ISSN 1064-5624, Doklady Mathematics, 2022, Vol. 106, Suppl. 1, pp. S118–S128. © The Author(s), 2022. This article is an open access publication.
ISSN 1064-5624, Doklady Mathematics, 2022. © The Author(s), 2022. This article is an open access publication.
Russian Text © The Author(s), 2022, published in Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya, 2022, Vol. 508, pp. 134–145.

ADVANCED STUDIES IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

eco2AI: Carbon Emissions Tracking of Machine Learning Models as the First Step Towards Sustainable AI

S. A. Budennyy[a,b,*], V. D. Lazarev[b], N. N. Zakharenko[a], A. N. Korovin[b], O. A. Plosskaya[a], D. V. Dimitrov[a], V. S. Akhripkin[a], I. V. Pavlov[a], I. V. Oseledets[b,c], I. S. Barsola[d], I. V. Egorov[d], A. A. Kosterina[d], and L. E. Zhukov[e]

Presented by Academician of the RAS A.P. Kuleshov

Received October 28, 2022; revised October 28, 2022; accepted November 1, 2022

Abstract—The size and complexity of deep neural networks used in AI applications continue to grow exponentially, significantly increasing energy consumption for training and inference by these models. We introduce an open-source package *eco2AI* to help data scientists and researchers to track the energy consumption and equivalent CO_2 emissions of their models in a straightforward way. In *eco2AI* we focus on accurate tracking of energy consumption and regional CO_2 emissions accounting. We encourage the research for community to search for new optimal Artificial Intelligence (AI) architectures with lower computational cost. The motivation also comes from the concept of AI-based greenhouse gases sequestrating cycle with both Sustainable AI and Green AI pathways. The code and documentation are hosted on Github under the Apache 2.0 license https://github.com/sb-ai-lab/Eco2AI.

Keywords: ESG, AI, sustainability, carbon footprint, ecology, CO_2 emissions, GHG

DOI: 10.1134/S1064562422060230

1. INTRODUCTION

While the global ESG agenda (Environment, Social, and Corporate Governance) is guided by agreements established between countries [1], the actual development of ESG principles occurs through corporate, research, and academic standards. Many companies have started to develop their ESG strategies, allocating full-fledged functions and departments dedicated to the agenda, publishing annual reports on sustainable development, providing additional funds for research, including digital technologies and AI.

Despite growing influence of the ESG agenda, the problem of transparent and objective quantitative evaluation of ESG progress in the field of environmental protection remains. This is of significant importance for IT industry, since about one percent of the world's electricity is consumed by cloud computing, and its share continues to grow [2]. Artificial intelligence (AI) and machine learning (ML) being a big part of the today's IT industry are rapidly evolving technologies with massive potential for disruption. There are a number of ways in which AI and ML could mitigate environmental problems and human-induced impact. In particular, they could be used to generate and process large-scale interconnected data to learn Earth more sensitively, to predict the environmental behavior in various scenarios [3]. This could improve our understanding of environmental processes and help us to make more informed decisions. There is also a potential for AI and ML to be used for simulating the results of harmful activities, such as deforestation, soil erosion, flooding, increased greenhouse gases in the atmosphere, etc. Ultimately, these technologies hold great potential to improve our understanding and control of the environment.

A number of AI-based solutions are being developed to achieve carbon neutrality as the part of the concept of Green AI. The final goal of these solutions is the reduction of greenhouse gases (GHG) emissions. In fact, AI can help to reduce the effects of the climate crisis, for example, by smart grid design, developing low-emission infrastructure and modelling climate changes [8]. However, it is also crucial to account for generated CO_2 emissions by AI itself as a

[a] *Sber AI Lab, Moscow, Russia*
[b] *Artificial Intelligence Research Institute, Moscow, Russia*
[c] *Skolkovo Institute of Science and Technology, Moscow, Russia*
[d] *Sber ESG, Moscow, Russia*
[e] *National Research University Higher School of Economics, Moscow, Russia*
**e-mail: sanbudenny@sberbank.ru*

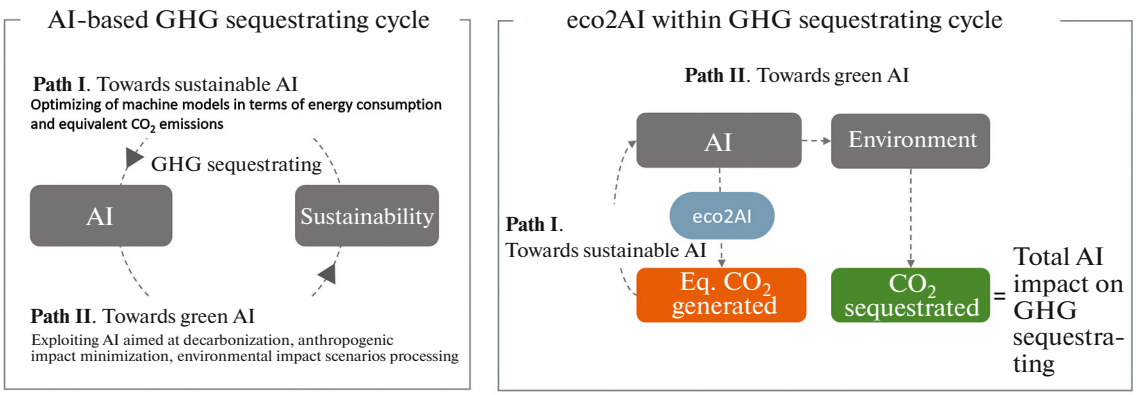

Fig. 1. High-level schemes of AI-based GHG sequestrating. The left scheme corresponds to AI-based GHG sequestrating cycle. The right scheme describes the role of *eco2AI* in this scheme.

result of learning and applying AI models. In fact, AI progresses to larger and larger models with increasing computational complexity and, thereby, electrical energy consumption and, as a result, equivalent carbon emissions (eq. CO_2). The ecological impact of AI is a major factor that needs to be accounted for in the eventual risks. For AI/ML models to be environmentally sustainable, they should be optimized not only for prediction accuracy, but also in terms of energy consumption and environmental impact. Therefore, tracking the ecological impact of AI is the first step towards Sustainable AI. A clear understanding of the ecological impact from AI motivates data science community to search for optimal architectures consuming less computational resources. An explicit call to promote research on more computationally efficient algorithms has been mentioned elsewhere [41].

To summarize the previous theses, we present the concept of AI-based GHG sequestrating cycle, which describes the relationship of AI with sustainability goals (Fig. 1). The request from Sustainability towards AI spawns demand for more optimized models in terms of energy consumption, shaping the path we have called "Towards Sustainable AI." On the other hand, AI creates additional opportunities to achieve sustainability goals, and we suggest to call this path "Towards Green AI." To understand the role of the *eco2AI* library in this cycle, the right panel of Fig. 1 provides a scheme with the paths mentioned. First, *eco2AI* motivates to optimize the AI technology itself. Secondly, if AI is aimed to sequestrate the GHG, then the total effect should be evaluated with account for generated eq. CO_2 at least during training sessions (and during model application/inference at best). In this article we only focus on examining the path "Towards Sustainable AI" (see examples in Section 4).

Contribution. The contribution of our paper is threefold:

- First, we introduce *eco2AI*, an open-source Python library we developed for evaluating equivalent CO_2 emissions during ML model training.
- Second, we define the role of *eco2AI* within the context of AI-based GHG sequestrating cycle concept.
- Third, we describe practical cases where *eco2AI* has been used as an efficiency optimization tracker for fusion models learning.

This paper is organized as follows. In Section 2 we review the existing solutions for CO_2 assessment and describe the differences from our library. Section 3 presents the calculation methodology, and Section 4 shows cases of using the library. Finally, in Section 5 we summarize our work. The appendix section briefly describes the code usage.

2. RELATED WORK

In this section, we describe recent practices of CO_2 emissions evaluation for AI-based models. In what follows, we give a brief description of the existing open-source packages, providing comparison summary.

2.1. Practice of AI Equivalent Carbon Emissions Tracking

Since the appearance of DL models, their complexity has been increasing exponentially, doubling the number of learnable parameters every 3–4 months since 2012 (AI) and reaching more than a trillion parameters in 2022. Among the most well known models are BERT-Large (Oct. 2018, 3.4×10^8), GPT-2 (2019, 1.5×10^9), T5 (Oct. 2019, 1.1×10^{10}), GPT-3 (2020, 1.75×10^{11}), Megatron Turing (2022, 5.30×10^{11}), and Switch Transformer (2022, 1.6×10^{12}).

Data accumulation, labeling, storage, processing, and exploitation consumes a lot of resources during

model lifespan from production to disposal. The impact of such models is presented in descriptive visual map on a global scale using Amazon's infrastructure as in [24]. Carbon emissions are only one of footprints of such an industry but their efficient monitoring is important for passing new regulation standards and laws as well as self-regulation [20].

Large-scale research was conducted in [41] focused on quantifying the approximate environmental costs of DL widely used for NLP tasks. Among examined DL architectures, there were Transformer, ELMo, BERT, NAS, and GPT-2. Total power consumption was evaluated as the combined GPU, CPU, and DRAM consumption, multiplied by the data center specific power usage effectiveness (PUE) with a default value of 1. Sampling of CPU and GPU consumption was being queried by the vendor specialized software interface packages: Intel Running Average Power Limit and NVIDIA System Management, respectively. The conversion of energy to carbon emissions was computed as a product of total energy consumption and carbon energy intensity. The authors estimated that the carbon footprint for training BERT (base) was about 652 kg, which is comparable to the carbon footprint of the "New York ↔ San Francisco" air travel per passenger.

The energy consumption and carbon footprint were estimated with the following NLP models: T5, Meena, GShard, Switch Transformer, and GPT-3 [30]. The key outcome resulted in opportunities to improve energy efficiency while training neural network models with methods, such as sparsely activating DL; distillation techniques [22]; pruning, quantization, efficient coding [19]; fine-tuning and transfer-learning [9]; training of large models in a specific region with low energy consumption; and the use of energy-optimized cloud data centers. The authors expect that the carbon footprint would be reduced by 10^2–10^3 times if the mentioned suggestions are taken into account.

2.2. Review of Open-Source Emission Trackers

Several libraries have been developed to track the AI equivalent carbon footprint. Here we are focusing on describing the most popular open-source libraries. They all have a common goal: to monitor CO_2 emissions during model training (see Table 1).

Cloud Carbon Footprint [5] is an application that estimates the energy and carbon emissions of public cloud providers. It measures cloud carbon and is intended to connect with various cloud service providers. It provides estimates for both energy consumption and carbon emissions for all types of cloud usage, including embodied emissions from production, with the option to drill down emissions by cloud provider, account, service, and time period. It provides recommendations to AWS and Google Cloud on saving money and minimize carbon emissions, as well as forecasts cost savings and actual outcomes in the term of trees planted. For hyperscale data centers, it measures the consumption at the service level using actual server utilization rather than an average. It provides a number of approaches for incorporating energy and carbon indicators into existing consumption and billing data sets, data pipelines, and monitoring systems.

CodeCarbon [6] is a Python package for tracking the carbon emissions produced by various computer programs, from simple algorithms to deep neural networks. By taking into account computing infrastructure, location, usage and running time, CodeCarbon provides an estimate of how much CO_2 was has been produced. It also provides comparisons with emissions from common transportation types.

Carbontracker [4] is a tool to track and predict the energy consumption and carbon footprint of training DL models. The package allows for a further proactive and intervention-driven approach for reducing carbon emissions using predictions. Model training can be stopped when the predicted environmental cost exceeds a rational threshold. The library support a variety of different environments and platforms such as clusters, desktop computers, Google Colab notebooks, allowing a plug-and-play experience [2].

Experiment impact tracker [16] is a framework that provides information of the energy, computational and carbon impacts of ML models. It includes the following features: extraction the CPU and GPU hardware information, setting the experiment start and end-times, accounting for the energy grid region where the experiment is being run (based on the IP address), the average carbon intensity in the energy grid region, memory usage, the real-time CPU frequency (in hertz) [20].

Green Algorithms [17] are an online tool that enables the user to estimate and report the carbon footprint of computation. It integrates with computational processes and does not interfere with existing code, while also accounting for a range of CPUs, GPUs, cloud computing, local servers, and desktop computers [26].

Tracarbon [42] is a Python library that tracks energy consumption of the device and calculates carbon emissions. It detects the location and the device model automatically and can be used as a command line interface (CLI) with predefined or calculated with the API (application programming interface) user metrics.

Similar to above-described libraries, in *eco2AI* we focus on the following: taking into account only those system processes that are related directly to models training (to avoid overestimation). We also use extensive database of regional emission coefficients (365 territorial objects are included) and information on CPU devices (3279 models).

Table 1. Features of open-source trackers for equivalent CO_2 emission evaluation of machine learning models

Library	Cloud Carbon Footprint	Code Carbon	Carbon Tracker	Experimental Impact Tracker	Tracarbon	Green Algorithms	eco2AI
General information							
Launch date	2020	2020	2020	2019	2022	2021	2022
License type	Apache 2.0	MIT	MIT	MIT	Apache 2.0	CC-BY-4.0	Apache 2.0
Carbon intensity	✓	✓	–	✓	✓	✓	✓*
OS compatibility							
Linux	✓	✓	✓	✓		✓	✓
Windows	✓	✓	✓			✓	✓
MacOS	✓	✓	✓	✓	✓	✓	✓
Hardware compatibility							
RAM	✓	✓	✓	✓	✓	✓	✓
CPU	✓	✓	Undefined	✓	✓	✓	✓**
GPU	✓	✓	✓	✓	✓	✓	✓
Supplementary							
Data encryption***							✓
WEB interface	✓	✓				✓	

*Account for 365 territorial objects including regional data for Australia (Emission factors sources; Science and Resources, Canada (Emission factors sources; UNFCCC), Russia (Rosstat; EMISS), and USA (Emission factors sources; USA EPA)).

***eco2AI* database includes data on 3279 models of CPU for Intel and AMD.

***Beneficial in scenarios where the authenticity of results is required.

3. METHODOLOGY

This section covers our approach to calculate electric energy consumption, extracting the emission intensity coefficient and conversion to equivalent CO_2 emissions. Each part is described below.

3.1. Electric Energy Consumption

The energy consumption of a system can be measured in joules (J) or kilowatt-hours (kW h)—a unit of energy equal to one kilowatt of power sustained for one hour. The problem is to evaluate energy contribution for each hardware unit [20]. We focus on GPU, CPU, and RAM energy evaluation for their direct and most significant impact on the ML processes. While examining CPU and GPU energy consumption we do not track the terminating processes effect, as its impact on overall power consumption is relatively small. We also do not account for data storage (SSD, HDD) energy consumption, since there not directly running processes.

GPU. The *eco2AI* library is able to detect NVIDIA devices. A Python interface for GPU management and monitoring functions has been implemented within the *Pynvml* library. This is a wrapper for the NVIDIA Management Library that detects most NVIDIA GPU devices and tracks the number of active devices, names, memory used, temperatures, power limits, and power consumption of every detected device. Correct functionality of the library requires CUDA installation on the computing machine. The total energy consumption of all active GPU devices E_{GPU} (k Wh) equals to product of the power consumption of GPU device and its loading time: $E_{\mathrm{GPU}} = \int_0^T P_{\mathrm{GPU}}(t)dt$, where P_{GPU} is the total power consumption of all GPU devices determined by *Pynvml* (kW) and T is GPU devices loading time (h). If the tracker does not detect any GPU device, then GPU power consumption is set to zero.

CPU. The Python modules *os* and *psutil* were used to monitor CPU energy consumption. To avoid overestimation, *eco2AI* takes into account only running processes related to model training. The tracker takes the percentage of CPU utilization and divides it by the number of CPU cores, obtaining CPU utilization percent. We have collected the most comprehensive database containing 3279 unique processors for Intel and AMD models. Each CPU model has information on thermal design power (TDP) which is equivalent to the power consumption at long-term loadings. The total energy consumption of all active CPU devices E_{CPU} (k Wh) is calculated as the product of the power consumption of all CPU devices and its loading time

Table 2. Emission intensity coefficients for selected regions

Country	ISO-Alpha-2 code	ISO-Alpha-3 code	UN M49 code	Emission coefficient, kg/(MW h)
Canada	CA	CAN	124	120.49
France	FR	FRA	250	67.53
India	IN	IND	356	625.57
Paraguay	PY	PRY	600	23.92
Zambia	ZM	ZMB	894	120.78

$E_{\text{CPU}} = TDP\int_0^T W_{\text{CPU}}(t)dt$, where TDP is the equivalent CPU model specific power consumption at long-term loading (kW) and W_{CPU} is the total loading of all processors (fraction). If the tracker can not match any CPU device, the CPU power consumption is set to a constant value of 100 W [27].

RAM. Dynamic random access memory devices are an important source of energy consumption in modern computing systems, especially when a significant amount of data have to be allocated or processed. However, accounting for RAM energy consumption is problematic as its power consumption is strongly depends on operation type, such as data being read, written or maintained. In *eco2AI* RAM power consumption is considered proportional to the amount of allocated power by the currently running process and is calculated as follows: $E_{\text{RAM}} = 0.375\int_0^T M_{\text{RAM}_i}(t)dt$, where E_{RAM} is the power consumption of all allocated RAM (k Wh), M_{RAM_i} is allocated memory (GB) measured via *psutil*, and 0.375 W/Gb is the estimated specific energy consumption of the DDR3 and DDR4 modules [27].

3.2. Emission Intensity

There are difference in emissions among countries due to various factors: geographical location, type of fuel used, level of economic and technological development. To account for the regional dependence, we use the emission intensity coefficient γ that is the weight in kilograms of emitted CO_2 per each megawatt-hour (MW h) of electricity generated by a particular power sector of the country. The emission intensity coefficient is defined by regional energy consumption, or $\gamma = \sum_i f_i e_i$, where i is the ith energy source (e.g., coal, renewable, petroleum, gas, etc.), f_i is a fraction of the ith energy source for specific region, and e_i is its emission intensity coefficient. Therefore, the higher is the fraction of renewable energy is, the less is the total emission intensity coefficient. the opposite case, a high fraction of hydrocarbon energy resources implies a higher value of emission intensity coefficient. Thereby, the carbon emission intensity varies significantly depending on the region (see Table 2).

The *eco2AI* library includes a continuously supported database of emission intensity coefficients for 365 regions based on the publicly available data in 209 countries [11], as well as regional data for such countries as Australia (Emissions factors sources [13], Science and Resources [39]), Canada (Emissions factors sources [13], UNFCCC [43]), Russia (Rosstat [34], EMISS [12], Minprirody (Russia) [28]), and the USA (Emissions factors sources [13], USA EPA [44]). Currently, this is the largest database among the reviewed trackers allowing higher accuracy of energy consumption estimations.

The database contains the following data: country name, ISO-Alpha-2 code, ISO-Alpha-3 code, UN M49 code and emission coefficient value. As an example, the data for selected regions are presented in Table 2. The *eco2AI* library automatically determines the country and region of the user by IP and chooses the corresponding emission intensity coefficient. If the coefficient cannot be determined, it is set to 436.5 kg/(MW h), which is the global average [11]. In *eco2AI* region, country, emission intensity coefficient can be specified manually.

3.3. Equivalent Carbon Emissions

The total equivalent emission value of carbon footprint (CF) generated during AI models learning is defined by multiplication of the total power consumption from CPU, GPU, and RAM by the emission intensity coefficient γ (kg/(kW h)) and the PUE coefficient:

$$CF = \gamma PUE(E_{\text{CPU}} + E_{\text{GPU}} + E_{\text{RAM}}).$$

Here, PUE is the power usage effectiveness of data center when training is done on a cloud. PUE is an optional parameter with a default value of 1. It is defined manually in the *eco2AI* library.

4. EXPERIMENTS

In this section, we present experiments of tracking equivalent CO_2 emissions using *eco2AI* while training of Malevich (ruDALL-E XL 1.3B) and Kandinsky

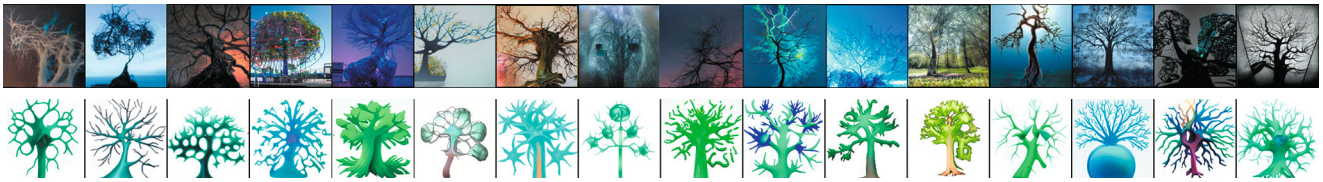

Fig. 2. Images generation of Malevich (top) vs Emojich XL (bottom) by text input "Tree in the form of a neuron."

Fig. 3. Images generation of Kandinsky (top) vs Emojich XXL (bottom) by text input "Green artificial intelligence."

(ruDALL-E XXL 12B) models. Malevich and Kandinsky are large multimodal generative models [18] with 1.3 and 12 billion parameters, respectively, capable of generating arbitrary images from a textual description.

We present the results of fine-tuning Malevich and Kandinsky on the Emojis dataset [40] and of training Malevich with an optimized version of the GELU activation function [21].

4.1. Fine-Tuning of Multimodal Models

In this section, we present *eco2AI* use cases for monitoring the fine-tuning of Malevich and Kandinsky models characteristics (e.g., CO_2, kg; power, kW h) on the Emojis dataset. Malevich and Kandinsky are multi-modal pre-trained transformers that learn the conditional distribution of images. More precisely, they autoregressively model text and image tokens as a single stream of data (see, e.g., DALL-E [33]). These models are transformer decoders [45] with 24 and 64 layers, 16 and 60 attention heads, 2048 and 3840 hidden dimensions, respectively, and standard GELU nonlinearity.

Both Malevich and Kandinsky work with 128 text tokens, which are generated from the text input using YTTM tokenizer (YouTokenToME), and 1024 image tokens, which are obtained encoding the input image using generative adversarial network Sber-VQGAN encoder part (sber-vq-gan) (it is pretrained VQGAN [15] with Gumbel Softmax Relaxation [25]).

The dataset of Emojis (russian-emoji) used for fine-tuning contains 2749 unique emoji icons and 1611 unique texts that were collected by web scrapping (the difference in quantities is due to the fact that there are sets, within which emojis differ only in color, moreover, some elements are homonyms).

The Malevich and Kandinsky models were trained in fp16 and fp32 precision, respectively. Adam 8-bit optimizer [7] used for optimization in both experiments. This realization reduces the amount of GPU memory required for gradient retention. OneCycle learning rate is chosen as a scheduler with the following parameters: start learning rate (lr) 4×10^{-7}, max lr 10^{-5}, final lr 2×10^{-8}. The models were fine-tuned for 40 epochs with a warmup lr of 0.3, batch size of 4 for Male-vich and batch size of 12 for Kandinsky, with large image loss coefficient of 1000 and with frozen feed forward and attention layers.

Distributed model training optimizer DeepSpeed (ZeRO-3) was used to train the Kandinsky model. The source code used for fine-tuning of Malevich is available in Kaggle (emojich ruDALL-E). Malevich and Kandinsky models were trained at 1 GPU Tesla A100 (80 Gb) and 8 GPU Tesla A100 (80 Gb), respectively.

We have named the results of Malevich and Kandinsky fine-tuning as Emojich XL and Emojich XXL, respectively. We compare the results of image generation by Malevich vs Emojich XL and Kandinsky vs Emojich XXL on some text inputs (see Figs. 2 and 3) in order to assess visually the quality of fine-tuning (how much the style of generated images is adjusted to the style of emojis). The image generation starts with a text prompt that describes the desired content. When the tokenized text is fed to Emojich, the model generates the remaining image tokens auto-regressively.

Every image token is selected item-by-item from a predicted multinomial probability distribution over the image latent vectors using kernel top-p and top-k sampling with a temperature [23] as a decoding strat-

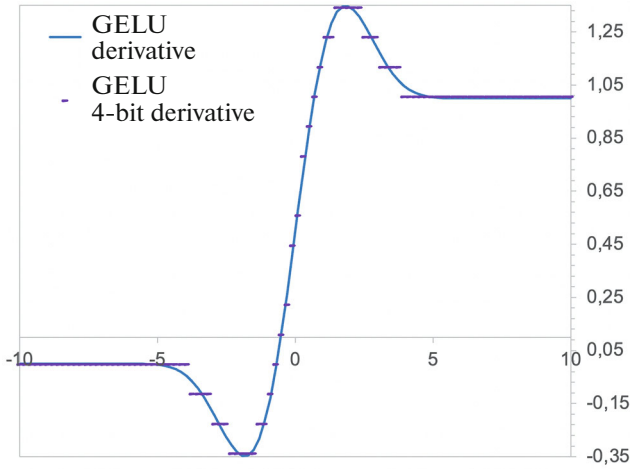

Fig. 4. Optimized 4-bit piecewise-constant approximation of the derivative of the GELU activation function.

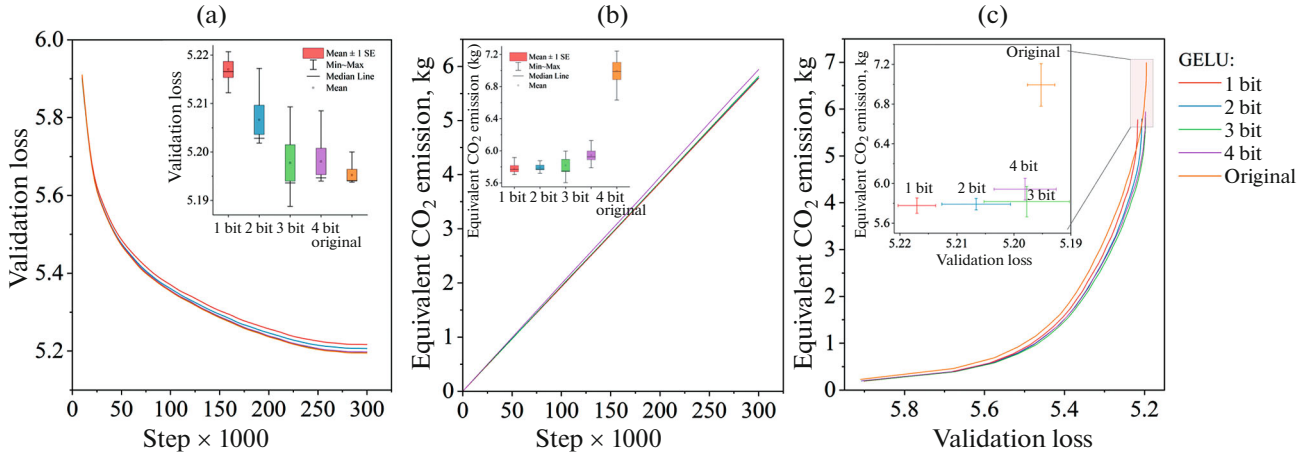

Fig. 5. The comparison of GELU and GELU few-bit activation functions integrated to Malevich model: (a) Validation loss at every step of pre-training (box-plot within the inset indicates deviation of validation loss of each model at 300 000 step), (b) accumulated CO_2 at every step of models pre-training (box-plot within the inset indicates deviation of accumulated CO_2 of each model at 300 000 step), (c) accumulated CO_2 for achieved validation loss of each model (the inset depicts zoomed area of graph with peak accumulated CO_2 to stress the difference between models).

egy. The image is rendered from the generated sequence of latent vectors by the decoder part of the Sber-VQGAN.

All examples below are generated automatically with the following hyper-parameters: batch size 16 and 6, top-k 2048 and 768, top-p 0.995 and 0.99, temperature 1.0, 1 GPU Tesla A100 for Malevich (as well as Emojich XL) and Kandinsky (as well as Emojich XXL), respectively.

Summary of fine-tuning parameters, energy consumption results and eq. CO_2 is given in (Table 3). One can note that the fine-tuning Kandinsky consumes more than 17 times more energy than Malevich.

Thus, the *eco2AI* library makes it straightforward to control the energy consumption while training (and fine-tuning) large models not only on one GPU, but also on multiple GPUs, which is essential when using optimization libraries for distributed training, for example, DeepSpeed.

4.2. Pre-training of Multimodal Models

Training large models like Malevich is highly resource demanding. In this section we give an example of model improvement in terms of energy efficiency referring to low precision computing using few-bits GELU activation function as an example. Few-bits GELU [29] is a variation of the GELU [21] activation function that preserves model gradients with few-bit resolution, thus allocating less GPU memory and spending less computational resources (see Fig. 4).

Table 3. Carbon emissions and power consumption of the fine-tuning of Malevich and Kandinsky models

Model	Train time	Power, kWh	CO_2, kg	GPU	CPU	Batch size
Malevich	4h 19m	1.37	**0.33**	A100 Graphics, 1	AMD EPYC 7742 64-Core	4
Kandinsky	9h 45m	24.50	**5.89**	A100 Graphics, 8	AMD EPYC 7742 64-Core	12

Table 4. Carbon emissions and power consumption of the pre-trained Malevich model on 300 000 dataset during 1 epoch (A100 Graphics, AMD EPYC 7742 64-Core)

Model	Train time, h	Power, kW h	CO_2, kg	Valid loss
Malevich, GELU original	75.2 ± 1.3	29.1 ± 1	7.0 ± 0.24	5.195 ± 0.002
Malevich, GELU 4-bit	67.2 ± 1.5	24.7 ± 0.5	5.94 ± 0.12	5.198 ± 0.005
Malevich, GELU 3-bit	67.2 ± 1.5	24.2 ± 0.7	5.81 ± 0.17	5.198 ± 0.008
Malevich, GELU 2-bit	66.5 ± 0.3	24.0 ± 0.3	5.79 ± 0.06	5.207 ± 0.006
Malevich, GELU 1-bit	67.7 ± 0.73	24 ± 0.36	5.77 ± 0.09	5.217 ± 0.003

More precisely, we compare training Malevich with regular GELU and version of Malevich with GELU 4-bit, 3-bit, 2-bit, and 1-bit sampling using the *eco2AI* library.

We used the same optimizer, scheduler, and training strategy as in the fine-tuning experiments. To ensure reproducibility, we ran each experiment 5 times with a random seed. The training dataset consisted of 300 000 samples. Each sample was passed through the model only once with a batch size of 4. The validation dataset consisted of 100 000 samples. The *eco2AI* library was used to track the carbon footprint during the training in real-time.

As we can see in Fig. 5a, the validation loss of Malevich with 4-bit and 3-bit GELU and Malevich with regular GELU is almost the same (0.06% growth), whereas the 2-bit GELU and 1-bit GELU demonstrate an about 0.42% increase in validation loss. In contrast, 4-bit GELU and 3-bit GELU are about 15 and 17% more efficient, respectively, comparing to the original GELU and accumulate less CO_2 emissions at the same training step (Fig. 5b).

The use of 1-bit GELU resulted in an additional saving of only 0.05% CO_2. The performance of the models is summarized in Fig. 5c and Table 4. The 3-bit GELU seems to provide model performance close to the original GELU, while consuming 17% less power and hence, producing less equivalent CO_2 emissions.

Thus, the *eco2AI* library can monitor the power consumption and carbon footprint of training models in real-time and helps to implement and demonstrate various memory and power optimization algorithms (such as quantization of gradients of activation functions).

5. CONCLUSIONS

In spite of the great potential of AI to solve environmental issues, AI itself can be the source of an indirect carbon footprint. In order to help the AI-community to understand the environmental impact of AI models during training and inference and to systematically monitor equivalent carbon emissions in this paper we introduced the *eco2AI* tool. The *eco2AI* is an open-source library capable of tracking equivalent carbon emissions while training or inferring Python-based AI models accounting for energy consumption of CPU, GPU, RAM devices. In *eco2AI* we focused on accuracy of energy consumption tracking and correct regional CO_2 emissions accounting due to precise measurement of process loading, extensive database of regional emission coefficients and CPU devices.

We present examples of *eco2AI* usage for tracking the fine-tuning of big text2image models Malevich and Kandinsky, as well as for optimization of GELU activation function integrated to Malevich model. For example, using *eco2AI*, we have demonstrated that usage of 3-bit GELU decreased equivalent CO_2 emissions by about 17%. We expect that *eco2AI* can help the ML community to pace to Green and Sustainable AI within the presented concept of the AI-based GHG sequestrating cycle.

APPENDIX

USAGE OF *eco2AI* LIBRARY

The *eco2AI* library is available as a Python package. It is open-source, distributed under the Apache 2.0 license [3],[1] available for download and installation from PyPI [32],[2] and source-code deposited on GitHub [10].[3]

After installation and import into the Python session, it is necessary to add *.start()* method before the main code and *.stop()* method after it (see Listing 1).

[1] https://www.apache.org/licenses/LICENSE-2.0.
[2] https://pypi.org/project/eco2AI/.
[3] https://github.com/sb-ai-lab/eco2AI.

Listing 1: Base example of using

```
import eco2ai
tracker = eco2ai.Tracker(project_name="YourProjectName,"
        experiment_description="training_the_<your_model>_model")

tracker.start()
<your gpu & (or) cpu calculations>
tracker.stop()
```

Another way of tracker usage is to use decorators (see Listing 2). Decorator *track* allows wrapping any function and writing emission information in *"emission.csv"* file every time when it is executed.

Listing 2: Example of using decorators

```
from eco2ai import track

@track
def train_func(model, dataset, optimizer, epochs, *args, **kwargs):
  ...
train_func(model, dataset, optimizer, epochs, *args, **kwargs)
```

Every time the method *.stop()* is executed, all the results is written in a local file named *"emission.csv."* The file name can be set by the user (parameter *file_name* of the *eco2ai.Tracker()* class). The resulting file includes the following data:

- *id* is the unique experiment ID;
- *project_name* is the project name given by the user;
- *experiment_description* is the experiment description given by the user;
- *epoch* is epoch information. When epoch accounting is enabled (see Listing 2) here can be placed any epoch-specific information during training of ML models;
- *start_time* is the the start date of the experiment in the following format: yyyy-mm-dd hh:mm:ss;
- *duration(s)* is the experiment duration in seconds;
- *power_consumption(kWh)* is the energy indirectly spent during the experiment in kW h;
- *CO_2_emissions(kg)* is the weight of CO_2 indirectly spent during the experiment in kg;
- *CPU_name* is the model name and the number of CPU used during the experiment;
- *GPU_name* is the model name and the number of GPU used during the experiment;
- *OS* is the operation system of the device used for training;
- *region/country* is the country and region automatically defined by IP or manually defined by the user via setting the parameters *alpha_2_code* and *region* of the *eco2ai.Tracker()* class;
- *cost* is the cost of electricity spent during the experiment. It is calculated only when the *electricity_pricing* parameter is set.

The *eco2AI* allows one to record information about training sessions in encrypted form (see Listing 3). If the parameter *encode_file* of the *eco2ai.Tracker()* class is enabled, encrypted data is written to the *"encoded_emission.csv"* file. This functionality is useful when the authenticity of results is required.

Listing 3: Function of encoding data

```
import eco2ai

tracker = eco2ai.Tracker(
        file_name='encoded_emissions.csv',
        project_name="Test_1",
        experiment_description="testing_Eco2AI_in_encoding_mode,
        encode_file=True,
)
```

eco2AI also provides the possibility of accounting for the training process of AI models during training. Listing 4 illustrates the case of using this possibility on PyTorch framework.

Listing 4: Experiment for ML and DL

```
tracker = eco2ai.Tracker(
        project_name="CIFAR10_for_ESG",
        experiment_description="ML_tracking",
        file_name="emission.csv",
        emission_level=None,
        alpha_2_code="AU",
        region="Queensland",
        )
tracker.start_training()

net = Net()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
parameters_to_save = dict()
for epoch in range(epochs):
parameters_to_save["loss"] = train_epoch(net, optimizer, trainloader)
        parameters_to_save["train_accuracy"] = get_accuracy(net, trainloader)
        parameters_to_save["test_accuracy"] = get_accuracy(net, testloader)

        tracker.new_epoch(parameters_to_save)
tracker.stop_training()
```

Comprehensive and detailed documentation for every method or parameter of *eco2ai.Tracker()* class that is enough for advanced usage of the library and can be obtained by running *help(eco2ai.Tracker())* function.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

OPEN ACCESS

REFERENCES

1. Paris Agreement, in *Report of the Conference of the Parties to the United Nations Framework Convention on Climate Change* (*21st Session, 2015, Paris*). Retrieved December, HeinOnline (2015), Vol. 4, p. 2017.
2. L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," arXiv preprint arXiv:2007.03051 (2020).
3. Apache License 2.0. https://www.apache.org/licenses/LICENSE-2.0
4. Carbontracker. https://github.com/lfwa/carbontracker
5. Cloud Carbon Footprint. https://github.com/cloud-carbon-footprint/cloud-carbon-footprint
6. Codecarbon. https://github.com/mlco2/codecarbon
7. T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, "8-bit optimizers via blockwise quantization," arXiv preprint arXiv:2110.02861 (2021).
8. P. Dhar, "The carbon impact of artificial intelligence," Nat. Mach. Intell. **2** (8), 423–425 (2020).
9. J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," arXiv preprint arXiv:2002.06305 (2020).
10. eco2ai GitHub. https://github.com/sb-ai-lab/eco2AI
11. Ember. Global Electricity Review 2022, March 2022. https://ember-climate.org/insights/research/global-electricity-review-2022/.

12. EMISS. The Unified Interdepartmental Information and Statistical Systems. https://fedstat.ru/indicator/58506
13. Emissions Factors Sources, 2021. https://www.carbonfootprint.com/docs/2022_01_emissions_factors_-sources_for_2021_electricity_v10.pdf
14. Emojich-ruDALL-E. https://www.kaggle.com/shonenkov/emojich-rudall-e
15. P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis" (2020).
16. Experiment Impact Tracker. https://github.com/Breakend/experiment-impact-tracker
17. Green Algorithms Tool. https://github.com/GreenAlgorithms/green-algorithms-tool
18. J. Gusak, D. Cherniuk, A. Shilova, A. Katrutsa, D. Bershatsky, X. Zhao, L. Eyraud-Dubois, O. Shlyazhko, D. Dimitrov, I. Oseledets, and O. Beaumont, "Survey on large scale neural network training" (2022).
19. S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," arXiv preprint arXiv:1510.00149 (2015).
20. 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," J. Mach. Learn. Res. **21**, 1–43 (2020).
21. D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," arXiv preprint arXiv:1606.08415 (2016).
22. G. Hinton, O. Vinyals, J. Dean, et al., "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531 (2015).
23. A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," CoRR, arXiv preprint arXiv:1904.09751 (2019).
24. V. Joler and K. Crawford, "Anatomy of an AI system" (2018). http://www.anatomyof.ai
25. M. J. Kusner and J. M. Hernández-Lobato, "GANS for sequences of discrete elements with the Gumbel-softmax distribution" (2016).
26. L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: Quantifying the carbon footprint of computation," Adv. Sci. **8** (12), 2100707 (2021).
27. D. A. Maevsky, E. J. Maevskaya, and E. D. 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. Minprirody (Russia). https://xn--d1ahaoghbejbc5k.xn--p1ai/documents/active/664/.
29. G. Novikov, D. Bershatsky, J. Gusak, A. Shonenkov, D. Dimitrov, and I. Oseledets, "Few-bit backward: Quantized gradients of activation functions for memory footprint reduction" (2022).
30. D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," arXiv preprint arXiv:2104.10350 (2021).
31. M. Pesce, "Cloud computing's coming energy crisis," IEEE Spectrum (2021).
32. PyPi Eco2AI. https://pypi.org/project/eco2AI/
33. A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, July 18–24, 2021*, Ed. by M. Meila and T. Zhang, Virtual Event, *Proceedings of Machine Learning Research* (PMLR, 2021), Vol. 139, pp. 8821–8831. http://proceedings.mlr.press/v139/ramesh21a.html
34. Rosstat. Russian Federal State Statistics Service. https://rosstat.gov.ru/enterprise_industrial
35. ruDALL-E XL 1.3B. rudall-e: Generating images from text. facing down the biggest computational challenge in Russia. https://habr.com/ru/company/sberbank/blog/589673/
36. ruDALL-E XXL 12B. https://github.com/sberbank-ai/ru-dalle
37. Russian-Emoji. https://www.kaggle.com/datasets/shonenkov/russian-emoji
38. Sber-VQ-GAN. https://github.com/sberbank-ai/sber-vq-gan
39. Science and Resources. National Greenhouse Accounts Factors. https://www.industry.gov.au/sites/default/files/August%202021/document/national-greenhouse-accounts-factors-2021.pdf
40. A. Shonenkov, D. Bakshandaeva, D. Dimitrov, and A. Nikolich, "Emojich zero-shot emoji generation using Russian language: A technical report," arXiv preprint arXiv:2112.02448 (2021).
41. E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," arXiv preprint arXiv:1906.02243 (2019).
42. Tracarbon. https://github.com/fvaleye/tracarbon
43. UNFCCC. Canada, National Inventory Report (2021), Part 3, p. 60. https://unfccc.int/sites/default/files/resource/can-2021-nir-12apr21.zip
44. USA EPA. egrid2020, May 2020. https://www.epa.gov/system/files/documents/2022-01/egrid2020_data.xlsx
45. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*, Ed. by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett (2017), pp. 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
46. R. Vinuesa, H. Azizpour, I. Leite, M. Balaam, V. Dignum, S. Domisch, A. Felländer, S. D. Langhans, M. Tegmark, and F. F. Nerini, "The role of artificial intelligence in achieving the sustainable development goals," Nature Commun. **11** (1), 1–10 (2020).
47. YouTokenToMe. https://github.com/VKCOM/YouTokenToMe
48. ZeRO-3. https://www.deepspeed.ai/2021/03/07/zero3-offload.html