ECO2AI: CARBON EMISSIONS TRACKING OF MACHINE LEARNING MODELS AS THE FIRST STEP TOWARDS SUSTAINABLE AI

Semen Budennyy[1, 2*], Vladimir Lazarev[2], Nikita Zakharenko[1], Alexey Korovin[2], Olga Plosskaya[1], Denis Dimitrov[1, 2], Vladimir Arkhipkin[1], Ivan Oseledets[2, 3], Ivan Barsola[1], Ilya Egorov[1], Aleksandra Kosterina[1], and Leonid Zhukov[4]

[1]Sber (AI Lab, Sber AI, ESG), Moscow
[2]Artificial Intelligence Research Institute (AIRI), Moscow
[3]Skolkovo Institute of Science and Technology, Moscow
[4]Higher School of Economics University, Moscow
[*]Corresponding author: Semen Budennyy, sanbudenny@sberbank.ru

ABSTRACT

The size and complexity of deep neural networks continue to grow exponentially, significantly increasing energy consumption for training and inference by these models. We introduce an open-source package *eco2AI*[1] to help data scientists and researchers to track energy consumption and equivalent CO_2 emissions of their models in a straightforward way. In *eco2AI* we put emphasis on accuracy of energy consumption tracking and correct regional CO_2 emissions accounting. We encourage research community to search for new optimal Artificial Intelligence (AI) architectures with a lower computational cost. The motivation also comes from the concept of AI-based green house gases sequestrating cycle with both Sustainable AI and Green AI pathways.

Keywords ESG · Sustainable AI · Green AI · Sustainability · Ecology · Carbon footprint · CO_2 emissions · GHG

1 Introduction

While the global ESG agenda (Environment, Social, and Corporate Governance) is guided by agreements established between countries[1]), the development of ESG principles is happening 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 ESG agenda, it remains the problem of transparent and objective quantitative evaluation of ESG progress in particular in environmental protection. This is of great importance for IT industry, as 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 today's IT industry are rapidly evolving technologies with massive potential for disruption. There are 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 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 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 within the concept of Green AI. The final goal of these solutions is the reduction of Green House Gases (GHG) emissions. In fact, AI can help to reduce the effects of the climate crisis, for example, in smart grid design, developing low-emission infrastructure and modelling

[1]Source code for *eco2AI* is available at `https://github.com/sb-ai-lab/Eco2AI`

climate changes.[4] However, it is also crucial to account for generated CO_2 emissions while training AI models. In fact, development of AI results into increasing computing complexity and, thereby, electrical energy consumption and resulting equivalent carbon emissions (eq. CO_2). The ecological impact of AI is a major concern that we need to account for to be aware of eventual risks. We need to ensure ML models to be environmentally sustainable, to be optimized not only in term of 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. Clear understanding of ecological impact from AI motivates data science community to search for optimal architectures consuming less computer resource. An explicit call to promote research on more computationally efficient algorithms was mentioned elsewhere.[5]

To summarize the previous theses, we present the concept of AI-based GHG sequestrating cycle that describes the relationship of AI with sustainability goals (Figure 1). The request from Sustainability towards AI spawns demand for more optimized models in terms of energy consumption forming the path we named "Towards Sustainable AI". On the other hand, AI creates additional opportunities for sustainability goals' achievement, and we suggest naming this path "Towards Green AI". To understand the role of *eco2AI* library in this cycle, in the right part of Figure 1 the scheme is given with paths mentioned. First, *eco2AI* motivates to optimize AI technology itself. Second, if AI is aimed to sequestrate the GHG, then the total effect should be evaluated with account for generated eq. CO_2 during training sessions at least (and during model exploitation at its best). In the frame of this article, we are constrained to examining the path "Towards Sustainable AI" only (see examples in the Chapter "Experiments").

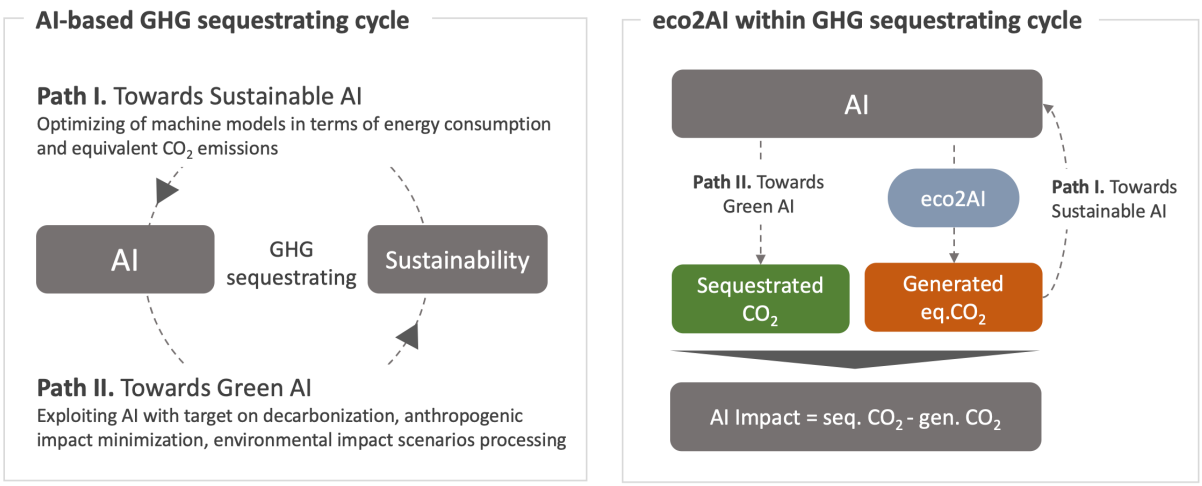

Figure 1: High-level schemes of AI-based GHG sequestrating. The left scheme corresponds to AI-based GHG sequestrating cycle. The right scheme describes the role of *eco2AI* in this scheme

Contribution. The contribution of our paper is threefold:

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

The paper is organized as follows. In section **2** we review the existing solutions for CO_2 assessment and describe their difference from our library. Section **3** presents the methodology of calculations, section **4** shows the use case of the library. Finally, in section **5** we summarize our work. The appendix section describes briefly the code usage.

2 Related work

In this chapter, we describe recent practices of CO_2 emissions evaluation for AI-based models. Further on, we give a brief description of the existing open-source packages, providing the summary of comparisons.

2.1 Practice of AI equivalent carbon emissions tracking

Since the appearance of DL models, their complexity has been increasing exponentially, doubling number of parameters every 3-4 months since 2012 [6] and reaching more than a trillion parameters in 2022. Among the most well known models are BERT-Large (Oct 2018, 340M), GPT-2 (2019, 1.5B), T5 (Oct, 2019, 11B), GPT-3 (2020, 175B), Megatron Turing (530M), Switch Transformer (2022, 1.6T).

Data accumulation, labeling, storage, processing and exploitation consumes a lot of resources during their 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 an example.[7] 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.[8]

Large-scale research was conducted focusing on quantifying the approximate environmental costs of DL widely used for NLP problems.[5] Among the examined DL architecture, there were Transformer, ELMo, BERT, NAS, GPT-2. The total power consumption was evaluated as combined GPU, CPU and DRAM consumptions, multiplied by data center specific Power Usage Effectiveness (PUE) with default value equals 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 generally carried out by multiplication of total energy consumption and carbon energy intensity. The authors estimated that carbon footprint for training BERT (base) was about 652 kg that is comparable to the footprint of the "New York <-> San Francisco" air travel per passenger.

The energy consumption and carbon footprint for the following NLP models was estimated : T5, Meena, GShard, Switch Transformer, GPT-3.[9] The key outcome resulted in opportunities to improve energy efficiency while training neural network models: sparsely activating DL; distillation techniques [10]; pruning, quantization, efficient coding [11]; fine-tuning and transfer-learning [12]; large models training in a specific region with low energy mix, exploiting cloud data centers optimized in terms of energy consumption. The authors advocated for reducing the carbon footprint by 10^2-10^3 times if the mentioned suggestions had been taken into account.

2.2 Review of open-source emission trackers

A list of libraries have been developed to track the AI equivalent carbon footprint. Here we are focusing on describing the most widespread open-source libraries. They all have a common key goal: to monitor CO_2 emissions during training models (see Table 1). Having much in common with recent analogs, in *eco2AI* we focused on the following: taking into account only those system processes that are related directly to models training (to avoid over-estimation); extensive and constantly updated database of regional emission coefficients (365 territorial objects are included) and information on CPU devices (3278 models).

Cloud Carbon Footprint[2] is an application that estimations the energy and carbon emissions of public cloud provider utilization. It measures cloud carbon and is intended to connect with various cloud service providers. It provides estimates for both energy and carbon emissions for all types of cloud usage, including embodied emissions from production, with the opportunity to drill down into emissions by cloud provider, account, service, and time period. It provides real recommendations for AWS and Google Cloud to save money and minimize carbon emissions, as well as forecasting cost savings and actual outcomes in the form of trees planted. For hyperscale data centers, it measures consumption at the service level using real server utilization rather than average server utilization. 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[3] is a Python package for tracking the carbon emissions produced by various kinds of computer programs, from straightforward algorithms to deep neural networks. By taking into account the computing infrastructure, location, usage and running time, CodeCarbon provides an estimate of how much CO_2 was produced, and gives comparisons with common modes of transportation to give an idea about scope within an order of magnitude.

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 to reducing carbon emissions by supporting predictions. Model training can be stopped at the user's discretion if the predicted environmental cost is exceeded. Authors support a variety of different environments and platforms such as clusters, desktop computers, and Google Colab notebooks, allowing for a plug-and-play experience. [13]

[2] `https://github.com/cloud-carbon-footprint/cloud-carbon-footprint`

[3] `https://github.com/mlco2/codecarbon`

[4] `https://github.com/lfwa/carbontracker`

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 Algo-rithms**	**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	✓	✓	Undefined	✓	✓	✓	✓*
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[15, 16], Canada[15, 17], Russia[18, 19] and USA[15, 20]
** *eco2AI* database includes data on 3278 models of CPU for Intel and AMD
*** beneficial in scenarios where the authenticity of results is required

Experiment impact tracker[5] is a framework providing information of energy, computational and carbon impacts of ML models. It includes the following features: extraction of CPU and GPU hardware information, setting experiment start and end-times, accounting for the energy grid region where the experiment is being run (based on IP address), the average carbon intensity in the energy grid region, memory usage, the real-time CPU frequency (in Hz).[8]

Green Algorithms[6] is online tool that enables a user to estimate and report the carbon footprint from computation. It integrates with computational processes and does not interfere with the existing code, while also accounting for a range of CPUs, GPUs, cloud computing, local servers and desktop computers.[14]

Tracarbon[7] 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.

3 Methododology

The methodology covers the following: calculation of electric energy consumption, extracting of emission intensity coefficient and conversion to equivalent CO_2 emissions. Each part is described below.

3.1 Electric energy consumption

The energy consumption of the system can be measured in Joules (J) or kilowatt-hours (kWh) - unit of energy equal to one kilowatt of power sustained for one hour. The task is to evaluate energy contribution for each hardware unit.[8] We focused on the 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 aware of importance of tracking terminating processes but we neglect those tail effect for its relatively small impact to the total energy consumption. The storage (SSD, HDD) is also an energy consuming process but we do not take it into account as it has lack of direct relationship with running process (it is rather an issue of permanent data storage process).

GPU. The *eco2AI* library is able to detect NVIDIA devices. A Python interface for GPU management and monitoring functions was implemented within the *Pynvml* library. This is a wrapper for the NVIDIA Management Library which detects most of 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

[5] `https://github.com/Breakend/experiment-impact-tracker`
[6] `https://github.com/GreenAlgorithms/green-algorithms-tool`
[7] `https://github.com/fvaleye/tracarbon`

installation on a computing machine. The total energy consumption of all active GPU devices E_{GPU} (kWh) equals to product of power consumption of GPU device and its loading time: $E_{GPU} = \int_0^T P_{GPU}(t)dt$, where P_{GPU} is total power consumption of all GPU devices defined by *Pynvml* (kW), T is GPU devices loading time (h). If the tracker does not detect any GPU device, then GPU power consumption is set equal to zero.

CPU. The python modules *os* and *psutil* were used to monitor CPU energy consumption. To avoid overestimation, *eco2AI* takes into account the current process running in the system related only to model training. Thereby, the tracker takes percentage of CPU utilization and divides it by number of CPU cores, obtaining CPU utilization percent. We realized currently the most comprehensive database containing 3279 unique processors for Intel and AMD models. For each CPU model name provided thermal design power (TDP) which is equivalent power consumption at long-term loadings. The total energy consumption of all active CPU devices E_{CPU} (kWh) is calculated as a product of the power consumption of the CPU devices and its loading time $E_{CPU} = TDP \int_0^T W_{CPU}(t)dt$, where TDP is equivalent CPU model specific power consumption at long-term loading (kW), 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 constant value equal to 100 W[21].

RAM. Dynamic random access memory devices is important source of energy consumption in modern computing systems especially when significant amount data should be allocated or processed. However, accounting of RAM energy consumption is problematic as its power consumption is strongly depends if data is read, written or maintained. In *eco2AI* RAM power consumption is considered proportional to amount of allocated power by current running process calculated as follows: $E_{RAM} = 0.375 \int_0^T M_{RAM_i}(t)dt$, where E_{RAM} - power consumption of all allocated RAM (kWh), M_{RAM_i} is allocated memory (GB) measured via *psutil* and 0.375 W/Gb is estimated specific energy consumption of DDR3, DDR4 modules[21].

3.2 Emission intensity

There is variation in emissions among countries due to different factors, such as climate change, geographical position, economic development, fuel use and technological advancement. To account for regional dependence we use the emission intensity coefficient γ that is a weight in kilogram of emitted CO_2 per each megawatt-hour (MWh) of electricity generated by the particular power sector of the country. The emission intensity coefficient is totally defined by regional energy mix, or $\gamma = \sum_i f_i e_i$, where i is an index related to the i-th energy source (e.g. coal, renewable, petroleum, gas, etc.), f_i is a fraction of the i-th energy source for specific region, e_i is its emission intensity coefficient. Consequently, the higher fraction of renewable energy is, the less the total emission intensity coefficient we expect. In the opposite case, high fraction of hydrocarbon energy resources implies a higher value of emission intensity coefficient. Thereby, the emission intensity varies significantly depending on the regional allocation (see Table 2).

Table 2: Emission intensity coefficients for selected regions

Country	**ISO-Alpha-2 code**	**ISO-Alpha-3 code**	**UN M49 code**	**Emission coefficient, kg/MWh**
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

The *eco2AI* library includes permanently enriched and maintained database of emission intensity coefficients for 365 regions based on the public available data in 209 countries[22] and also regional data for such countries as Australia[15, 16], Canada[15, 17], Russia[18, 19, 23] and the USA[15, 20]. Currently, this is the largest database among the trackers reviewed, which allows to enrich the higher precision 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 is presented in Table 2. The *eco2AI* library automatically defines a user calculation facility country by IP and extracts its emission intensity coefficient. If the coefficient is not extracted for some reason, it is set to 436.5 kg/MWh, which is global average.[22]

3.3 Equivalent carbon emissions

Finally, the total equivalent emission value as an AI carbon footprint CF (kg) generated during models learning is defined by multiplication of total power consumption from CPU, GPU and RAM by emission intensity coefficient γ

(kg/kWh) and PUE coefficient: $CF = \gamma \cdot PUE \cdot (E_{CPU} + E_{GPU} + E_{RAM})$. Here, PUE is power usage effectiveness of data center required if the learning process is run on cloud. PUE is the optional parameter with default value 1. It is defined manually in the *eco2AI* library.

4 Experiments

In the current chapter, we present experiments of tracking equivalent CO_2 emissions using *eco2AI* while training of Malevich (ruDALL-E XL 1.3B) [24] and Kandinsky (ruDALL-E XXL 12B)[8] models. Malevich and Kandinsky are large multimodal models[25] with 1.3 billion and 12 billion parameters correspondingly capable of generating arbitrary images from a russian text prompt that describes the desired result.

We present results for fine-tuning Malevich and Kandinsky on the Emojis dataset[26] and for training of Malevich with optimised variation of GELU[27] activation function. Training of the last mentioned version of Malevich allows us to consume about 10% less power and, consequently, produce less equivalent CO_2 emissions.

4.1 Fine-tuning of multimodal models

In this section we present *eco2AI* use cases for monitoring fine-tuning of Malevich and Kandinsky models characteristics (e.g., CO_2, kg; power, kWh) on the Emojis dataset. Malevich and Kandinsky are multi-modal pre-trained transformers that learn the conditional distribution of images with by some string of text. More precisely, they autoregressively model the text and image tokens as a single stream of data (see, e.g., DALL-E [28]). These models are transformer decoders [29] 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[9], and 1024 image tokens, which are obtained encoding the input image using generative adversarial network Sber-VQGAN encoder part[10] (it is pretrained VQGAN [30] with Gumbel Softmax Relaxation [31]). The dataset of Emojis[11] 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).

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

Malevich and Kandinsky were trained in fp16 and fp32 precision correspondingly. Adam (8-bit) [32] is used for optimization in both experiments. This realization reduces the amount of GPU memory required for gradient statistics. One cycle learning rate is chosen as a scheduler with the following parameters: start learning rate (lr) $4 \cdot 10^{-7}$, max lr 10^{-5}, final lr $2 \cdot 10^{-8}$. Models fine-tuned for 40 epochs with warmup 0.1, gradient clipping 1.0, batch size 4 for Malevich and batch size 12 for Kandinsky, with large image loss coefficient 1000 and with frozen feed forward and attention layers. Malevich and Kandinsky model were trained at 1 GPU Tesla A100 (16 GB) and 8 GPU Tesla A100 (80 Gb), respectively. It is worth mentioning that distributed model training optimizer DeepSpeed ZeRO-3 [33] was used to train Kandinsky model. The source code used for fine-tuning of Malevich is available in Kaggle[12]. Summary of fine-tuning parameters, energy consumption results ans eq. CO2 is given in (Table 3). One can note that fine-tuning of Kandinsky consume more than 17 times more than Malevich.

We have named the results of Malevich and Kandinsky fine-tuning as Emojich XL and Emojich XXL respectively. We compare the results of generation by Malevich vs by Emojich XL and by Kandinsky vs by Emojich XXL on some text inputs (see Figures 2 and 3) to assess visually the quality of fine-tuning (how the style of generated images is adjusted to the style of emojis).

[8] `https://github.com/sberbank-ai/ru-dalle`

[9] `https://github.com/VKCOM/YouTokenToMe`

[10] `https://github.com/sberbank-ai/sber-vq-gan`

[11] `https://www.kaggle.com/datasets/shonenkov/russian-emoji`

[12] `https://www.kaggle.com/shonenkov/emojich-rudall-e`

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 nucleus top-p and top-k sampling with a temperature [34] as a decoding strategy. 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.

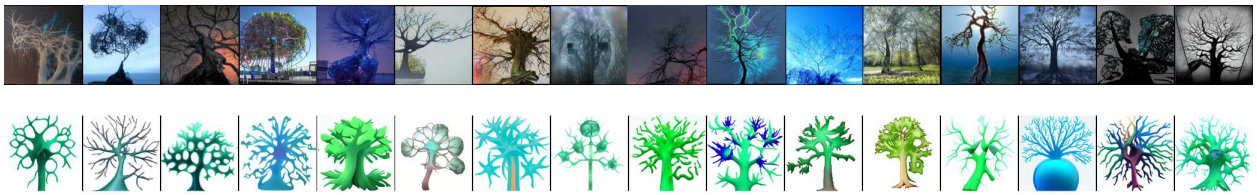

Figure 2: Images generaton of Malevich (top) vs Emojich XL (bottom) by text input "Tree in the form of a neuron"

Figure 3: Images generation of Kandinsky (top) vs Emojich XXL (bottom) by text input "Green Artificial Intelligence"

Thus, one can see 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 in case of using of optimisation libraries for distributed training, for example DeepSpeed ZeRO-3.

4.2 Pre-training of multimodal models

Training large models like Malevich is highly resource demanding task. In this section we give an example of improvement its energy efficiency referring to low precision computing using 4-bit GELU activation functon as example. More precisely, we compare training of version of Malevich with regular GELU and version of Malevich with GELU 4-bit using *eco2AI* library.

GELU 4-bit [35] is variation of GELU [27] activation function that saves model gradients with 4-bit resolution thus allocating less GPU memory and spending less computational resources (see Figure 4). Here we present the comparison of loss and energy efficiency Malevich model with integrated GELU and GELU 4-bit activation functions. We used the same optimizer, scheduler and training strategy as in fine-tuning experiments. To rule out randomness, we fixed seed equls to 6965. Training dataset was consisted of 250000 samples (pairs of images and corresponding description in natural language that was balanced over the following 15 domains: animal, nature, city, indoor, person, food, vehicle, device, tool, accessory, product, clothes, sport, art, other). Each sample was passed through the model only once with batch size 4. Validation dataset was consisted of 5000 samples (pairs of images and text that have been balanced over the same domains). *eco2AI* library was used to track the carbon footprint during the training in real time.

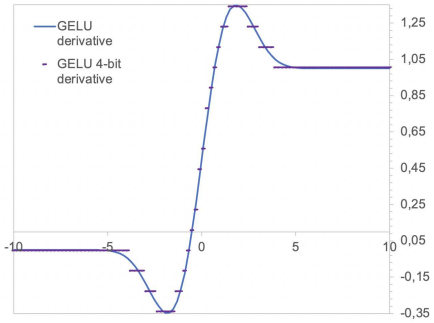

Figure 4: Optimized 4-bit piecewise-constant approximation of the derivative of the GELU activation function.

As we can see in Figure 5(a) validation losses of Malevich with GELU 4-bit and Malevich with regular GELU are almost the same. But GELU 4-bit is more efficient accumulating less CO_2 emissions at the same training step Figure 5(b) or achieved model loss Figure 5(c).

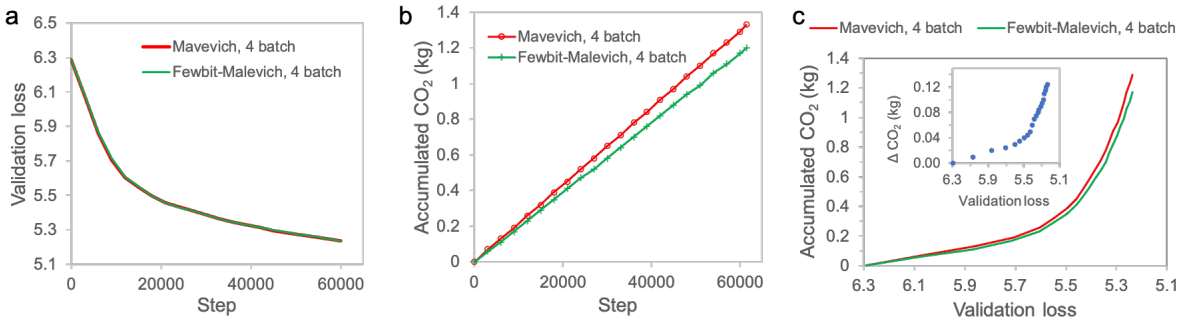

Figure 5: The comparison of GELU and GELU 4-bit activation functions integrated to Malevich model: (a) Validation loss at every step of pre-training, (b) Accumulated CO_2 at every step of models pre-training, (c) Accumulated CO_2 for achieved validation loss of each model (the inset depicts the difference of accumulated CO_2 between models)

As one can see in the Table 4 GELU 4-bit allows us to consume about 10% less power and, consequently, produce less equivalent CO_2 emissions.

Table 4: Carbon emissions and power consumption of the pre-trained Malevich model on 250000 dataset during 1 epoch

Model	Train time	Power, kWh	CO_2, kg	GPU	CPU	Valid Loss
Malevich	15h 23m	5.51	**1.33**	A100 Graphics, 1	AMD EPYC 7742 64-Core	5.24
Malevich, GELU 4-bit	14h 5m	4.99	**1.20**	A100 Graphics, 1	AMD EPYC 7742 64-Core	5.24

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

5 Conclusions

Despite the great potential of AI to solve environmental issues, AI itself can be the source of indirect carbon footprint. In order to help AI-community to understand the environmental impact of AI models during training and inference and to systematically monitor equivalent carbon emissions in the this paper we introduced the tool *eco2AI*. The *eco2AI* is an open-source library capable to track equivalent carbon emissions while training or inferring python-based AI models accounting for energy consumption of CPU, GPU, RAM devices. In *eco2AI* we put emphasis 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 fine-tuning of big text2image models Malevich and Kandinsky and also for optimisation of GELU activation function integrated to Malevich model. With the help of *eco2AI* we demonstrated that usage of 4-bit GELU decreased equivalent CO_2 emissions by about 10%. We expect that *eco2AI* could help the ML community to pace to Green an Sustainable AI within the presented concept of AI-based GHG sequestrating cycle.

Appendix. Usage of *eco2AI* library

The *eco2AI* library is available as Python package. It is open-source, distributed under under the Apache 2.0 license[13] and available for download and installation from PyPI [14] and one can also find its source-code on GitHub [15].

Once it is installed and imported into Python session, it will require to add start and stop code lines to frame the tracking session.

Listing 1: Code integration

```
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 to start working with the tracker is to use decorators. It allows marking any function and writing emission information in *"emission.csv"* file every time when it is executed.

Listing 2: using decorators

```
from eco2ai import track

@track
def train_func(model, dataset, optimizer, epochs):
    ...
train_func(your_model, your_dataset, your_optimizer, your_epochs)
```

After the end of the session, all the results will be recorded in a local file *"emission.csv"* or another name set by user. This file includes the following data: Project name (customized by user), Experiment description (customized by user), Start time (yyyy-mm-dd hh:mm:ss), Duration (sec), Power consumption (kWh), CO_2 emission (kg), CPU name, GPU name, OS, Country.

The *eco2AI* allows users to record information about training sessions in encrypted form as an extra function. This functionality is beneficial in scenarios where the authenticity of results is required. It is need to use the tracker property "*encode*" to enable output encryption.

Listing 3: using encrypted mode

```
import eco2ai

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

For users convenience, the *eco2AI* implements the summary function. It aggregates information in the .csv file by the "project name" column. If user defines the kWh_price argument, information about financial costs for each of the projects will be additionally calculated based on the duration time and price information provided by the user.

Listing 4: using summary function

```
eco2ai.summary('emission.csv',kwh_price=0.117)
```

[13] https://www.apache.org/licenses/LICENSE-2.0

[14] https://pypi.org/project/eco2AI/

[15] https://github.com/sb-ai-lab/eco2AI

References

[1] Paris Agreement. Paris agreement. In *Report of the Conference of the Parties to the United Nations Framework Convention on Climate Change (21st Session, 2015: Paris). Retrived December*, volume 4, page 2017. HeinOnline, 2015.

[2] M Pesce. Cloud computing's coming energy crisis. *IEEE Spectrum*, 2021.

[3] Ricardo Vinuesa, Hossein Azizpour, Iolanda Leite, Madeline Balaam, Virginia Dignum, Sami Domisch, Anna Felländer, Simone Daniela Langhans, Max Tegmark, and Francesco Fuso Nerini. The role of artificial intelligence in achieving the sustainable development goals. *Nature communications*, 11(1):1–10, 2020.

[4] Payal Dhar. The carbon impact of artificial intelligence. *Nat. Mach. Intell.*, 2(8):423–425, 2020.

[5] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[6] Open AI. AI and Compute. `https://openai.com/blog/ai-and-compute/`.

[7] Kate Crawford, VladanJoler's, and Vladan Joler. Anatomy of an ai system, 2018.

[8] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

[9] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[10] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[12] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

[13] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051*, 2020.

[14] Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green algorithms: quantifying the carbon footprint of computation. *Advanced science*, 8(12):2100707, 2021.

[15] Carbon footprint. 2021 report. `https://www.carbonfootprint.com/docs/2022_01_emissions_factors_sources_for_2021_electricity_v10.pdf`.

[16] Science Australian departnent of Industry and Resources. Australian national greenhouse accounts. national greenhouse accounts factors. `https://www.industry.gov.au/sites/default/files/August%202021/document/national-greenhouse-accounts-factors-2021.pdf`.

[17] United Nations Framework Convention on Climate Change (UNFCCC). Canada. national inventory report (nir), 2021. Part 3, page 60.

[18] Russian federal state statistics service. `https://rosstat.gov.ru/enterprise_industrial`.

[19] The unified interdepartmental information and statistical systems (emiss). `https://fedstat.ru/indicator/58506`.

[20] USA Environmental Protection Agency. egrid2020. `https://www.epa.gov/system/files/documents/2022-01/egrid2020_data.xlsx`, May 2022.

[21] DA Maevsky, EJ Maevskaya, and ED Stetsuyk. Evaluating the ram energy consumption at the stage of software development. In *Green IT Engineering: Concepts, Models, Complex Systems Architectures*, pages 101–121. Springer, 2017.

[22] Ember. Global electricity review 2022. `https://ember-climate.org/insights/research/global-electricity-review-2022/`, Mar 2022.

[23] Ministry of natural resources and environment (Russia). `https://xn--d1ahaoghbejbc5k.xn--p1ai/documents/active/664/`.

[24] ruDALL-E: Generating Images from Text. Facing down the biggest computational challenge in Russia. https://habr.com/ru/company/sberbank/blog/589673/.

[25] Julia Gusak, Daria Cherniuk, Alena Shilova, Alexander Katrutsa, Daniel Bershatsky, Xunyi Zhao, Lionel Eyraud-Dubois, Oleg Shlyazhko, Denis Dimitrov, Ivan Oseledets, and Olivier Beaumont. Survey on large scale neural network training, 2022.

[26] Alex Shonenkov, Daria Bakshandaeva, Denis Dimitrov, and Aleksandr Nikolich. Emojich–zero-shot emoji generation using Russian language: a technical report. *arXiv preprint arXiv:2112.02448*, 2021.

[27] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[28] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 2021.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[30] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020.

[31] Matt J. Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution, 2016.

[32] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

[33] DeepSpeed ZeRO-3 Offload. https://www.deepspeed.ai/2021/03/07/zero3-offload.html.

[34] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.

[35] Georgii Novikov, Daniel Bershatsky, Julia Gusak, Alex Shonenkov, Denis Dimitrov, and Ivan Oseledets. Few-bit backward: Quantized gradients of activation functions for memory footprint reduction, 2022.