/B.

3.2 PUE

Power Usage Efficiency is the efficiency coefficient of the data center. If PUE is not given, we recommend considering the 2022 average value of 1.55 [32]. For personal computers, PUE=1 as there are no other large devices consuming power. We review in table 4 the PUEs used by each tools. All except Cumulator report the total energy consumed, including PUE. To calculate this value for Cumulator, we can divide the reported value of greenhouse gas emissions (GHG) by the emission intensity (EI) of servers location: $Energy = GHG/EI$. Note that for the purpose of comparing reported energy consumption between tools, PUE is not taken into account, since each tool uses a different value.

Table 4: PUE values used in the different tools

Green-Algorithms	configurable. The default value is 1.67 (2019) [33].
CodeCarbon	not taken into consideration, except for cloud providers.
Eco2AI	configurable. The default value is 1.
CarbonTracker	configurable. Although the paper indicates that the 2020 PUE (1.58) is used, the 2022 PUE (1.55) is used in the code [32].
EIT	configurable. The default value is 1.58 (2020) [34].
MLCO2	not taken into consideration.
Cumulator	not taken into consideration.

3.3 Carbon emission and emission intensity

The origin of the energy used is key when determining greenhouse gas emissions from electricity production. To carry out the calculation, the average emission intensity (or carbon intensity) of the country or region where the calculations were made is used. Countries report these values, which can then be used by the tools to calculate emissions.

It is important to mention that most of the tools do not yet take the information of carbon intensity in real time. Only CarbonTracker (for UK and Denmark) and Experiment-Impact-Tracker (for California) do it. In most cases, average values from previous years are used. Some variables, such as the time of day of execution, or the distribution of energy sources at a given moment, are not represented, but can have an important influence on the emissions, as shown on table 5. Machine learning users could look at current and planned energy consumption of most of the countries before running their experiments, e.g. on Electricity Maps. In some cases, if users are running on clouds that have different geographic locations, users could choose where to run the algorithms to emit fewer GHGs. For example, table 5 presents some values at different locations for two different days. While it can be wise to carefully choose datacenter locations, developers must keep in mind that transferring large datasets from one location to the other also has environmental impacts (section 3.1.4). Therefore, depending on the training time, it might be better to remain on the same server when training on the same large dataset. We present in table 6 how each tool handles carbon intensity.

Table 5: Daily average carbon intensity for two different days. Data taken from Electricity Maps

	March 5th 2023	March 29th 2023
France	64	137
North Sweden	16	14
South Africa	684	702
South Carolina - USA	432	786

Table 6: Emission intensity used in the different tools

Green-Algorithms	Most emission intensity data come from Carbon Footprint but the tool also uses other sources like Electricity Maps. Information is collected in the CI_aggregated.csv file. The default value is 475 gCO2eq/kWh (world average in 2018).
CodeCarbon	For United States and Canada, CodeCarbon uses regional data on emissions per unit of power consumed. For other countries, the tool uses the energy mix of the country, i.e. intensity data of each energy source (carbon, solar, wind, etc.), to calculate the intensity of the country. The average energy mix for each country is taken from Global Petrol Prices. The information is collected in the files under data folder. The sources of each data are specified in the files. The default value is 475 gCO2eq/kWh (world average in 2018).
Eco2AI	For all countries the emission intensity calculation was made using the intensity data of each energy source (carbon, solar, wind, etc.) and the energy mix of each country. The values used for the calculations nor their sources are not explained, and only the final result of the intensity of emissions for each country is published in carbon_index.csv. The default value is 436.5 gCO2eq/kWh [35].
CarbonTracker	CarbonTracker supports the fetching of carbon intensity in real-time through external APIs. It is currently limited to Denmark and Great Britain. For Denmark they use data from Energi Data Service and for Great Britain they use the Carbon Intensity API. For other countries, it uses fixed values available in the carbon-intensities.csv file. The sources are not published. The default value is 475 gCO2eq/kWh (2019).
EIT	EIT supports the fetching of carbon intensity in real-time through external APIs. It is currently limited to California using the API of California ISO. For other countries, it uses fixed values available in the co2eq_parameters.json file. The sources are published and are mostly from Electricity Maps. The default value is 301 gCO2eq/kWh (annual mean carbon intensity of all electricityMap zones).
MLCO2	MLCO2 published the sources and contains the information of the Cloud providers in the impact.csv file. For private infrastructure, it is necessary to provide the emission intensity value, which must be obtained by user own means.
Cumulator	The data of emission intensity is from Electricity Maps. Information is collected in the country_dataset_adjusted.csv file. The default value is 447 gCO2eq/kWh (average carbon intensity value in gCO2eq/kWh in the EU in 2018 [36]).

3.4 Measuring whole equipment consumption with wattmeters

Wattmeters are physical instruments that are used to measure the active electrical energy of a certain circuit. By plugging them into the physical infrastructure, we can get the exact total consumption of the machine. With wattmeters, it is not possible to determine how much energy each component of the machine consumes, neither to discriminate consumption by process. It is also important to note that wattmeters have measurement frequencies. Different wattmeters may have different measurement frequencies and therefore different accuracies depending on the duration of processes.

3.5 Errors reported and found in the tools

Some tools had to be modified to be used, as they had bugs not yet fixed by the authors. The modifications we had to make can be found in B.

3.6 Summary of the characteristics of existing tools

In addition to the tables presented in [10] and [16], we summarize in table 7 what is configurable and what are default values for each component, and add details on usage factor.

4 Infrastructure

Depending on the infrastructure, users will have access to different resources, which restricts the list of tools that can be used. The most commonly used infrastructures for machine learning are physical or virtual servers, virtualized environments in the cloud, supercomputers or personal computers. Table 8 summarizes the tools' requirements and hardware compatibility.

Table 7: Summary of the characteristics of the energy and CO2eq measurement tools. Wattmeters are not included in the table.

	Green-Algorithms	CodeCarbon	Eco2AI	CarbonTracker	EIT	MLCO2	Cumulator
General Information							
1. Type of tool	Online calculator and Server-side tool	Embedded package	Embedded package	Embedded package	Embedded package	Online calculator	Embedded package
2. Embodied emissions	no	no	no	no	no	no	no
3. Static (idle) emissions w/o runs	no	no	no	no	no	no	no
4. Process/machine estimation	process	both	both	machine	process	machine	machine
5. Measurement frequency (sec)	-	15	10	10	10	-	-
Energy Consumption CPU							
1. Measured	yes	yes	yes	yes	yes	no	yes (if chosen)
2. Use Model of CPU	yes	yes (if no tracking tool)	yes	no	no	-	yes
3. Use RAPL files or Power Gadget	no	yes	no	yes (RAPL files)	yes	-	no
4. Default TDP	12 (normalized by core)	85	100	-	-	-	250
5. Usage Factor considered	yes	50% (if default TDP used)	yes	no	yes	-	no
6. Tool for usage factor	-	-	psutil	-	psutil	-	-
Energy Consumption GPU							
1. Measured	yes	yes	yes	yes	yes	yes	yes (if chosen)
2. Use Model of GPU	yes	no	no	no	no	yes	yes
3. Default TDP	200	no	no	no	no	no	250
4. Tool to get power	-	pynvml	pynvml	pynvml	nvidia-smi	-	-
5. Usage Factor considered	yes	no	no	no	yes	no	no
6. Tool for usage factor	-	-	-	-	nvidia-smi	-	-
7. Only Nvidia GPUs	no	yes	yes	yes	yes	no	no
Energy Consumption Memory							
1. Measured	yes	yes	yes	yes	yes	no	no
2. Source of information	-	system	system	RAPL files	RAPL files	-	-
3. Usage Factor considered	no	yes (if tracking mode)	yes	no	yes	-	-
4. Tool for usage factor	-	psutil	psutil	-	psutil	-	-
5. Formula	0.3725 W/GB	0.375 W/GB	0.375 W/GB	-	-	-	-
Emission intensity							
1. Default E.I value	475	475	436.5	475	301	-	447
2. Real time	no	no	no	yes (just UK and Denmark)	yes (just California)	no	no
PUE							
1. PUE considered	yes	yes (just cloud)	yes	yes	yes	no	no
2. PUE configurable	yes	no	yes	no	yes	-	-
3. Default PUE value	1.67	-	1	1.58	1.58	-	-
Errors							
1. Need code modification	-	-	-	yes (with Python 3.10)	yes	-	yes

Table 8: Requirements to run the tools.

	Green-Algorithms	CodeCarbon	Eco2AI	CarbonTracker	EIT	MLCO2	Cumulator
Requirements							
1. Operating System	-	-	-	Linux (if no-NVIDIA GPU)	-	-	-
2. Access to RAPL files	no	no	no	yes (if no-NVIDIA GPU)	yes	no	no
3. Power Gadget	-	no	no	-	yes	-	no
Compatibility							
1. Non Intel CPUs	yes	yes	yes	no	no	does not measure CPU	yes
2. Non Nvidia-GPUs	yes	no	no	no	no	yes	yes

4.1 Access to information and resources

We explain below how each type of infrastructure handles access to hardware information.

Virtual environments Some tools require knowing the available CPU model to make a better estimation. In virtual environments, the information in the `/proc/cpuinfo` file (or equivalent tools for Windows or macOS) may not be correct, and may represent some characteristics of the CPU emulated by the virtualizer. Unfortunately, from the virtual environment, there is no way for users to know exactly the real CPU that is being used for the execution.

RAPL files Some tools require read access to the RAPL files. Access to these files is restricted by default to the root user. An administrator must be asked to grant read permission to those files. Also, these files are available only if the machine has Intel CPUs, and has Linux as an operating system. A similar situation is experienced with Power Gadget: it is exclusive to Intel CPUs, and the tool need to be installed.

Usage factor Unfortunately, there is no tool that can be used with the command line that gives us the total time of the script (whole time), the CPU time and the GPU time, in order to calculate the CPU and GPU usage factor required by Green-Algorithms. However, workload managers such as SLURM commonly log this information. One option is to take empirical and specific measurements of the use of the GPU during the execution of their algorithm using the `nvidia-smi` tool, and extrapolate that value of GPU utilization to the entire execution. It is important to note that this utilization percentage corresponds to the total utilization, and not just the utilization of the process. There could be

other processes running on the available GPUs. Up to our knowledge, there is also no tool that measures GPU time for non-Nvidia GPUs.

In addition, when calculating the CPU usage factor, it is important to consider whether the infrastructure where the process is running has hyperthreading enabled. When hyperthreading is available and enabled, the hardware components of one physical core are shared between several threads. Each thread has its own set of registers, but most resources of the core are shared between the threads. Estimating the real usage factor can be difficult in this scenario. According to some studies, the maximum capacity is up to 30% more than without hyperthreading [3].

Wattmeter Finally, using a wattmeter requires having one, and in the case of institutional infrastructure, consulting with a systems administrator to make the physical connection. It is important to note that the wattmeter will measure the consumption of the entire node, so ideally there should not be other processes running on the node, or if there are, it is key to take it into account when analyzing the value returned by the device.

4.2 Description of the infrastructures used for experimentation

In this guide we have tested on resources in two French laboratories (Labri and MAP5), Grid5000, personal computers and we will also mention Google Colab. In table 9 we detail the hardware specifications of the infrastructure used for the experiments.

Table 9: Hardware specifications of infrastructure used for experiments

	Gemini-1 (Grid5000)	**Rosenblatt (MAP5)**	**Server (Labri)**	**Personal Computer**	**Colab**
Operating System	Linux	Linux	Linux	Linux	Linux
CPU					
1. Quantity	2	2	1	1	1
2. Model	Intel Xeon E5-2698 v4	Intel Xeon E5-2609 v4	Intel Core i9-7940X CPU @ 3.10GHz	AMD Ryzen 5 2600 Six-Core Processor	(VE) Intel Xeon CPU @ 2.20GHz
3. TDP	135W	85W	165W	65W	Unknown
GPU					
1. Quantity	8	2	3	1	1
2. Model	NVIDIA Tesla V100-SXM2-32GB	NVIDIA TITAN Xp	NVIDIA TITAN Xp	NVIDIA TITAN V	NVIDIA Tesla T4
3. TDP	250W	250W	250W	250W	70W
Memory					
1. Quantity	512 GB	62 GB	126 GB	32 GB	12 GB
Wattmeters					
1. Available	yes	no	yes	yes	no
2. Frequency	second	-	minute	minute	-

4.2.1 Laboratory servers

We have tested the different measuring tools in Labri (computer science laboratory of Bordeaux) and MAP5 (laboratory of applied mathematics in Paris 5 University). Labri has physical servers with NVIDIA GPUs, Intel CPUs and Linux operating system. We have had the possibility to experiment using Wattmeter. Access to the RAPL files is restricted to root, so the execution of the scripts need to be done by an administrator, in order to use Experiment-Impact-Tracker and CarbonTracker.
MAP5 has physical servers with NVIDIA GPUs, Intel CPUs and Linux operating system. Access to the RAPL files is currently available. We can test all the tools, but we do not have a Wattmeter.

4.2.2 Super computers

We experimented one super computer: Grid5000 which is a large-scale and flexible testbed for experiment-driven research in all areas of computer science, with a focus on parallel and distributed computing including Cloud, HPC and Big Data, and AI. Grid5000 cluster allows numerous configurations and is very well documented. The cluster has servers with NVIDIA GPUs, Intel CPUs, Linux operating system and access to RAPL files. Access to Wattmeter measurements on selected nodes is possible, so that all the tools can be used.

However, by requesting only a portion of the node, the wattmeter value, that measures the entire node, might not be really useful as other jobs can be running in the same server. Also, note that without booking the whole node, it is not possible to get user privileges so EIT cannot be used, Carbontracker will not measure CPU, and CodeCarbon will use TDP to calculate CPU consumption.

[3] Source: https://www.intel.com/

4.2.3 Personal computers

In these machines, we could install the necessary tools and enable the permissions that are required. CarbonTracker can be used if at least one of the 2 conditions is met: having Intel CPUs or NVIDIA GPUs. If neither of the two conditions is met, the tool cannot be used. The tool will measure the power consumption of the CPUs and Memory only if the CPUs are Intel, and it will measure the power consumption of the GPUs only if they are Nvidia.

If we have non-NVIDIA GPUs, we can only use Green-Algorithms, MLCO2 (if the GPU is on the list), Cumulator and CarbonTracker (if we have Intel CPUs).

If we have non-Intel CPUs, we will not be able to use Experiment-Impact-Tracker and if we have only CPUs, we will not be able to use MLCO2 either. This explains the N/A value reported in results tables.

4.2.4 Colab

Google Colab is a widely use resource, with data centers located around the world, but unfortunately the data center cannot be selected when the environment is created. The execution location can be checked with the command `curl ipinfo.io` and then using this information to determine the data center being used [4].

When running a notebook, a virtual environment is generated, for which some commands are not available, users are not administrators, do not have access to RAPL files and do not know the real resources that are being used. This limits the tools that can be used. Experiment-Impact-tracker cannot be used. Green-Algorithms, CodeCarbon, Eco2AI and Cumulator can be used, assuming an average consumption. This assumption can lead to reporting values of carbon emissions that are not the correct real ones. CarbonTracker can be used, but only with GPU runtime, and will not measure energy consumption of CPU nor Memory.

5 Experiments and results analysis

We will now compare the different tools and their use in different infrastructures for image processing and analysis. Section 5.1 details the experimental settings. Then, in section 5.2 we present the results. In section 5.2.1 we explain the high variability between the different tools, their differences with wattmeter measurements (section 5.2.2) and the impact of the infrastructure (section 5.3). Later, focusing more on the second experiment, we analyze the influence of the data load (section 5.4), of the batch size (section 5.5), of saving the checkpoints (section 5.6) and of the energy consumption of the tools themselves (section 5.8). Finally, we comment on additional idle consumption (section 5.9).

The theoretical analysis of the tools and results provides a better understanding of differences in measurement between the tools, which [10] indicated was needed.

In order to also transparently acknowledge the impact of our work, we conducted an analysis using wattmeters when available and CodeCarbon when not (machine tracking) to determine the total energy consumed throughout all our experiments. The results revealed a cumulative consumption of approximately 14.5 kWh. This value includes all the runs that led to the paper. It does not include PUE.

5.1 Experiments settings

We carried out two experiments, with different characteristics, in different infrastructures.

First, we trained a manually written digit classifier on the MNIST dataset. The MNIST dataset is a collection of images of handwritten digits. Its training set has 60,000 examples, with a size of 50 MB.The classifier is implemented with a fully connected, two-layer network (an inner layer of 32 neurons, and an output layer of 10 neurons), over 5 epochs and normally takes less than a minute on different infrastructures. This experiment runs on a single GPU.

Second, we trained an image denoiser on the Imagenet validation dataset. The ImageNet dataset is a collection of images depicting diverse objects and scenes. Its validation set has 50,000 examples, with a size of 6 GB. The Denoiser is implemented with a DnCNN network [37] over 80 epochs and takes approximately two hours to run. This experiment runs in parallel on all available GPUs. In order to measure the impact of other configurations, small variations of this experiment were also performed.

The experiments were performed using Pytorch. Since each experiment has a different configuration regarding the use of the GPUs, the choice of framework is key to enable the use of all available GPUs. PyTorch enabled multi-GPU training. This is also the case with Tensorflow, but it would have requirer additional configuration to the default installation in order to use the available GPUs.

[4] https://cloud.google.com/about/locations?hl=es

The experiments were carried out in the infrastructures detailed in section 4. We also ran the experiments on Gemini-1 requesting only a quarter of the resources (two GPUs, 128 GB memory and 10 cores of the 40 available). Depending on the available resources, certain tools could be used only on some infrastructures. We now discuss the main observations from our results.

5.2 Results

This section presents and analyzes the results obtained for the two experiments on the different infrastructures. In table 10 we present the energy consumption for the first experiment, which corresponds to the training of a manually written digit classifier. In table 11 we present the consumption for the second experiment, which corresponds to the training of an image denoiser.

The reported values correspond to individual runs and are not averaged values. However, multiple runs of the experiments were performed on different infrastructures to validate the consistency of these numbers. Experiment 1 was executed 3 times on Grid5000, 2 times on MAP5, Labri and Colab. Experiment 2 was executed twice on Grid5000 and Labri.

As said before, Cumulator does not report energy consumed. The values presented in the table were not reported by Cumulator, but calculated by us from carbon footprints.

	Green-Algorithms	CodeCarbon (P)	CodeCarbon (M)	Eco2AI (P)	Eco2AI (M)	CarbonTracker	EIT	MLCO2	Cumulator	Wattmeter
Gemini-1 Whole node (57 sec)										
Tot. Energy reported	5.990	8.800	12.50	7.200	7.100	13.30	2.570	38.00	4.771	
Tot. Energy w/o PUE	3.590	8.80	12.50	7.200	7.100	8.580	1.630	38.00	4.771	13.00
Energy for CPU	0.007	1.500	1.500	-	-	-	-	-	-	-
Energy for GPU	0.395	7.200	7.200	-	-	-	-	-	-	-
Energy for Memory	3.16	0.0184	3.700	-	-	-	-	-	-	-
Carbon emissions	0.307	0.480	0.690	0.490	0.480	0.777	0.140	2.53	0.563	
Gemini-1 2 GPUs (56 sec)										
Tot. Energy reported	1.689	1.630	4.570	1.640	1.620	2.350	N/A			
Tot. Energy w/o PUE	1.008	1.630	4.570	1.640	1.620	1.516	N/A	9.333	3.729	N/A
Energy for CPU	0.130	0.000	0.000	-	-	-		-	-	
Energy for GPU	0.139	1.620	1.620	-	-	-		-	-	
Energy for Memory	0.739	0.013	2.950	-	-	-		-	-	
Carbon emissions	0.086	0.090	0.250	0.110	0.110	0.140		0.622	0.440	
Rosenblatt (1min 36 sec)										
Tot. Energy reported	1.030	3.190	3.800	2.000	2.100	4.56	3.860	13.30	6.711	
Tot. Energy w/o PUE	0.617	3.190	3.800	2.000	2.100	2.940	2.440	13.30	6.711	N/A
Energy for CPU	0.148	1.200	1.200	-	-	-	-	-	-	
Energy for GPU	0.086	1.900	1.900	-	-	-	-	-	-	
Energy for Memory	0.389	0.0276	0.600	-	-	-	-	-	-	
Carbon emissions	0.0527	0.170	0.200	0.138	0.140	0.266	0.210	0.533	0.792	
Labri (45 sec)										
Tot. Energy reported	1.94	1.689	2.287	1.1459	1.126	2.219	1.91	9.375	2.093	
Tot. Energy w/o PUE	1.16	1.689	2.287	1.1459	1.126	1.432	1.209	9.375	2.093	2.241
Energy for CPU	0.255	0.565	0.565	-	-	-	-	-	-	-
Energy for GPU	0.128	1.111	1.097	-	-	-	-	-	-	-
Energy for Memory	0.777	0.013	0.626	-	-	-	-	-	-	-
Carbon emissions	0.099	0.093	0.126	0.074	0.076	0.13	0.107	0.375	0.247	
Personal computer (57 sec)										
Tot. Energy reported	0.356	1.000	1.190	0.733	0.728	1.415	N/A	4.167	3.949	
Tot. Energy w/o PUE	0.356	1.000	1.190	0.733	0.728	0.913	N/A	4.167	3.949	1.404
Energy for CPU	0.032	0.330	0.330	-	-	-		-	-	-
Energy for GPU	0.125	0.660	0.660	-	-	-		-	-	-
Energy for Memory	0.199	0.015	0.195	-	-	-		-	-	-
Carbon emissions	0.018	0.056	0.065	0.049	0.049	0.083		0.167	0.466	
Colab - Oregon (1 min 6 sec)										
Tot. Energy reported	0.381	1.500	1.600	3.000	3.000	0.805	N/A	1.280	5.15	
Tot. Energy w/o PUE	0.343	1.500	1.600	3.000	3.000	0.519	N/A	1.280	5.15	N/A
Energy for CPU	0.219	0.900	0.900	-	-	-		-	-	
Energy for GPU	0.041	0.600	0.600	-	-	-		-	-	
Energy for Memory	0.0913	0.0206	0.100	-	-	-		-	-	
Carbon emissions	0.024	0.200	0.200	0.600	0.600	0.290		0.367	1.03	

Table 10: Results for the training of a digit classifier (experiment 1). All consumption values are in Wh. Carbon emissions are in gCO2e. For CodeCarbon and Eco2AI, (P) refers to the process tracking mode and (M) to the machine tracking mode.

5.2.1 Variability between the different tools

From the two tables 10 and 11, we observe a large difference between the energy consumption and carbon emissions reported by the different tools. For instance, a 400% increase of consumption for MLCO2 compare to Eco2AI on the Gemini-1 node of Grid5000.

Machine vs Process Some tools are focused on estimating the consumption of the entire machine, and are comparable with wattmeters, but others estimate the consumption of the process, trying to isolate it from other processes that may be running on the machine.

	Green-Algorithms	CodeCarbon (P)	CodeCarbon (M)	Eco2AI (P)	Eco2AI (M)	CarbonTracker	EIT	MLCO2	Cumulator	Wattmeter
Gemini-1 whole node (2 hs)										
Total Energy reported	1.92	1.39	1.69	1.07	1.10	2.09	2.09	4.80	0.5	
Tot. Energy w/o PUE	1.15	1.39	1.69	1.07	1.10	1.35	1.32	4.80	0.5	2.10
Energy for CPU	0.09	0.22	0.22	-	-	-	-	-	-	-
Energy for GPU	0.69	1.14	1.09	-	-	-	-	-	-	-
Energy for Memory	0.37	0.03	0.37	-	-	-	-	-	-	-
Carbon emissions	100	80	90	70	80	120	120	280	60	
Gemini-1 2 GPUs (1h 17 min)										
Total Energy reported	0.76	0.36	0.61	0.35	0.37	0.59	N/A	0.77	0.45	
Tot. Energy w/o PUE	0.47	0.36	0.61	0.35	0.37	0.38	N/A	0.77	0.45	N/A
Energy for CPU	0.05	0	0.00	-	-	-		-	-	
Energy for GPU	0.36	0.359	0.37	-	-	-		-	-	
Energy for Memory	0.06	0.008	0.24	-	-	-		-	-	
Carbon emissions	40	20	34	24	25	34		51	38	
Rosenblatt (3hs 16 min)										
Tot. Energy reported	1.77	1.07	1.12	0.89	0.99	1.71	1.75	1.63	0.84	
Tot. Energy w/o PUE	1.06	1.07	1.12	0.89	0.99	1.10	1.11	1.63	0.84	N/A
Energy for CPU	0.10	0.17	0.17	-	-	-	-	-	-	
Energy for GPU	0.88	0.89	0.87	-	-	-	-	-	-	
Energy for Memory	0.08	0.02	0.08	-	-	-	-	-	-	
Carbon emissions	90	60	60	60	70	100	100	90	100	
Labri (1h 13 min)										
Tot. Energy reported	0.80	0.76	0.79	0.69	0.72	1.16	1.17	0.9	0.3	
Tot. Energy w/o PUE	0.48	0.76	0.79	0.69	0.72	0.75	0.74	0.9	0.3	0.83
Energy for CPU	0.15	0.097	0.097	-	-	-	-	-	-	-
Energy for GPU	0.27	0.66	0.64	-	-	-	-	-	-	-
Energy for Memory	0.06	0.03	0.056	-	-	-	-	-	-	-
Carbon emissions	41	42	44	47	48	68	65	36	24	
Personal computer (1h 49 min)										
Tot. Energy reported	0.37	0.34	0.35	0.25	0.27	0.52	N/A	0.45	0.46	
Tot. Energy w/o PUE	0.37	0.34	0.35	0.25	0.27	0.34	N/A	0.45	0.46	0.40
Energy for CPU	0.001	0.09	0.09	-	-	-		-	-	-
Energy for GPU	0.35	0.24	0.24	-	-	-		-	-	-
Energy for Memory	0.02	0.01	0.02	-	-	-		-	-	-
Carbon emissions	19	19	19	17	18	30		18	54	
Colab - Oregon (17 hs est.)										
Tot. Energy reported	1.22	1.49	1.56	1.03	1.82	0.96	N/A	1.19	0.36	
Tot. Energy w/o PUE	1.10	1.49	1.56	1.03	1.82	0.62	N/A	1.19	0.36	N/A
Energy for CPU	0.07	0.73	0.73	-	-			-		
Energy for GPU	0.95	0.75	0.75	-	-	-		-	-	
Energy for Memory	0.08	0.02	0.08	-	-	-		-		
Carbon emissions	199	206	216	184	328	369		100	72	

Table 11: Results for the training of an image denoiser (experiment 2). All consumption values are in kWh. Carbon emissions are in gCO2e. The consumption indicated for Colab is extrapolated. An epoch was executed, the consumptions were obtained, and the values were extrapolated.

CodeCarbon and CarbonTracker have similar strategies for GPU and CPU consumption estimation, focusing on full machine estimation. They differ in method for the estimation of memory consumption. We can say that CodeCarbon strategy is more accurate, since it reaches a value more similar to that of the wattmeter.

Eco2AI and EIT focus more on isolating the consumption of the process that is measured. It can be seen from both experiments that these tools show a lower consumption estimate than CodeCarbon and CarbonTracker. Green-Algorithms approach also attempts to isolate consumption from the process.

Multiple GPUs Cumulator only measures CPUs or GPUs, according to what we specify when creating the tracker. In both cases it considers a single unit of the hardware it is measuring, without checking how many CPUs or GPUs exist on the machine.

MLCO2 also has a simplified view, only measuring the consumption of 1 GPU. The values reported in the tables were obtained by multiplying the value obtained by the number of GPUs available. The reported values for 1 GPU for Cumulator and MLCO2 are very similar because they follow the same strategy. In the case of the personal computer or Colab, having a single GPU, we can come to consider these two tools, but we are also not measuring CPU consumption. In addition, the tools only multiply the time consumed by the TDP, so it does not verify actual consumption or compute usage factors. Their results can only be useful when we have a single unit of the component to be measured (CPU or GPU for Cumulator, and only GPU for MLCO2), and it has a usage factor close to 100%.

Usage factor The web calculator of Green-Algorithms and their server tool G4HPC set default usage factor to 100% CPU and GPU loads if these data are not provided. This will overestimate power consumption in most cases. To be considered by GA, usage factors must be calculated by the user. The CPU usage factor can be calculated using the CPU time and the process time, but there is no easy way to get the GPU usage factor. We can get empirical values from measurements using `nvidia-smi` while the algorithm is running, and assume that it maintains that usage factor throughout the run. In this case we are assigning all the utilization percentage reported by `nvidia-smi` to the process, but there could be other processes using the GPU. In our study, since for both experiments the only process running on GPUs was the one measured, we took one sample per epoch of the `nvidia-smi` output during code execution. We averaged the utilization percentage values of all GPUs across all samples. Results are shown in table 12. We observe a low usage factor, especially on servers. As shown in table 11, MLCO2 seems to largely overestimate consumption on

Grid5000, which is because it does not take into account the usage factor of the GPUs. The average usage factor is 14%, but MLCO2 is considering 100% for all 8 GPUs. EIT queries and calculates usage factors during execution. Eco2AI only does this for CPU, as it directly queries the consumed energy for GPU. CodeCarbon and CarbonTracker directly query the consumed energy for both GPU and CPU, without using the usage factor.

Table 12: Usage factor of CPU and GPU in the infrastructures used. This computed values are used by Green-Algorithms.

	CPU Expe. 1	GPU Expe. 1	CPU Expe. 2	GPU Expe. 2
Gemini-1 (Grid5000)	5%	0.3%	16%	14%
Gemini-1 2 GPUs (Grid5000)	12%	1.5%	58%	46%
Server (Labri)	9%	1%	73%	35%
Rosenblatt (MAP5)	16%	1%	39%	54%
Personal Computer	22%	3%	4%	77%

5.2.2 Comparison between software tools and wattmeter

Wattmeters were present in Labri server, the personal computer and Gemini-1. Table 13 shows a summary of the comparison presented. For experiment 1, wattmeter on the personal computer and labri server only made one measurement during the entire experiment, so the reported value may not be exact.

In the first experiment, the value reported by the consumption of the machine with CodeCarbon is almost exactly the same as that reported by the wattmeter. For the second experiment the value is not as precise, but it is still more than 80% for all infrastructures. This measuring tool is the one that gives the closest value with respect to wattmeters, followed by CarbonTracker, with more variability between infrastructures.

Eco2AI and EIT report values larger than the wattmeter. Since these tools try to isolate the consumption of the process, and not measure the total consumption of the machine, then the reports of energy consumption are not comparable with the wattmeter value.

	CodeCarbon (M)	Eco2AI (M)	CarbonTracker	EIT
Expe. 1 Grid5000	96%	55%	66%	13%
Expe. 2 Grid5000	80%	60%	64%	63%
Expe. 1 Personal comp.	85%	52%	65%	N/A
Expe. 2 Personal comp.	88%	68%	85%	N/A
Expe. 1 Labri	102%	50%	64%	54%
Expe. 2 Labri	95%	87%	90%	89%

Table 13: Comparison between software tools and wattmeter in Grid5000 (without considering PUE), personal computer and Labri server. Values represent the percentage of energy reported by tools wrt the value reported by the wattmeter.

5.3 Influence of infrastructures

We ran the same experiments on different infrastructures. For both experiments, power consumption is higher on larger infrastructures (e.g. Gemini-1).

As an example, the Denoiser training experiment took 2 hours on Gemini-1 (Grid5000 server), while on Rosenblatt (MAP5 server) it took 3 hours and 16 minutes. Usage factor of CPU was lower in Grid5000: 16% in Grid5000 and 39% in MAP5. The estimation of usage factor of GPU was also lower in Grid5000: 14.3% in Grid5000 while in MAP5 it was 54%. The consumption reported in Gemini-1 by CodeCarbon (Machine tracker) is 1.69 kWh, while the consumption reported in Rosenblatt by CodeCarbon (Machine tracker) was 1.12 kWh. Rosenblatt's hardware is considerably smaller than Gemini-1's (see table 9).

It can also be seen that in experiment 2 for Labri, the personal computer and on Gemini-1 booking only 2 GPUs, the execution time was less than in the case of execution on the entire Gemini-1 node. This longer execution is more likely due to the parallelization strategy (using nn.DataParallel) that runs the training on all GPUs without requiring their full computing power.

This might be a good reason for using, as much as possible, a hardware which size is adapted to the experiments where resources can be used as much as possible, even if the experiments take more time. Gemini-1 node has 8 GPUs which is not useful for both our experiments.

5.4 Data load

In the Denoiser training experiment, we seperately quantified the energy consumption of data loading (6GB Imagenet validation split) vs training the model and found that only 0.5% of the energy was used to load the data. This is partly because the data was already on the server, the impact of downloading the data and of data storage is not being measured.

5.5 Batch size

To study the impact of batch size during training, we used CodeCarbon during experiment 2 (denoiser) on the Gemini-1 node for 10 epochs. Using three batch sizes (32, 64 and 128), we showed that there is a tradeoff between energy used and runtime (Table 14). While larger batch sizes led to faster runtimes, the largest energy usage was measured for the smallest batch size (32), closely followed by the largest one (128). In this situation, an intermediate batch size of 64 looks like a better compromise, combining a runtime not far off the shortest one and minimising energy usage.

However, when we decrease the batch more, the experiment takes longer, and the idle consumption of the resources starts to weigh on the total consumption of the experiment. If we compare the GPU consumption of experiments with batch size 32 and 128, we see that experiment 32 consumes less, still taking almost 3 times longer. Nevertheless, comparing the experiment of 32 with that of 64, we have that the consumption is higher, probably because the experiment takes almost 10 minutes more, and we have the static consumption of the resources.

In conclusion, a balance is required between the length of the experiment, and the greater consumption of the GPU memory to obtain a minimum energy consumption.

	Experiment with batch size 32	Experiment with batch size 64	Experiment with batch size 128
Total Energy (CodeCarbon)	252	184	246
CPU (CodeCarbon)	41	29	20
GPU (CodeCarbon)	205	152	224
Memory (CodeCarbon)	6	3	2.3
Total Energy (Wattmeter)	391	280.3	320
Time spent	25:54	16:29	10:30

Table 14: Results of experiment 2 with different batch sizes. All consumption values are in Wh.

5.6 Checkpoints

We found that checkpointing had no impact on energy consumption or runtime (Table 15). We tested this on experiment 2 on Gemini-1 using CodeCarbon and a wattmeter. In the first scenario, the values of the network parameters were saved every epoch (ten epochs in total) and in the second scenario values were saved only once.

	Experiment with one checkpoint	Experiment with ten checkpoints
Total Energy reported (CodeCarbon)	161	160
Energy for CPU (CodeCarbon)	24	24
Energy for GPU (CodeCarbon)	134	133
Energy for Memory (CodeCarbon)	3	3
Total Energy reported (Wattmeter)	206	206
Time spent (min)	14:10	13:47

Table 15: Results of experiment with different frequency of checkpoints. Both experiments are run for 10 epochs. On the left column, only one checkpoint has been saved at the end of these epochs. On the right column, one checkpoint is saved per epoch. All consumption values are in Wh.

5.7 Variability of consumption through epochs

It is interesting to determine if it is possible to extrapolate the energy consumption of a training phase from the values observed on only few epochs. To determine it, the Denoiser training experiment was executed during different number of epochs on Gemini-1; the time consumed was measured, as well as the energy consumption. Results in the Fig. 3 show

that epochs duration and consumption are constant. It might therefore be possible to extrapolate energy consumption for large experiments from experiments on just a few epochs. Same conclusion was reached in [21] when using CarbonTracker.

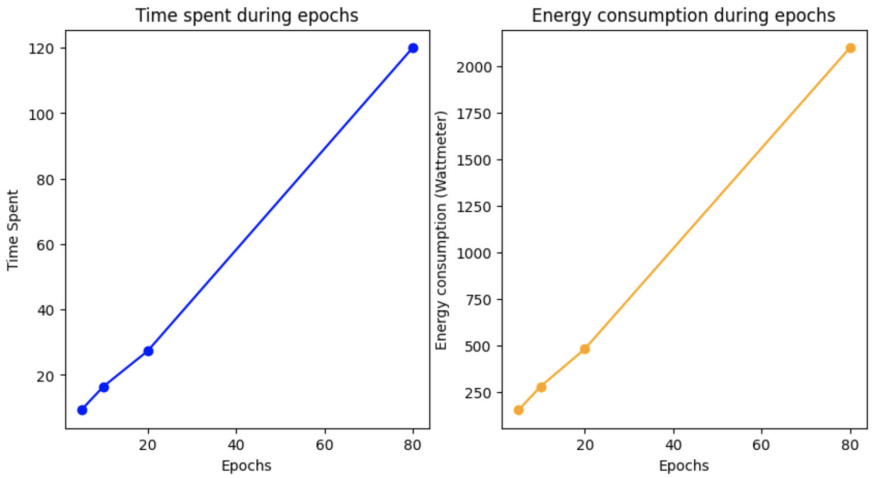

Figure 3: Duration and energy consumption after different number of epochs of Experiment 2. All consumption values are in Wh.

5.8 Is measuring really eco-friendly?

To compare the extra energy consumption of the tools themselves, we run 2 processes of experiment 2 in parallel, one with all seven trackers and one without any. We report energy consumption provided by the wattmeter. We found that the code with trackers was almost 10% slower and ended 11 minutes later than the one without trackers. The energy consumption during this extra time was 0.19 kWh, while it was 2.58 kWh for the time when both processes were running in parallel (+7.4%).

Another experiment was performed, testing each tracker at a time. As in previous test, we run two processes in parallel, one with a given tracker and one without any tracker. Energy is measured with wattmeter.

Table 16 shows the result with 10 epochs. It can be seen that the additional energy is around 1% of the total consumption for all the tools, except for Eco2AI, where consumption reaches 3.5%, a value that is not negligible. We think that the biggest consumption compared to the other tools is not using the RAPL files to obtain the memory and CPU consumption, but rather making queries to the operating system to later do the calculations. Although other tools also do it this way, none do it to calculate the energy of both resources.

	CodeCarbon(P)	Eco2AI(P)	CarbonTracker	EIT	Cumulator
Run time w/ tracker (min)	15:09	15:33	16:35	16:29	15:02
Run time w/o tracker (min)	15:05	14:57	16:24	16:35	14:49
Extra time with tracker (min)	0:04	0:36	0:11	-0:06:00	0:13
Energy Cons.when 2 processes running (Wh)	335.5	334	358	358.5	331.6
Energy Cons. during extra time (Wh)	3.1	12.2	4.29	0	5.4
Percentage of overload (%)	0.92	3.5	1.2	0	1.6

Table 16: Results running experiment 2 twice in parallel on Gemini-1: one process using trackers, the other without.

As a result of both experiments, we can conclude that measuring the processes has an impact, but a small one. The first experiment carried out with all the trackers has a longer execution time, probably due to delays while access to resources. It might be a good idea to use online tools such as Green-Algorithms, in order not to add additional load to the algorithm and still being able to measure the impact.

5.9 Static and deployment consumption

All the tools discussed in this guide are limited to quantifying energy consumption while training a deep learning approach. But infrastructures also us energy when nodes are not used or when the final solution is deployed. The authors of [14] studied static infrastructure emission and deployment emissions when training BLOOM, a large language model and found these to be substantial.

We measured the energy consumption of idle resources on Gemini-1 over the same period of time it takes to run experiment 1. In an idle situation, no process is being run beyond those required by the operating system. We performed the same procedure with experiment 2 (executed for 10 epochs). The results are shown in the table 17. Idle energy consumption is around 745Wh. We see that the consumption of idle resources is high comparing with the consumption reported during training: 84.4% for experiment 1 and 72.9% for experiment 2. Note that for both experiments, the resources are not fully used. In the table 12 we can see the percentage of CPU and GPU utilization during.

Table 17: Static (Idle) and dynamic energy consumption measured with wattmeters

	Time (min)	Energy consumption (Wh)
Experiment 1	00:53	12.96
Idle	00:53	10.95
Experiment 2 (10 epochs)	16:29	280.3
Idle	16:29	204.4

This result is interesting since we can see that most of the consumption occurs simply by having the hardware available to use it. This tells us that we have to be very careful when leaving hardware on for availability. The availability and immediacy of resources is very expensive in terms of energy consumption. When we use hardware where we do not have the power to turn it off when we are not using it, such as the cloud, or shared computers, we must remember that there is an additional consumption to be able to make a reservation at any time for a given resource.

6 Discussions

This section summarizes our observations and anticipates questions that AI practitioners may have when starting to measure the energy consumption of their codes.

6.1 When to measure impacts?

Contrary to tracking tools, online ones like Green-Algorithms make it possible to estimate consumption both after training, as concluded in [10], and before training. Although this will be less precise, it anticipating the environmental impacts of a project.

If we use software tools and perform more than one run, we recommend performing the measurement only for some runs.

Given that is possible to extrapolate the energy consumption of a training phase from the values observed on only few epochs, we could measure the consumption of the firsts epochs, and then estimate the consumption of the total training. In this way, the consumption corresponding to the measurement will be slightly lower.

6.2 Which tools to use

Estimating power consumption using software tools adds small load, so it might be a good idea to use online tools like Green-Algorithms.

Green-Algorithms is the most versatile tool, as it can be used under different infrastructures, brands of CPUs and GPUs. However, online tools requires manual intervention to obtain the information and may be less precise. A first step to remedy this is the tool GA4HPC which is used to obtain the resource reservation data of a job in clusters that use SLURM as workload manager.

MLCO2 is an online tool but is much more limited. It just account for GPU consumption and the value returned must be correctly weighted according to the number of GPUs and the correct execution time of the algorithm.

If we want to use software tools, we found that CodeCarbon is the best tool among those studied to estimate the total consumption of the machine. The consumption reported with it is more accurate when accessing RAPL files. However, a strength of this tool is that it can be used without access to them. On the contrary, if what you want is to isolate the

consumption of the process using software tools, Eco2AI and EIT are those that try to do it. Eco2AI does not require access to the RAPL files and is maintained and updated. By contrast, EIT requires access to the RAPL files and it is necessary to modify the code to use the tool.

6.3 Which infrastructure to use

Since the idle consumption of resources is a large percentage of the total consumption, we recommend only keeping available the resources needed to achieve a high usage factor and have the minimum idle consumption, even if the execution time is longer.

With supercomputers, we recommend requesting only the necessary resources, and if it is adequate, to share the infrastructure with other user processes.

If possible, we recommend turning off personal computers or servers as soon as computation is done.

If we are using cloud infrastructure, as far as possible, choose data centers that have the lowest PUE and that are located in areas with low gas emissions. We recommend choosing low emission hours for the execution of training. Carbon aware schedulers such as CATS, grid-intensity-go or carbon-aware-scheduler can be used to help with this.

6.4 Other impacts

In this paper, we have been focusing only on energy consumption, and associated greenhouse gas emissions, for training AI models. This only a small part of total energy consumption of the complete life cycle of the AI service.

For the training phase, an AI practitioner generally trains the model several times. Complete training emissions should consider all runs. In Green-Algorithms, we can model multiple runs, associated with retraining using the "pragmatic scaling factor" parameter.

As mentioned in previous studies [10, 14, 38], the energy consumption is underestimated, since all the tools only measure the consumption during training and not during deployment. Studies [38, 14] have measured the consumption of deployment phases that can be much higher than the one of training. Here again, choosing appropriate resources to have a high usage factor seems to be essential.

Many other environmental impacts (resource depletion, ecotoxicity, etc.) linked to the life cycle of equipments (manufacturing, transport, distribution, use, end of life), are here not discussed and should be investigated. Even of carbon footprint, computing embodied emissions is a challenge since all data are not made public by manufacturers. From several assumptions, the authors of [14] propose an estimation of embodied emissions equal to half the ones of training.

Datasets creation, transfer and storage are also very important aspects of AI. An estimate by [39] is 0.023 kWh/GB for transferring data on the IP core. For storage, there are various estimates. Following Seagate measurement [5], [9] consider an order of magnitude of the carbon footprint of storing 1 terabyte of data to be around 10 kgCO2e per year. Another study [40] mention 52 Wh for storing one gigabyte for one year. To know more about energy management techniques for database systems, we refer the reader to the systematic review [41].

6.5 Predicting impacts

Systematically estimating the carbon footprint of AI project can raise awareness, encourage the development of energy-efficient software and limit the waste of resource [9]. Importantly, these impacts should be anticipated before the start of a project. Authors of [42] propose a list of criteria for assessing the environmental impacts of projects involving Artificial Intelligence (AI) methods. In addition to measuring while training or deploying an AI model, AI users should try to anticipate as much as possible the impacts of their computations are likely to have, as well as the behavioral, economic, or societal changes that might be induced by the project. In the same line, [43] review ethics, explainability, responsibility, and accountability concepts in AI and propose a model for sustainable AI in the public sector.

7 Conclusion

In this paper we have presented and analyzed seven existing tools for estimating energy consumption when training a deep learning model. We have explained the specificities of each tool and detailed the notions that may be not well

[5] https://www.seagate.com/gb/en/global-citizenship/product-sustainability/

known by AI practitioners. From our study, we have drawn some analysis and recommendations in previous sections. Remark that our two experiments were related to training regular CNNs for image processing and analysis. We believe that the main results would hold for other types of architectures, as carbon footprint estimators have shown the same behaviors for other applications or workloads in [16, 10, 12]. In the paper we have highlighted the advantages and limits of online tools, and that the choice of the software tool depends on the infrastructure and on either one wants to measure the whole node or the process only. We have also shown that measuring with software tools has a small impact that can become non negligible for large experiments. We observed that consumption is constant through epochs, and therefore measuring only on few epochs and extrapolating can be sufficient. We have confirmed that it is important to train models on infrastructures that is scaled to the need, not booking a whole node when not necessary. Finally, all these tools measure only dynamic energy consumption of computing and further studies are required to include static consumption and environmental impacts.

8 Acknowledgments

This study has been carried out with financial support from the French Research Agency through the PostProdLEAP project (ANR-19-CE23-0027-01). Loïc Lannelongue was supported by core funding from the British Heart Foundation (RG/18/13/33946); the NIHR Cambridge Biomedical Research Centre (BRC-1215-20014; NIHR203312)[*]; the Cambridge British Heart Foundation Centre of Research Excellence (RE/18/1/34212); and the BHF Chair Award (CH/12/2/29428). *The views expressed are those of the authors and not necessarily those of the NIHR or the Department of Health and Social Care. The authors thank Michael Clément and Boris Mansencal for running experiments in Labri and personal computer. We also thank Mathilde Jay, Denis Trystram, Laurent Lefèvre and Anne-Laure Ligozat for fruitful discussions.

References

[1] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, S. Luccioni, T. Maharaj, E. D. Sherwin, S. K. Mukkavilli, K. P. Kording, C. Gomes, A. Y. Ng, D. Hassabis, J. C. Platt, F. Creutzig, J. Chayes, and Y. Bengio, "Tackling Climate Change with Machine Learning," *arXiv:1906.05433 [cs, stat]*, Nov. 2019, arXiv: 1906.05433. [Online]. Available: http://arxiv.org/abs/1906.05433

[2] R. Vinuesa, H. Azizpour, I. Leite, M. Balaam, V. Dignum, S. Domisch, A. Felländer, S. D. Langhans, M. Tegmark, and F. Fuso Nerini, "The role of artificial intelligence in achieving the Sustainable Development Goals," *Nature Communications*, vol. 11, no. 1, p. 233, Dec. 2020. [Online]. Available: http://www.nature.com/articles/s41467-019-14108-y

[3] A. K. Kar, S. K. Choudhary, and V. K. Singh, "How can artificial intelligence impact sustainability: A systematic literature review," *Journal of Cleaner Production*, vol. 376, p. 134120, Nov. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0959652622036927

[4] E. Strubell, A. Ganesh, and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," *arXiv:1906.02243 [cs]*, Jun. 2019, arXiv: 1906.02243. [Online]. Available: http://arxiv.org/abs/1906.02243

[5] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing Carbon: The Elusive Environmental Footprint of Computing," *IEEE Micro*, vol. 42, no. 4, pp. 37–47, Jul. 2022. [Online]. Available: https://doi.org/10.1109/MM.2022.3163226

[6] A. Gupta, C. Lanteigne, and S. Kingsley, "Secure: A social and environmental certificate for ai systems," *arXiv preprint arXiv:2006.06217*, 2020.

[7] A.-L. Ligozat, J. Lefevre, A. Bugeau, and J. Combaz, "Unraveling the hidden environmental impacts of ai solutions for environment life cycle assessment of ai solutions," *Sustainability*, vol. 14, no. 9, p. 5172, 2022.

[8] L. H. Kaack, P. L. Donti, E. Strubell, G. Kamiya, F. Creutzig, and D. Rolnick, "Aligning artificial intelligence with climate change mitigation," Oct. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03368037

[9] L. Lannelongue and M. Inouye, "Carbon footprint estimation for computational research," *Nature Reviews Methods Primers*, vol. 3, no. 1, p. 9, 2023.

[10] N. Bannour, S. Ghannay, A. Névéol, and A.-L. Ligozat, "Evaluating the carbon footprint of nlp methods: a survey and analysis of existing tools," in *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, 2021, pp. 11–21.

[11] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[12] 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 *ACM Conference on Fairness, Accountability, and Transparency*, 2022, p. 1877–1894.

[13] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *Journal of Machine Learning Research*, vol. 21, no. 1, 2020.

[14] S. Luccioni, S. Viguier, and A.-L. Ligozat, "Estimating the carbon footprint of bloom, a 176b parameter language model," *arXiv preprint arXiv:2211.02001*, 2022.

[15] P. Arias, N. Bellouin, E. Coppola, C. Jones, G. Krinner, J. Marotzke, V. Naik, G.-K. Plattner, M. Rojas, J. Sillmann, T. Storelvmo, P. Thorne, B. Trewin, K. Achutarao, B. Adhikary, K. Armour, G. Bala, R. Barimalala, S. Berger, and K. Zickfeld, "Climate change 2021: The physical science basis. contribution of working group i to the sixth assessment report of the intergovernmental panel on climate change; technical summary." IPCC, 2021.

[16] M. Jay, V. Ostapenco, L. Lefèvre, D. Trystram, A.-C. Orgerie, and B. Fichel, "An experimental comparison of software-based power meters: focus on CPU and GPU," in *IEEE/ACM international symposium on cluster, cloud and internet computing*, May 2023. [Online]. Available: https://hal.inria.fr/hal-04030223

[17] A.-L. Ligozat and S. Luccioni, "A Practical Guide to Quantifying Carbon Emissions for Machine Learning researchers and practitioners," MILA ; LISN, Research Report, Jul. 2021. [Online]. Available: https://hal.science/hal-03376391

[18] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: Quantifying the carbon emissions of computation," *CoRR*, vol. abs/2007.07610, 2020. [Online]. Available: https://arxiv.org/abs/2007.07610

[19] K. Lottick, S. Susai, S. A. Friedler, and J. P. Wilson, "Energy usage reports: Environmental awareness as part of algorithmic accountability," 2019. [Online]. Available: https://arxiv.org/abs/1911.08354

[20] S. Budennyy, V. Lazarev, N. Zakharenko, A. Korovin, O. Plosskaya, D. Dimitrov, V. Arkhipkin, I. Oseledets, I. Barsola, I. Egorov, A. Kosterina, and L. Zhukov, "Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai," 2022. [Online]. Available: https://arxiv.org/abs/2208.00406

[21] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," 2020. [Online]. Available: https://arxiv.org/abs/2007.03051

[22] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," 2020.

[23] A. Lacoste, S. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," 2019. [Online]. Available: https://arxiv.org/abs/1910.09700

[24] M. J. Tristan Trebaol, Mary-Anne Hartley and H. S. Ghadikolaei, "A tool to quantify and report the carbon footprint of machine learning computations and communication in academia and healthcare," *Infoscience EPFL: record 278189*, 2020.

[25] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[27] D. Maevsky, E. Maevskaya, and E. Stetsuyk, "Evaluating the ram energy consumption at the stage of software development," in *Green IT Engineering: Concepts, Models, Complex Systems Architectures*. Springer, 2017, pp. 101–121.

[28] M. Hodak, M. Gorkovenko, and A. Dholakia, "Towards power efficiency in deep learning on data center hardware," *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1814–1820, 2019.

[29] A. Karyakin and K. Salem, "A survey of main-memory energy efficiency techniques," in *Proceedings of the 13th International Workshop on Data Management on New Hardware (DaMoN)*. Chicago: ACM, 2017, pp. 1–9.

[30] B. Guo, J. Yu, D. Yang, H. Leng, and B. Liao, "Energy-efficient database systems: A systematic survey," *ACM Computing Surveys*, 2022.

[31] The Shift Project, "Lean ICT, towards digital sobriety," Mar. 2019. [Online]. Available: https://theshiftproject.org/en/lean-ict-2/

[32] U. Institute, "2022 data center industry survey," 2022. [Online]. Available: https://uptimeinstitute.com/uptime_assets/6768eca6a75d792c8eeede827d76de0d0380dee6b5ced20fde45787dd3688bfe-2022-data-center-industry-survey-en.pdf

[33] A. Lawrence, "Is PUE Actually Going Up?" 2019, [Online; accessed March 2023].

[34] ——, "Is PUE Actually Going Up?" 2020, [Online; accessed March 2023].

[35] Ember, "Global electricity review 2022," https://ember-climate.org/insights/research/global-electricity-review-2022/, 2022.

[36] A. Moro and L. Lonza, "Electricity carbon intensity in european member states: Impacts on ghg emissions of electric vehicles," *Transportation Research Part D: Transport and Environment*, vol. 64, pp. 5–14, 2018, the contribution of electric vehicles to environmental challenges in transport. WCTRS conference in summer. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361920916307933

[37] E. K. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," 2019. [Online]. Available: https://arxiv.org/abs/1905.05406

[38] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. A. Behram, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. S. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, and K. Hazelwood, "Sustainable AI: Environmental Implications, Challenges and Opportunities," *arXiv:2111.00364 [cs]*, Oct. 2021, arXiv: 2111.00364. [Online]. Available: http://arxiv.org/abs/2111.00364

[39] J. Malmodin and D. Lundén, "The energy and carbon footprint of the ict and e&m sector in sweden 1990-2015 and beyond," in *ICT for Sustainability 2016*. Atlantis Press, 2016, pp. 209–218.

[40] J. Gröger, R. Liu, L. Stobbe, J. Druschke, and N. Richter, "Green cloud computing," *Life cyclebased data collection on environmental impacts of cloud computing*, 2021.

[41] B. Guo, J. Yu, D. Yang, H. Leng, and B. Liao, "Energy-efficient database systems: A systematic survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–53, 2022.

[42] L. Lefèvre, A.-L. Ligozat, D. Trystram, S. Bouveret, A. Bugeau, J. Combaz, F. Emmanuelle, G. Guennebaud, J. Lefèvre, J.-P. Nicolaï *et al.*, "Environmental assessment of projects involving ai methods," 2023.

[43] C. Wilson and M. van der Velden, "Sustainable ai: An integrated model to guide public sector decision-making," *Technology in Society*, vol. 68, p. 101926, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0160791X22000677

[44] B. Petit, "scaphandre," 2021.

A Methodologies to estimate energy consumption of CPUs and GPUs

This appendix described the two methods used to estimate energy consumption of CPUs and GPUs

Knowing the model of the CPU or GPU, the first method multiplies the TDP provided by the manufacturer by the duration of training to obtain the energy used in kWh. TDP is a specification that indicates the maximum amount of power that a computer processor (CPU or GPU) can dissipate when operating at its maximum performance. It refers to the power consumption under the maximum theoretical load. In general, CPUs with a higher number of cores will have a higher TDP because they require more power to operate at maximum performance. However, the relationship between TDP and the number of cores is not always straightforward. Some CPUs may have a higher TDP even though they have fewer cores, because they are designed to operate at a higher clock speed or have a less efficient architecture.

The second method uses the Intel RAPL (Running Average Power Limit) system management interface integrated in INTEL CPUs or the Power Gadget tool. RAPL allows software to monitor and control the power usage of the processor and its components, such as the CPU cores, memory controllers and GPUs. The Linux powercap driver has the ability to expose the RAPL hardware energy counters by a set of files that can be accessed through the Linux file system. These files make it possible to read the current power usage of the processor and its components, as well as to set power limits to control power usage. Drivers are being developed to get the information from RAPL interface from Windows. A recent implementation is the windows-rapl-driver [6] from the Scaphandre project [44].

Power Gadget is a standalone software application developed by Intel that provides real-time monitoring of the power usage of Intel processors. It does not rely on the RAPL files, but rather uses its own proprietary methods to access and analyze power consumption data. Power Gadget presents power consumption data in a user-friendly graphical interface that displays real-time power usage of the processor, CPU cores, memory controller, and other components. This tool can be used on Windows and macOS.

[6] https://github.com/hubblo-org/windows-rapl-driver

B Bugs fix of some software tools

Some tools must be modified to be used, as they have bugs that have not been fixed by the authors. Here are the changes to make for each one.

B.1 Experiment-Impact-Tracker

- PyPi package is not the latest, and does not not correspond to documentation (issue).
- getiterator in file `/gpu/nvidia.py` must be changed to iter.
- For long runs, the INFO log level is too heavy. Change it to the ERROR level.
- If you have other experiment-impact-tracker logs in the same folder or subfolders, correct the `data_interface.py` file so that the results are shown only from the logs folder that was determined.

B.2 Cumulator

Correct imports in `base.py` (structure defined in this file does not correspond to the structure of the package. issue)

B.3 CarbonTracker

Correct decode deprecated function in Python 3.10 in file *carbontracker/components/gpu/nvidia.py*.

C Neural network architectures of experiments

The neural network architecture of Experiment 1 is a fully connected network with a single hidden layer of 32 neurons and an output layer of 10 neurons. The image 4 shows the architecture.

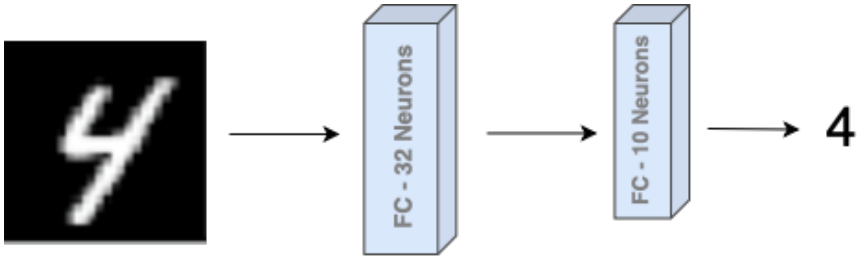

Figure 4: Experiment 1 Network Architecture.

The neural network architecture of Experiment 2 is the DnCNN network presented in [37]. The image 5 shows the architecture proposed in the original paper, which is the one we used in the experiment.

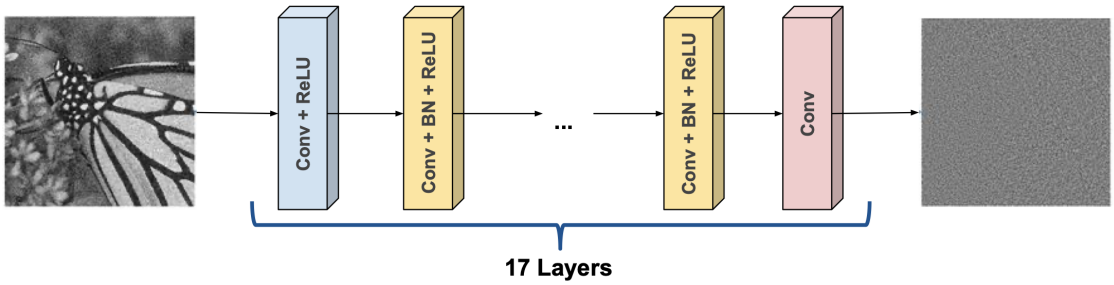

Figure 5: DnCNN Network Architecture. Image taken from [37].