CICO$_2$e: A Compute Carbon Footprint Estimation Tool based on Time Series Data

Christian Plewnia and Horst Lichter

Research Group Software Construction
RWTH Aachen University, Aachen, Germany
{plewnia,lichter}@swc.rwth-aachen.de

Abstract. In recent years, multiple research papers and tools were published that addressed the interest of estimating the carbon footprint of computational activities. Generally, the presented carbon footprint estimation methods calculate the product of the compute hardware's energy consumption and a factor expressing the emissions for the corresponding energy produced in the region where the compute hardware is located. However, there are three open issues. First, the methods for determining the energy consumption are inaccurate or lack an evaluation of the accuracy. Second, most tools use as carbon intensity a static long-term average, e.g., over a year, that the tool authors gathered once, but since some regions have a carbon intensity varying each day and throughout the year, the accuracy of using a static carbon intensity is unclear and requires an evaluation. Third, most tools enable estimates for a single computer or a homogeneous set of computers only, excluding the easy carbon footprint estimation for scenarios with a heterogeneous set of compute hardware.
In this paper, we make three contributions. First, we analyze the evaluation gap regarding methods for determining the compute hardware's power consumption. Second, using example cases, we show that estimating a carbon footprint using a static long-term average carbon intensity compared to hourly carbon intensity time series data can lead to large errors; in the worst case among the example, the error was 325.8%. Third, we present a tool assisting with carbon footprint estimates using time series carbon intensity data and also supporting heterogeneous compute hardware.

Keywords: carbon footprint, energy consumption, green computing

1 Introduction

Carbon emissions are made responsible for the climate change [1], creating a strong interest in understanding and reducing the carbon footprint of many kinds of energy-consuming activities, e.g., industry production processes, energy production, and traveling. Understanding an activity's carbon footprint involves quantifying the greenhouse gas (GHG) emissions attributed to the activity. GHGs include carbon dioxide (CO_2), methane (CH_4), and nitrous oxide

(N_2O). To simplify the quantification, the individual emissions can be expressed as CO_2-equivalent (CO_2e) emissions.

The interest in understanding an activity's carbon footprint reached computer science. Contributions come from the area of machine learning [3–6, 15] and cloud computing [9–12]. The contributions include various tools for estimating the carbon footprint of compute activities [14–20]. Briefly summarized, these tools first measure or estimate the energy consumption of a single computer or compute infrastructure and then estimate the carbon footprint caused by the energy consumption by multiplying the energy consumption with a carbon intensity (CI) for a region chosen by the user. The CI expresses the mass of CO_2e emissions per unit of energy consumption, e.g., as gram CO_2e per kilowatt hour.

There are three issues with the existing tools for estimating the carbon footprint. First, these tools use methods for estimating the energy consumption that are inaccurate or lack an evaluation of the accuracy. We explain this in the related work in more detail.

Second, many tools use per selected region a static CI gathered once by the tools' authors, e.g., an annual average CI. Using a static CI is accurate if the user knows the time interval covered by the average CI and makes an estimate for this time interval. However, users generally do not intend to estimate the carbon footprint for this predefined time interval only. If the real CI rarely deviates from the static CI, this issue could be negligible, but in some regions, the mix of energy production and the CI varies strongly over time; an example is Finland in 2022 as shown in figure 1. Thus, the issue is that the accuracy of the estimated carbon footprint depends on how well the static CI represents the time interval chosen by the user.

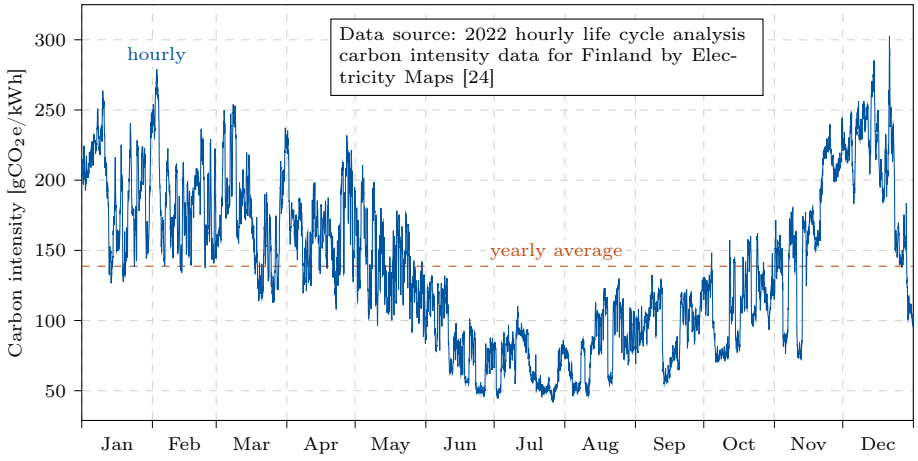


Fig. 1: Finland's energy production carbon intensity in 2022.

Third, most of the tools assume a compute infrastructure consisting of a single server or a homogeneous set of servers. However, there are scenarios like the operation of web applications where the underlying compute infrastructure can be heterogeneous, i.e., it consists of different kinds of hardware with individual power. Thus, the third issue is the currently limited applicability of the estimation tools that excludes some computational activities.

In this paper, we make three contributions. First, in section 2, we present an overview of the existing tools and their carbon footprint estimation methods, we describe the lack of evaluation, and we discuss the data gap that prevents an evaluation. Second, in section 3, we quantify the error of estimating a carbon footprint with an annual average CI for exemplary cases; we compared estimations based on an annual average CI with estimations based on an hourly CI time series for selected regions and time intervals in 2022. Third, in section 4, we present an estimation tool that fills the gap of relying on hourly time series CI data; this tool enables estimates for single computers and a set of different computers, i.e., a heterogeneous compute infrastructure. Besides the contributions, we provide some ideas for future work in section 5 and summarize our findings in section 6.

2 Related work

The interest in determining the carbon footprint of computational tasks is not new. As shown in table 1, there are multiple tools for assisting users in estimating the carbon footprint of a computational activity:

- **Reporting dashboards** Some cloud providers give their customers access to dashboards reporting the customer's past cloud computing usage's carbon footprint [9–11]. Of all the tools discussed here, they are the only ones considering emissions from other activities (e.g., production and waste) besides the data centers' energy consumption. Those emissions are reported using the Greenhouse Gas Protocol scopes as categorization [2]. While this provides a much broader perspective on the emissions, the tools have only a rough documentation or none at all and the exact calculations are not made public.
  An alternative integrating estimates for multiple cloud providers in one dashboard is the open source tool "Cloud Carbon Footprint" (as of September 2023, the supported providers are AWS, Azure, and GCP) [18]. For each cloud provider considered, the tool obtains the user's cloud usage using the provider's API, maps the usage of cloud resources to hardware (considering CPUs, GPUs, memory, storage and network traffic), estimates the hardware's energy consumption, and, finally, estimates the carbon footprint.
- **Instrumentation libraries** There are multiple libraries that allow developers to instrument their code such that the code execution's energy consumption is measured and the carbon footprint is estimated [3, 6, 7, 15].
- **Estimation assistants** The web applications "Green Algorithms" and "ML CO2 Impact" enable their users to estimate a compute activity's carbon

footprint based on the activity's runtime and the used hardware [4, 5]. ML CO2 Impact considers either a CPU or GPU, the computation's runtime and the computation's location, while Green Algorithms also considers memory and allows to consider both types of processors at the same time.

In the remainder of this section, we provide an overview of the methods used by the aforementioned tools since the methods' details are crucial to the estimations' accuracy. For this analysis, we studied the openly available documentations and source codes for all but the cloud providers' tools. The tools have in common that they first determine a compute activity's energy consumption and then multiply the result with a carbon intensity to obtain the footprint. They differ in how they determine energy consumption and what carbon intensity information they use, which we explain subsequently.

2.1 Determination of the energy consumption

The tools compute the energy consumption by summing up the *estimated* or *measured* power of selected hardware components as shown in table 1.

Power measurements for CPUs, GPUs, and memory rely on processor features. The Running Average Power Limit (RAPL) interface initially developed by Intel can provide the CPU's and memory's power consumption. The NVIDIA Management Library (NVML) interface does the same for GPUs by NVIDIA.

Power estimations for CPUs and GPUs are based on either the processor's *Thermal Design Power (TDP)* (Green Algorithms, ML CO2 Impact) or on factors derived from datasets with power measurements (Cloud Carbon Footprint).

Processor power estimations using the TDP assume it to be an approximation of the processor's power. A processor's TDP specifies the thermal energy that the processor may dissipate, which serves as a reference for what the cooling must be able to handle. Since most of a processor's power ends up as heat and processor manufacturers have an interest in sufficient cooling to ensure the processor's stable operation, we conclude that the TDP can be considered as an upper bound to the processor's power.

The processor power estimations by Cloud Carbon Footprint base on factors derived from the SPECpower dataset for CPUs and a dataset with power measurements for GPUs created by the company Teads [13]. Each record in the SPECpower database contains information on the total power consumption of one or more servers when the CPU idles and when it is at max load. From this data, the authors of Cloud Carbon Footprint derived an aggregated idle and max load energy consumption per CPU thread (also referred to as vCPU) for each CPU family, e.g., Intel Coffee Lake and Skylake. With a mapping of cloud services to the combination of CPU families, a number of vCPUs, an assumed load as well as hours of cloud service usage, the CPU energy consumption is estimated. The approach for GPUs is similar.

To estimate the power of memory, storage, and network traffic factors are used. For memory, the factor relies on statements made by memory manufacturers Crucial and Micron [21, 22]. Cloud Carbon Footprint's power estimations

Table 1: Software tools for determining a compute activity's carbon footprint and their methods for determining the energy consumption and carbon intensity

	Tool	**Power** [(m)easure/(e)stimate (method)]					**Carbon intensity**
		CPU	**GPU**	**memory**	**storage**	**network**	
reporting dashboards	AWS Customer Carbon Footprint Tool	unknown	unknown	unknown	unknown	unknown	unknown
	Azure Emissions Impact Dashboard[1]	unknown	unknown	unknown	unknown	unknown	unknown
	Google Cloud Carbon Footprint[1]	unknown	unknown	unknown	unknown	unknown	unknown
	Cloud Carbon Footprint	e (dataset)	e (dataset)	e (factor)	e (factor)	e (factor)	static
instrumentation libraries	carbontracker [3]	m (RAPL)	m (NVML)	m (RAPL)	not considered	not considered	time series
	code carbon[2]	m (RAPL)	m (NVML)	e (factor)	not considered	not considered	static
	Eco2AI [7]	e (TDP)	m (NVML)	e (factor)	not considered	not considered	static
	Experiment Impact Tracker [6]	m (RAPL)	m (NVML)	m (RAPL)	not considered	not considered	static
	Tracarbon[3]	m (RAPL)	m (NVML)	m (RAPL)	not considered	not considered	static
estimation assistants	Green Algorithms [5]	e (TDP)	e (TDP)	e (factor)	not considered	not considered	static
	ML CO2 Impact	e (TDP)	e (TDP)	e (factor)	not considered	not considered	static
	$CICO_2e$ (this paper)	e (TDP)	e (TDP)	e (factor)	not considered	not considered	time series

(1) provides emissions for GHG protocol scopes 1, 2, and 3, i.e., covers more than emissions related to computational energy consumption **(2)** if RAPL is not available, falls back to TDP-based estimate for CPU; if available, uses latest carbon intensity retrieved from API for private compute infrastructures **(3)** row values only apply for local computers running linux; different method used for cloud servers

for storage and network rely on estimated factors [13]. The factor for storage is derived from projections for 2020 published in the 2016 U.S Data Center Usage Report.

A current issue of all power estimation approaches is a missing validation. The TDP-based estimation is likely inaccurate as it does not consider known variances in the power depending on the processor's load, but it provides an upper bound for a processor's power. Although the derivation of the factors used for memory, storage, and network traffic appears plausible, their accuracy is unknown; the authors of Cloud Carbon Footprint describe this transparently in their documentation's methodology section [13]. Thus, currently all estimation tools are experimental. In our opinion, a thorough evaluation of the existing estimation methods cannot be conducted without more comprehensive data on the power consumption of the hardware components.

2.2 Determination of the carbon intensity

All considered tools estimate the carbon footprint using a CI reflecting the energy mix of the server's location, i.e., a CI for the data center or the region thereof. Many tools use per location a static CI value gathered once by the authors. We refer to such a CI that is constant over time as *static*. An exception is the instrumentation library carbontracker [3], which recurrently requests an updated CI during the execution of the instrumented code. This is an example of what we refer to as *time series* CI, i.e., a series of CI values over time.

3 Carbon footprint estimation with static vs. time series carbon intensity

In this section, we study the accuracy of estimating the carbon footprint of a compute activity with a static CI as it is practiced by many tools presented in the related work. We quantify the error of estimating with a static CI by comparing it to estimating with hourly time series CI data, i.e., as a baseline we use the more accurate estimation based on the time series CI data. As figure 1 shows, a region's CI can vary over the time of day and throughout the year, indicating that estimating with a static CI may exhibit strong inaccuracies.

The error cannot be expressed as a single static number but depends on three variables: the CI development (the time series CI data) of the region of the data center we use to run a compute activity as well as the compute activity's starting time and runtime. Thus, our study has to consider these three dimensions when quantifying the error. Conducting a study across the full extent of these dimensions is not feasible and also not necessary to provide first insights into the accuracy of estimating with a static CI. Thus, we limit the considered regions and time to exemplary cases.

3.1 Method

To quantify the error, we start by estimating the carbon footprint for compute activities of varying runtimes using a static CI and time series CI data. As static CI, we do not use the values used by the tools of the related work as this comparison would be convoluted and unfair for the tools: convoluted since not all tools use the same static values and they do not all cover the same regions; unfair since their static CI values are not all based on the same time frames. Instead, we calculate each region's average CI over a common time frame and use this average CI as the region's static CI.

We continue with estimating the carbon footprint for hypothetical compute activities with runtimes from one hour to the whole year, e.g., for 2022 the runtimes are in the interval [1, 8760]. For each runtime, we calculate the carbon footprint for every possible starting hour such that the compute activity completes within the time frame. For example, considering 2022 again, for a 1-day runtime, we calculate the carbon footprint for the job starting on the 1st of January 2022 at 00:00, on the first of January at 01:00 until the latest possible starting point on the 31st of December 2022 at 00:00; for a 1 year runtime, there is only one possible start time to stay within 2022, which is January 1st at 00:00. Similar to the tools of the related work, we assume the compute activity causes a static power consumption. We calculate with a value of 1 kilowatt, but the value is not important as we are only interested in how the carbon footprints estimated with the static CI and the time series CI data relate to each other.

Then, we determine how much the carbon footprint estimated with the static CI deviates from the carbon footprint estimated with the time series CI data, i.e., we determine the error in the estimation using the static CI. For the comparison, we look at the *relative error*, i.e., for a geographical region g, a starting time s, and a runtime r, the relative error is the difference in the carbon footprint estimated using the static CI and the time series CI data in relation to the carbon footprint estimated using the time series CI data:

$$relError\,(g, s, r) = \frac{footprint_{static}(g, s, r) - footprint_{ts}\,(g, s, r)}{footprint_{ts}\,(g, s, r)}$$

For the error analysis, we look at maxima as well as average values. In the first case, we are interested in finding out what error to expect in the worst case. For this, we proceed as follows: let S_r be the set of all starting times such that a job with a runtime t completes within the considered time frame; then, for every region and runtime, we search across the starting times S_r for where the absolute value of the relative error is the highest:

$$maxError\,(g, r) = \max_{s \in S_r} abs\,(relError\,(g, s, r))$$

In the second case, we are interested in finding out what error to expect when we would start a job at a randomly selected time within the considered time frame. To achieve this, we calculate for every region and runtime the average of the relative errors' absolute values across all starting times:

$$avgError(g,r) = \frac{\sum_{s \in S_r} abs(relError(g,s,r))}{|S_r|}$$

3.2 Dataset (regions & time frame)

We applied the method to CI data provided by the company Electricity Maps [24]. It gathers CI data for many regions in the world from various sources and provides unified datasets. In June 2023, Electricity Maps published historic datasets for 2021 and 2022 under an open-source license. The datasets contain CI data for the direct emissions of the energy production as well as life cycle analysis (LCA) emission data that includes not only direct emissions from the energy production but also indirect emissions, e.g., "emissions from the extraction of resources required to build up installed capacity, emissions from direct operations, and end-of-life related emissions" [23]. The datasets come in multiple resolutions, with the most fine granular being hourly CI data.

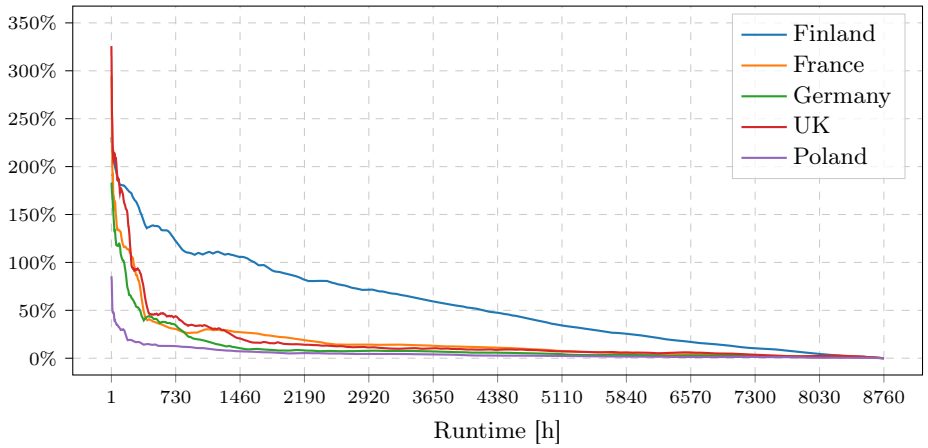
In this study, we used as time series CI data the hourly LCA CI data by Electricity Maps for 2022 for the regions Finland, France, Germany, the UK, and Poland. We found those regions to be interesting as they exhibited different mixes in their energy production. However, the choice of regions is not crucial as we only intend to demonstrate a few exemplary cases of quantifying the error.

3.3 Results

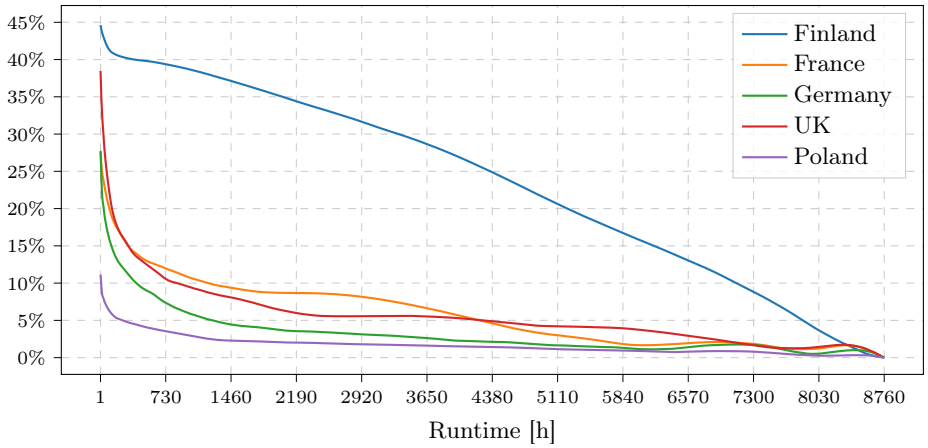
In the presentation of the results, we focus on runtimes of up to 30 days as this covers some of the largest batch jobs reported in machine learning in recent years, e.g., Google's Meena (30 days), T5 (20 days), Switch Transformer (27 days), and OpenAI's GPT-3 (about 15 days) [8]. Common compute activities are nightly batch jobs or build jobs, which typically run no more than half a day. Therefore, we decided to select runtimes of the extreme values (1 hour and 8760 hours), as well as a few representative runtimes of nightly jobs (3, 6, and 12 hours) and long-running jobs (15 and 30 days).

Figure 2a and table 2a depict the maximum errors when estimating with the region's annual CI instead of the region's hourly CI. For example, for a compute activity running in a data center in France for 15 days, the error in the carbon footprint estimation will be at most 46.3%. We find a few aspects of the results noteworthy:

1. With longer runtimes, the maximum errors tend to get smaller, although they are not monotonically falling.
2. The runtimes representative for nightly and machine learning jobs are within the lower range of runtimes where the maximum error is rather high.
3. The maximum errors are always the highest for compute activities running for 1 hour. This coincides with extremes in the hourly CI data.
4. The maximum errors are always the lowest for compute activities running for 8760 hours, i.e., jobs running for the whole year 2022. The error is always



(a) Maximum errors



(b) Average errors

Fig. 2: Carbon footprint estimation errors using an annual CI relative to the estimation using hourly CIs for runtimes between 1 hour and 1 year (8760 hours) and selected regions in the year 2022.

Table 2: Errors in estimating a compute activity's carbon footprint using an annual CI relative to estimating the footprint using hourly CIs for a few representative compute activity runtimes for selected regions in the year 2022.

(a) Maximum errors

	Maximum error [%]				
Runtime [h]	**Finland**	**France**	**Germany**	**UK**	**Poland**
8760 (= 1 year)	0.0	0.0	0.0	0.0	0.0
720 (= 30 days)	123.5	30.7	35.1	43.3	12.7
360 (= 15 days)	145.0	52.9	41.3	77.2	14.4
12	221.1	201.5	169.3	243.2	52.2
6	225.4	267.3	175.3	276.2	76.7
3	228.2	293.2	181.2	313.5	83.8
1	231.4	294.7	183.3	325.8	85.6

(b) Average errors

	Average error [%]				
Runtime [h]	**Finland**	**France**	**Germany**	**UK**	**Poland**
8760 (= 1 year)	0.0	0.0	0.0	0.0	0.0
720 (= 30 days)	39.4	12.0	7.4	10.6	3.6
360 (= 15 days)	40.0	14.3	10.5	14.1	4.6
12	43.9	25.7	23.8	34.4	9.3
6	44.3	26.7	26.2	36.6	10.4
3	44.5	27.3	27.3	37.9	10.9
1	44.6	27.6	27.8	38.5	11.1

0 for 8760 hours since the job's time frame is the same as the time frame over which the annual CI was calculated.

One aspect not shown by the here presented results is the direction of the deviation. In some cases, the relative error identified as the maximum error is negative, meaning the carbon footprint estimated with the annual CI is lower than the one estimated with the hourly CI.

Figure 2b and table 2b depict the average errors when estimating with the region's annual CI instead of the region's hourly CI. Due to the average applied, the errors shown here are lower than the maximum errors shown before. Similarly, as for the maximum errors, the average errors are the highest for a runtime of 1 hour and, with an error of 0%, the lowest for a runtime of 8760 hours.

The materials used to obtain the presented analysis results are provided in the paper's supplemental material. This covers the input datasets by Electricity Maps, the source code for computing the errors, and the resulting datasets with the maximum and average errors for all runtimes across the selected regions.

3.4 Discussion

The study allows us to draw three conclusions regarding the error of estimating a compute activity's carbon footprint with an annual CI instead of an hourly CI for the considered time frame of 2022 and the regions Finland, France, Germany, the UK, and Poland:

1. There can be large errors in the carbon footprint estimation. For instance, in the worst case, we would have had a large error of 325.8% (maximum error for the UK and a 1-hour runtime).
2. On average, the expected errors are much lower than the maximum error. For example, the average error for a 1-hour compute activity running in the UK is 38.5%.
3. The errors highly depend on the considered compute activity, i.e., the region where the compute activity is executed as well as the compute activity's starting time and runtime. For example, for compute activities running in the considered regions for 1 hour or 720 hours, the maximum error ranges from 12.7% to 325.8% and the average error ranges from 3.6% to 44.6%.

The above conclusions are specific to the year and regions and thus may not be applicable to other years and regions. In general, the study revealed that there is a risk of large errors when estimating a compute activity's carbon footprint with a static CI instead of time series CI data. Further, in general, we have no knowledge about the extent of the errors when using a static CI until we compare the results to estimations with time series CI data.

The aforementioned problems lead to the question of whether to always prefer the carbon footprint estimation with time series CI data. The tool we present in the next section makes it possible to conduct carbon footprint estimations using time series CI data with little additional effort compared to the tools

presented in the related work. The only additional input that a user must provide is information about the compute activity's starting time. The tool takes care of the estimations as long as time series CI data is available – an effort that can be taken care of by the tool developers. Thus, the only case where a simplified estimation with a static CI is preferable is when an estimation with time series CI data is impossible due to missing data.

3.5 Threats to validity

This study can only consider a certain time frame. Future changes in the energy mix, e.g., caused by building new renewable energy capacity, and varying weather conditions can affect a region's CI. Thus, the errors obtained by this study are not valid beyond the considered time frame. However, we think it is unlikely that many regions will have a (close to) constant CI in the foreseeable future, which is why we think the study's idea will remain relevant beyond the considered time frame.

4 Compute carbon footprint estimation tool ($CICO_2e$)

The Compute Infrastructure CO_2e estimator ($CICO_2e$) is a web application enabling its users to estimate the energy consumption and carbon footprint for a configurable heterogeneous compute infrastructure using CI time series data. The user can configure the composition of the compute infrastructure, which is used to determine the energy consumption, and choose what CI time series data to use, which is used together with the energy consumption to determine the carbon footprint. Subsequently, we describe the tool's usage.

To obtain an estimate, the user must first configure a compute infrastructure. The user can add data centers to the infrastructure and servers to each data center. An example of an infrastructure with one data center and two types of servers is shown in figure 3. Each server is defined by how much energy it consumes and a usage plan specifying one or more rules for how many instances of that server are running at a time, e.g., 1 server instance running in November 2022 or a weekly recurring pattern of 6 instances running on weekdays during 2022. Figure 4 shows an example of a usage plan and its rules configuration. The flexible configuration of the usage rules allows for modeling different kinds of compute activities, e.g., one-time scientific jobs, constantly running machine learning clusters and flexibly scaling web application clusters. To assist the user in estimating the energy consumption of a server, an estimate can be calculated based on a specification of the CPU, the GPU and the memory. The energy consumption is estimated based on the TDP (for the CPU and GPU) and on a factor (for the memory).

The servers only define the direct energy consumption of the compute hardware. Indirect energy consumption by supplemental hardware like cooling and network devices is covered by the Power Usage Effectiveness (PUE) factor, which

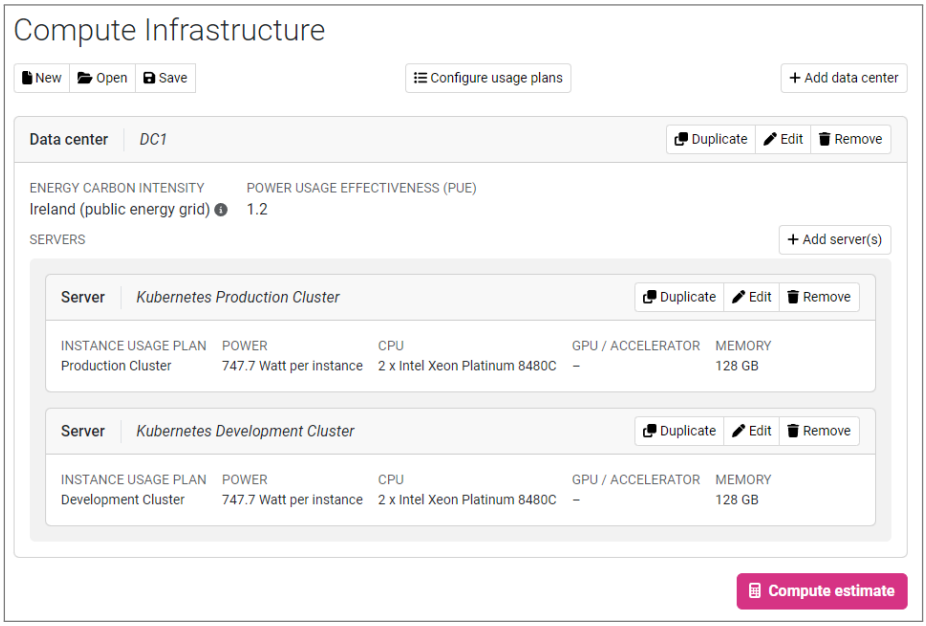


Fig. 3: Screenshot of the compute infrastructure composition in $CICO_2e$

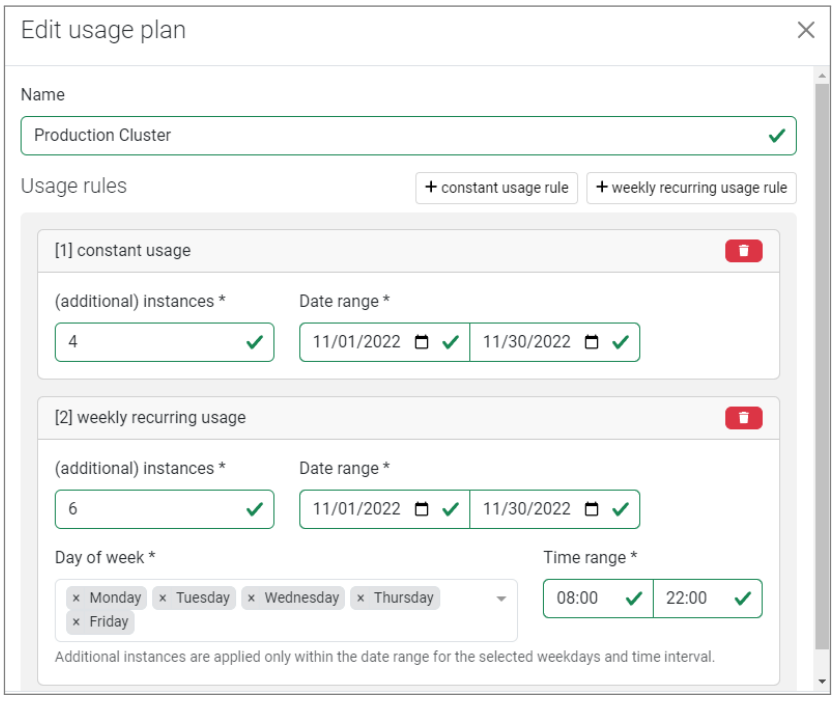


Fig. 4: Screenshot of the usage plan configuration in $CICO_2e$

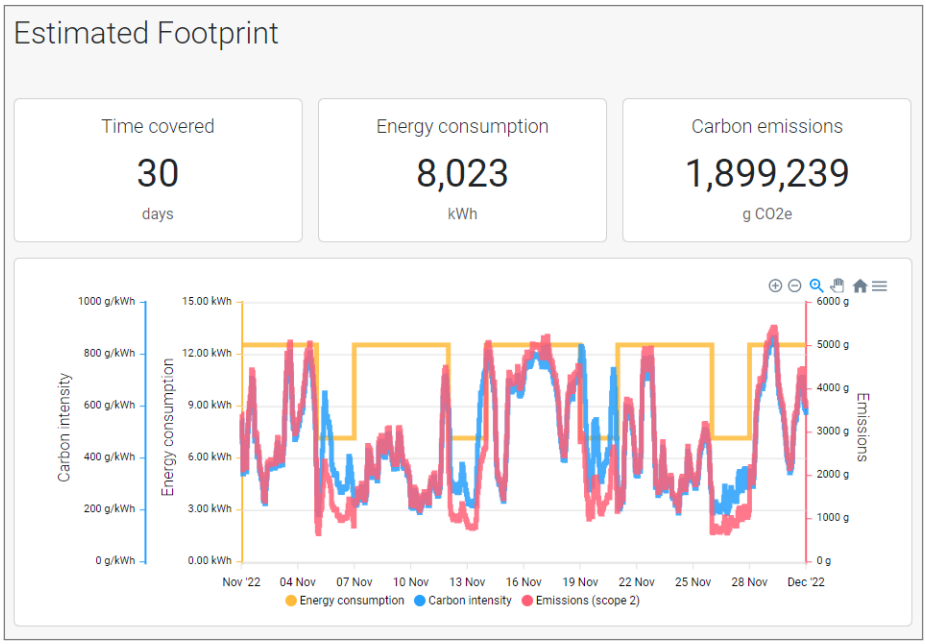


Fig. 5: Screenshot of the estimated carbon footprint in $CICO_2e$

a user can configure for the data center. It is up to the user to get the PUE from the data center's provider.

For the data center, the user must also configure the CI data matching the data center's energy mix. We added hourly CI datasets covering 12 regions for the years 2021 and 2022, which were released by Electricity Maps [24] using the Open Database License. In the example in figure 3, the data center uses the CI data for Ireland.

Once the compute infrastructure is configured, the user can request the carbon footprint estimate. It is computed by a server application in the background. The estimations use the processors' TDP and the factor for the memory power consumption that is also used by other tools. Since we use the TDP for the processor power estimation, the result is not the real consumption but an upper bound. Using the result from the server, the user interface shows the estimated total energy consumption, the estimated total carbon footprint of the energy consumed (GHG protocol scope 2 emissions), and a graph depicting a development of power consumption and carbon footprint over the time of the compute infrastructure usage. For example, figure 5 shows an estimate for the previously configured compute infrastructure.

The application was designed such that CI datasets can be added by adapting the backend's configuration. This allows developers to integrate updated hourly CI data by Electricity Maps or other sources, including the latest Electricity Maps data that is available for a fee.

5 Future work

Future work should focus on the evaluation of compute hardware energy estimation methods, which requires addressing the data gap. For processors and memory, we imagine that a benchmark similar to the SPECpower benchmark could be developed, which does not rely on a complex setup for measuring the AC input, but uses the RAPL and NVML interfaces discussed in the related work; however, this goes with the assumption that these interfaces provide reliable power measurements.

Once the evaluation is done, the next step is to look into methods for optimizing the carbon footprint of compute activities based on estimations; we found some work going in this direction, but we think this will not be fruitful until we have a reliable estimation method.

While in our experience many consumers of energy pay a fixed price per kilowatt hour, we can imagine that variable pricing can become more common in the future. The more the energy production becomes renewable, the more the availability of energy may vary – and with it the price. In such a case, being able to estimate energy costs using time series pricing data could become interesting, which could be achieved by replacing the CI time series data with pricing time series data in our tool. Similarly, paying compensation for carbon emissions could become more common in the future. If this happens, being able to estimate and minimize the compensation expenses could become an interesting topic as well.

We have also plans for the future of $CICO_2e$. Electricity Maps intends to provide free access to their carbon intensity datasets for 2023 at some point; as soon as they are available we want to integrate them into our tool. We also consider adding assistance for users in composing cloud compute infrastructures since hardware specifications of cloud servers are usually not easy to find; the Cloud Carbon Footprint project already gathered information that could be used to realize this assistance.

6 Conclusion

In this paper, we addressed three issues of carbon footprint estimation tools. First, in section 2 we show that many tools' methods or estimating the energy consumption of a computer lack an evaluation. Thus, for many tools the estimate's accuracy is unknown. An exception are tools relying on the processor's TDP: while the TDP does not correspond to the energy consumption, it provides an upper bound to it. The general issue with developing a reliable, more accurate method is the data gap with respect to the power consumption of compute hardware. Closing this gap is a challenge left for future work.

The second issue we addressed is that most tools use a static carbon intensity (CI) to estimate the carbon footprint. We showed in section 3 that the use of a static CI can lead to inaccurate carbon footprint estimates. Among the exemplary cases we analyzed, the error of calculating the footprint with a static CI was up to 325.8% compared to calculating with an hourly CI time series.

The highest error occur for the compute jobs with short runtimes and the error tends to decrease with longer compute job runtimes. To overcome these errors, we recommend carbon footprint estimation tools to rely on time series CI data instead of a static CI if possible or to clearly document what CI is used and what it means for the estimate's accuracy.

Third, there was a gap not covered by existing tools. They could create a carbon footprint estimate for either a past cloud infrastructure that could be composed of a heterogeneous set of servers or for a model of a past or planned infrastructure that, however, could consist of a single computer or a homogeneous set of servers only. $CICO_2e$ is an addition to the existing tools that fills this gap as it can estimate the carbon footprint for a model of a past and planned heterogeneous compute infrastructure alike. Following our own recommendation, $CICO_2e$ estimates carbon footprints based on time series CI data. We intend to continue the development of $CICO_2e$ (e.g, by supporting easier modelling of compute infrastructures composed of public cloud servers) and welcome contributions.

Supplemental material

- **Analysis**
  The datasets and the source code of the analysis in section 3 can be found in the paper's supplemental material published at GitHub:
  https://github.com/swc-rwth/CICO2e-paper-supplemental-material
- **$CICO_2e$ application**
  The $CICO_2e$ web application is online accessible at:
  https://cico2e.swc.rwth-aachen.de
- **$CICO_2e$ source code**
  The source code of $CICO_2e$ is published at GitHub:
  https://github.com/swc-rwth/CICO2e

References

1. Intergovernmental Panel on Climate Change (IPCC) (Ed.): Climate Change 2022 - Mitigation of Climate Change: Working Group III Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge: Cambridge University Press (2023). doi:10.1017/9781009157926
2. The Greenhouse Gas Protocol: A Corporate Accounting and Reporting Standard, revised ed. ed. o.O.: World Resources Institute and World Business Council for Sustainable Development (2004). Online: https://ghgprotocol.org/corporate-standard; last accessed: 2023-11-24
3. Anthony, L.F.W., Kanding, B., Selvan, R.: Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. 2020. doi:10.48550/arXiv.2007.03051
4. Lacoste, A., Luccioni, A., Schmidt, V., et al.: Quantifying the carbon emissions of machine learning. 2019. doi:10.48550/arXiv.1910.09700

5. Lannelongue, L., Grealey, J., Inouye, M.: Green algorithms: Quantifying the carbon footprint of computation. Advanced Science, vol. 8, no. 12, pp. 1–10, (2021). doi:10.1002/advs.202100707
6. Henderson, P., Hu, J., Romoff, J., et al.: Towards the systematic reporting of the energy and carbon footprints of machine learning. 2022. doi:10.48550/arXiv.2002.05651
7. Budennyy, S.A., Lazarev, V.D., Zakharenko, N.N. et al.: eco2AI: Carbon Emissions Tracking of Machine Learning Models as the First Step Towards Sustainable AI. Dokl. Math. 106 (Suppl 1), S118–S128 (2022). doi:10.1134/S1064562422060230
8. Patterson, D., Gonzalez, J., Le, Q., et al.: Carbon emissions and large neural network training. 2021. doi:10.48550/arXiv.2104.10350
9. Amazon: Amazon Web Services Customer Carbon Footprint Tool. Online: https://aws.amazon.com/aws-cost-management/aws-customer-carbon-footprint-tool/; last accessed: 2023-11-24
10. Microsoft: Azure Customer Carbon Footprint Tool. Online: https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard; last accessed: 2023-11-24
11. Google: Google Cloud Platform Customer Carbon Footprint Tool. Online: https://cloud.google.com/carbon-footprint; last accessed: 2023-11-24
12. Thoughtworks, Inc.: Cloud Carbon Footprint. Online: https://www.cloudcarbonfootprint.org/; last accessed: 2023-11-24
13. Thoughtworks, Inc.: Cloud Carbon Footprint – Methodology documentation. Online: https://www.cloudcarbonfootprint.org/docs/methodology/; last accessed: 2023-11-17
14. Carbontracker. Software source code repository. Online: https://github.com/lfwa/carbontracker; last accessed 2023-11-24
15. CodeCarbon. Software source code repository. Online: https://github.com/mlco2/codecarbon/; last accessed 2023-11-24
16. Eco2AI. Software source code repository. Online: https://github.com/sb-ai-lab/Eco2AI; last accessed 2023-11-24
17. Experiment Impact Tracker. Software source code repository. Online: https://github.com/Breakend/experiment-impact-tracker; last accessed 2023-11-24
18. Green Algorithms. Software source code repository; git commit: ef4d0a1. Online: https://github.com/GreenAlgorithms/green-algorithms-tool; last accessed 2023-11-24
19. ML CO2 Impact. Software source code repository. Online: https://github.com/mlco2/impact; last accessed 2023-11-24
20. Tracarbon. Software source code repository. Online: https://github.com/fvaleye/tracarbon; last accessed 2023-11-24
21. Crucial: How Much Power Does Memory Use? Online: https://www.crucial.com/support/articles-faq-memory/how-much-power-does-memory-use; last accessed: 2023-11-17
22. Micron: Calculating Memory Power for DDR4 SDRAM. Technical note, revision B 8/18 EN (2017). Online: https://www.micron.com/-/media/client/global/documents/products/technical-note/dram/tn4007_ddr4_power_calculation.pdf last accessed: 2023-11-24
23. Electricity Maps ApS: Electricity Maps Methodology. Online: https://www.electricitymaps.com/methodology; last accessed: 2023-11-24
24. Electricity Maps ApS: Electricity Maps Data Portal. Online: https://www.electricitymaps.com/data-portal; last accessed: 2023-11-24