Capability and Applicability of Measurement Tools for AI Model's Environmental Impact

Rui Zhou, Tao Zheng, Xin Wang, Lan Wang
Orange Innovation China
Beijing, China
e-mail: {rui.zhou, tao.zheng, xin2.wang, lan.wang}@orange.com

Emilie Sirvent-Hien
Orange Innovation
Châtillon, France
e-mail: emilie.hien@orange.com

***Abstract*—More and more of Artificial Intelligence (AI) systems have been adopted by Information and Communication Technology (ICT) solutions to make effective digital transformation. In recent years environmental impact of AI systems has been investigated and methodologies have been developed to calculate their cost. In this paper, we survey, analyze, and evaluate three types of tools for counting the energy consumption/CO2 emission of AI systems. By verifying them in sets of experiments, including centralized and distributed on devices architecture, we compare ease of use of tools, simulation result vs real measurement and finally bring advice to help AI developers to take into account environmental cost of AI models with measurement tools.**

Keywords-FLOPs; PUE; PSF; TDP.

I. INTRODUCTION

The global average temperature in the past decades has increased more than 1°C compared to the pre-industrial baseline (1850-1900) [1]. Such climate change has caused more extreme weather events, rising seas, reduction of biodiversity, and negative impact to global health and safety. The global warming is a critical issue facing all mankind. Paris agreement sets a global objective for the temperature increase below 2°C. Many nations, regions, industries, companies, and individuals have put in place climate actions on their agendas. The current rise is more rapid primarily as the result of greenhouse gas emissions by burning fossil fuels for energy used in industry, transport, building, etc., which took 73.2% of global greenhouse gas emissions according to the data obtained in the year 2016 [2].

Using Communication Technology (ICT) solutions in these sectors can have a calculated potential to reduce greenhouse gas emissions by up to 15% [3]. Their own contribution to greenhouse gas emissions should not be ignored. Our focus is on Artificial Intelligence (AI) as more and more of them have been adopted by ICT solutions to make the effective digital transformation. It is expected that the Artificial Intelligence industry will be worth $190 billion by 2025, with global spending in AI systems reaching $57 billion by 2021 already [4]. The demand for computing these AI systems is growing exponentially. The intensive computation nowadays not only takes place in datacenters but also in a huge amount of edge devices closed to consumers and enterprises to support AI applications to process big data with low latency and large bandwidth requirements.

Scientists and researchers have started to investigate the environmental impact of AI systems in recent years and have developed methodologies to calculate their costs. For example, Schwartz and Doge et al. [5] refer to the AI systems that focus on accuracy without any estimation on the economic, environmental, or social cost of reaching the claimed accuracy as Red AI. They have proposed a simplified estimation of the cost of an AI which grows linearly with the cost of processing a single example, the size of the training dataset, and the number of hyperparameter experiments. OpenAI [6] has pointed out that among the three factors driving the advance of AI: algorithmic innovation, data, and the amount of computing available for training, computing is unusually quantifiable. The number of FLoating point of OPerations (FLOPs) (adds and multiplies) in the described architecture per training example can be counted. If there is not enough information to directly count FLOPs, Graphics Processing Unit (GPU) training time, how many GPUs used and a reasonable guess at GPU utilization can be used to estimate the number of operations performed. Strubell et al. [7] have quantified the computational and environmental cost of training several popular Natural Language Processing (NLP) models. The total power required at a given instance during training is related to the average Power Usage Effectiveness (PUE) for datacenter multiplying the sum of average power draw from all Central Processing Unit (CPU) sockets, average power draw from all Dynamic Random Access Memory (DRAM) (main memory) sockets, and average power draw of a GPU during training multiplied by the number of GPU. The greenhouse gas emission equivalent per kilowatt-hour is then calculated based on data provided by the U.S. Environmental Protection Agency (EPA). Google research team R. So et al. [8] have evaluated Large Transformer models which have been central to recent advances in NLP and have developed a more efficient variant with a smaller training cost than the original transformer and other variants for auto-regressive language modelling. Patterson et al. [9] have calculated the energy use and carbon footprint of several recent large models and found that large but sparsely activated Deep Neural Networks (DNNs) can consume less energy than the large, dense DNNs without sacrificing accuracy despite using as many or even more parameters. The geographic location and specific data center infrastructure matters to

 ISBN: 978-1-68558-050-6

reduce the greenhouse gas emission equivalent of Machine Learning (ML) workload. Patterson et al. [10] have shared interesting information that the inference represents about 3/5 of total ML usage at Google across three years, due to the many billion-user services that use ML. The combined emissions of training and serving need to be minimized. Lacoste et al. [11] have developed a tool called "Machine Learning Emissions Calculator" for ML community to approximate the environmental impact of training ML models. Ligozat and Luccioni [12] have proposed a practical guide to quantifying carbon emissions for ML researchers and practitioners. To analyse the carbon impact of ML, besides the ML model emissions of greenhouse gas due to the power consumption incurred by the equipment at the running time, other dimensions of model impact should be considered such as model preparation overhead, static consumption of the equipment, infrastructure, as well as the overall life cycle analysis of the equipment. The authors also have suggested the most important steps to take for practitioners and institutions for example, as an institution, deploying computation in low-carbon regions, providing institutional tools for tracking emissions, capping computational usage, carrying out awareness campaigns, and facilitating institutional offsets.

Standardizations have begun to tackle the subject, and for example, a new work item proposal has been under discussion in the Joint Technical Committee on Artificial Intelligence (JTC 21) of European Committee for Standardization (CEN) and European Committee for Electrotechnical Standardization (CENELEC). The new work item is about "Green and sustainable AI" which will establish a framework for quantification of the environmental impact of AI and its long-term sustainability and encourage AI developers and users to improve the efficiency of AI use [13]. The "CEN/CENELEC standardization landscape for energy management and environmental viability of green datacenters [14]" defines Key Performance Indicators (KPIs) that address energy and environmental control. However, these KPIs are focused on datacenters and currently do not address the distributed or IoT energy and environmental control. These aspects should be addressed in the AI standards also including sustainable development goals [15].

As discussed before, quantifying greenhouse emissions for any AI system is very important and several simplified and applicable methods have been developed in recent studies. Some open-source tools are available applying and integrating these methods. These tools have been classified into three major categories: priori measurement tools which usually calculate operational points in training and inference; on-the-fly measurement tools which measure power consumption, etc., when an AI system is running on hardware; posteriori measurement tools refer to these tools to approximate greenhouse gas emissions for a given computation. In our experiments, we deep dive into PowerAPI [16], PyJoules [17], and other open-source tools, such as Keras-flops [18], Torchstat [19], torchsummaryX [20], Flops-counter [21], JouleHunter [22], Jtop [23], CarbonAI [24], MLCO2 [11] and Green Algorithm [25], in order to have first-hand experience and to understand their capabilities and limitations.

Most of the recent research work focuses on the environmental impact of ML models at the training stage. As [10] mentioned, the energy consumed at the inference stage was more than the energy consumed at training for a given few years. So our idea is to set up a framework to evaluate the AI systems at both the training and inference stages. Considering large scale of AI systems is running on the edge side, we set up a heterogenous edge platform with various device types where we can use the proper orchestration tool that we have evaluated to deploy ML model on these different edge devices which somehow can simulate distributed AI applications deployed in real scenarios. Both X86-based and ARM-based hardware are used in the platform. We have designed methodologies to perform sets of experiments to measure the power consumption of the ML model incurred on hardware for the training stage and inference stage. Unlike a data center, the power consumed by these edge devices is mainly coming from CPU/GPU computation and memory usage with very limited overhead for cooling components if they have any. Even so, we have measured the static power consumption of edge devices to get a more precise measurement of AI model power consumption by subtracting the static power consumption. We also select various types of ML models for one AI use case and compare the power consumption of these ML models when they are running on edge devices in addition to their performance.

Our objectives for the studies are to verify the measurement tools and improve them; to obtain greenhouse gas emissions of various ML models; to benchmark performance vs environmental impact; and to develop greener ML models. These experimental results can provide useful information and recommendations for organizations to build an institutional toolbox to track greenhouse gas emissions of AI systems and offer responsible AI applications to our customers. The scope and key point of our analysis are the comparison of training and inference energy/CO2 consumptions among different AI models solving the same problem, not the comparison between the centralized server and edge devices.

The organization of this paper is as follows. Section 1 is an introduction; Sections 2 to 4 analyze the three categories of measurement tools respectively; Section 5 provides our experiments and results analysis, and Section 6 gives the conclusion.

II. PRIORI MEASUREMENT TOOLS ANALYSIS

To evaluate the ML model's power consumption by comparing the model's calculation amount, the priori measurement tools are employed. They are used to evaluate AI models and algorithms through computing the flops/mult-adds/other parameters.

There are two usages of priori measurement tools:

- as an inline module to measure the AI model, for example, first install as a python module, and then call some tools' functions in the model source

program to get the model's related computing information.

- as a Command Line Interface (CLI) tool to handle the model's source program, for example, first install the tool, and execute the tool to process the model source program to get the model's related computing information.

Because priori measurement tools handle the source code of AI programs, they always process one specific framework and support a subset of types of layers.

For testing priori measurement tools, a test environment was built, and related AI frameworks and some candidate priori measurement tools were installed first; then, constructed some demo AI models with/without special layers based on relevant frameworks; finally, computed and compared these demo AI models' Flops/Mult-Adds and other related measurements using candidate priori measurement tools and evaluated them through these computed results.

We launched five tests for four priori measurement tools: keras-flops, torchstat, torchsummaryX and flops-counter.

Keras-flops as a python module can calculate the FLOPs of neural network architecture written in Tensorflow. Test1 verifies keras-flops' support for Conv2dTranspose layer. In this test, we constructed a model including Conv2dTranspose layer to test this tool's capability with two python programs (with/without Conv2dTranspose Layer). The difference value of flops means that Conv2dTranspose layer is supported by keras-flops. Test2 verifies keras-flops' support for Conv3dTranspose layer. According to the supporting table, Conv3dTranspose layer is not supported by keras-flops. In this test, we constructed a model including Conv3dTranspose layer and test the tool's capability with two python programs (with/without Conv3dTranspose layer). The same value of flops means that Conv3dTranspose layer is not supported by keras-flops.

Torchstat is a lightweight neural network analyzer based on PyTorch. Its usage is as a python module to measure an AI model or as a CLI tool to handle a python program including an AI model. Test3 verifies Torchstat's support for Conv2d layer and ConvTranspose2d layer. In this test, we constructed Convolutional Neural Network (CNN) models including Conv2d layer and ConvTranspose2d layer to test the tool's capability.

TorchsummaryX is also a tool based on the Pytorch framework. This tool can handle Recurrent Neural Network (RNN), Recursive Neural Network, or models with multiple inputs. In the test4, we constructed two models with Conv2d layer and ConvTranspose2d layer respectively. We can find Mult-Adds remains unchanged. It means that Convtranspose2d layer is supported for Mult-Adds by torchsummaryX.

Flops-counter is based on the PyTorch framework and designed to compute the theoretical number of multiply-add operations in CNNs. It can also compute the number of parameters and print the per-layer computational cost of a given network. In the test5, we constructed two models with Conv2d layer and ConvTranspose2d layer respectively. We can find the computational complexity (i.e., number of multiply-add operations) remains unchanged. It means that Convtranspose2d layer is supported for Mult-Adds by torchsummaryX.

Torchstat, torchsymmaryX, and flops-counter are all based on the PyTorch framework, but their outputs are different. Torchstat outputs numbers of parameters, amount of Multiply+Adds, number of flops, and memory usage. torchsummaryX and flops-counter just provide numbers of parameters and amount of multiply+adds. According to some feedback from Internet, the results of torchstat's MAdd and FLOPs are wrong, which should be swapped. The summary of the four tools is shown in the following Table I.

TABLE I. SUMMARY OF FOUR PRIORI MEASUREMENT TOOLS

priori measurement tools	support framework	outputs
keras-flops	Tensorflow	FLOPs
torchsummaryX	PyTorch	FLOPs, Multi-Add, memory, total params
torchstat	PyTorch	Multi-Add, total params
flops-counter	PyTorch	Multi-Add, total params

Through our five tests, we can find that the effectiveness of priori measurement tools relies on their detailed implementation. The application of priori measurement tools is limited. The tools we tested just support one special framework (Tensorflow or PyTorch) and a subset of types of model layers. In practice, most AI models usually include some specific layers (e.g., 3D ConvTranspose layer) which cannot be calculated by our tested priori measuring tools.

III. ON-THE-FLY MEASUREMENT TOOLS ANALYSIS

The most direct and precise way to measure an AI program's power consumption is to measure it in real-time while the process is going on. We named this type of tool the "on-the-fly tool". For this purpose, we have carried out the research and study of relevant tools and later carried out the comparison test and verification of their real use situation.

After preliminary selection, we choose the following three measurement tools as candidates, they are PowerAPI series (JouleHunter, PyJoules), CarbonAI, and Jtop. They have their own methods and application scenarios, and following their official instructions and guidance documents, we conducted a series of tests and applications on them.

The first one is the PowerAPI, the goal of this project is to provide a set of tools to go forward greener computing, the idea is to provide software-defined power meters to measure the power consumption of the program, the core of this project is the PowerAPI toolkit for building such power meters [29].

PowerAPI is a middleware toolkit for building software-defined power meters. Software-defined power meters are configurable software libraries that can estimate the power consumption of software in real-time. A power meter built on PowerAPI normally has two components -- the sensor and the formula. The sensor is also a software, which worked like the physical world sensor, queries the hardware's (host machine) data, and collects raw data correlated with the

 ISBN: 978-1-68558-050-6

power consumption of the software. All data will be stored in an external database to make the data available to the formula. For the other component, the formula is a computational module that uses the collected data to determine power consumption. Both are connected by a database that is used to transfer information. The global architecture of a power meter is represented in the Figure 1 below [26].

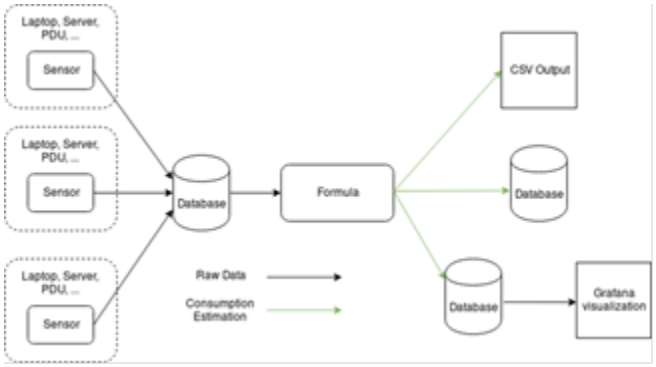

Figure 1. The global architecture of a power meter

For convenience and quick use, PowerAPI has provided several useful components. As Hwps-Sensor (Hardware Performance Counter), is a tool using the Running Average Power Limit (RAPL) technology to monitor the Intel CPU performance counter and power consumption of the CPU. Also, some matched formulas like “SmartWatts Formula” used for physical Linux machine, “VirtualWatts” used for a virtue machine, etc.

PowerAPI also packages up (with sensor and formula) a set of ready-to-use tools for diverse needs. Here Joule Hunter and PyJoules are the two we selected and used for our research.

JouleHunter runs on Linux machines with Intel RAPL support. This technology has been available since the Sandy Bridge generation [27]. JouleHunter can show what part of your program code is consuming considerable amounts of energy in detail. JouleHunter works similarly to pyinstrument [28], as it forked the repo and replaced time measuring with energy measuring. This tool can be easily installed and used with one or two command line(s), two key components of hardware the intel CPU and ram’s power consumption can be printed out. However, from its official documentation and real test we can see that JouleHunter this tool has its limitation, such as it only worked with Linux OS and no calculation for GPU power consumption.

Another software toolkit from PowerAPI is the PyJoules, which can be used to measure the energy footprint of a host machine along with the execution of a piece of Python code. Except for Intel CPU socket package and RAM (only for Intel server architectures), it also can monitor the energy consumed by the GPU of the host machine, supporting both for Intel integrated GPU (for client architectures) and Nvidia GPU (Uses the Nvidia "Nvidia Management Library" technology to measure the power consumption of Nvidia devices. The energy measurement Application Programming Interface (API) is only available on Nvidia GPU with Volta architecture 2018) [29]. PyJoules can only work with AI program coding on Python, and it should be installed and imported as a function into the target main project python file. It will report the total power consumption during the code is running. Its results contain not only the target project’s power consumption, thus including the OS and other applications running at the same time if have. That means it calculates the global power consumption of all the processes running on the machine during this period. With PyJoules, to get the closest measure to the real power consumption of the measured program, we need to try to eliminate any extra programs (such as graphical interface, background running task, etc.) that may alter the power consumption of the host machine and keep only the code under measurement. Same as JouleHunter, PyJoules currently can only work on GNU/Linux, and does not support on Windows and MacOS.

CarbonAI is another project which aims to raise AI the developers’ awareness of the AI’s carbon footprint. Firstly, like PyJoules it provides a python package that allows developer to monitor power consumption. Then based on the measurement results, CarbonAI will do a transition between power consumption and CO2 emissions, to provide a more intuitive understanding of how much our AI development is doing to the environment. For example, training an AI model for 100 rounds is the equivalent of driving from Paris to Marseille. Also, the power consumption results of CarbonAI are given as a CSV file, which includes most key devices of a host machine, like CPU, GPU, and RAM, but also the name of the country where the package was used (based on the IP or what the user set). So, the amount of CO2 emitted by the usage will be depends on the country and the energy mix used by the country to produce electricity. For combability, a different form that PyJoules only support Linux OS, CarbonAI package is compatible with most platforms (Linux, Windows, and MacOS) with the varying installation process.

At present, apart from x86 architecture servers and devices, ARM-based devices are also widely used in various fields of AI. However previous tree tools only work well on x86 hardware platforms, all of them will get several issues or bugs. For ARM platforms, Nvidia provides their official tool Jtop for the Jetson series, a platform designed for AI development and use cases. Jtop is one of jetson-stats, a package for monitoring and controlling NVIDIA Jetson (Xavier NX, Nano, AGX Xavier, TX1, TX2) Works with all NVIDIA Jetson ecosystems. Jtop can be run independently and show the real-time usage data of CPU, GPU, and RAM and also the actual frequency of the hardware. With its built-in graph user interface, we can easily read the results, but only the immediate frequency, so for the final power consumption we need manually calculate the Total power consumption using w=p*t, and “t” is the duration of the target AI program. Compared to other tools, although the result of power consumption cannot be directly obtained, Jtop can provide the usage rate of each device.

All those tools are not very difficult to install and use, some of them can be installed with several command lines,

like JouleHunter, CarbonAI, and Jtop; and for PyJoules, we can add it into our application code just like a function. For compatibility, except CarbonAI supports Linux, Windows and MacOS (we only use it on Linux machines), other tools currently can only be used on Linux. Most tools currently only support Intel CPU and RAM, for GPU's power consumption, we need external components or 3rd part tools. For the programming language, most tools are built up as a python package, so they only worked with AI apps coded with python.

For the usage, the reported power consumption is not only the power consumption of the code you are running. This includes the global power consumption of all the processes running on the machine during this period, thus including the OS and other applications. So, we need to eliminate any extra programs and get the value of the devices when idling as the base level if possible. This will give the closest measure to the real power consumption of the measured code.

IV. POSTERIORI MEASUREMENT TOOLS ANALYSIS

AI researchers also proposed to estimate carbon emissions of the AI computation by posteriori tools, i.e., ML CO2 Impact and Green Algorithms. The key methodology of the tools is to estimate the power consumption after the computation process and achieve the carbon emissions from power consumption and related carbon intensity.

As shown in Table II, ML CO2 Impact tool is designed to estimate the carbon emissions produced by training ML models. The inputs include the geographical zone of the server, the type of GPU, and the training time, and the output is the approximate amount of CO2e. The inventors collected available public data for the computation including the Thermal Design Power (TDP) of the hardware, the location of the hardware, and the related carbon intensity (CO2e emissions per kWh).

TABLE II. ML CO2 IMPACT AND GREEN ALGORITHMS

	ML CO2 Impact	Green Algorithms
Energy consumption	runtime * power draw for GPU	runtime * (power draw for cores * usage + power draw for memory) * PUE * PSF
Hardware type	Mainly GPU type	GPU, CPU, CPU/GPU co-existing case, number of cores, memory
Usage factor	100% by default	100% by default and configurable
Other factors	/	Power Usage Effectiveness: the extra energy needed to operate the data center (cooling, lighting , etc.) Pragmatic Scaling Factor: multiple identical runs (e.g. for testing or optimization)

Green Algorithms tool aims to estimate the carbon footprint of any computational task. Compared with ML CO2 Impact tool, it requires extra inputs of memory size, real usage factor of the processing core, PUE, and Pragmatic Scaling Factor (PSF).

Different from the on-the-fly measurement tools, the power consumption model of both tools uses TDP and runtime to achieve the power consumption, which means in the calculation the usage of cores is 100% by default. Green Algorithms tool allows to configure the real usage of cores and takes more quantifiable elements into consideration, i.e., memory power, PUE, and PSF, allowing users to estimate the power consumption more flexibly.

Considering carbon intensity, it is known that fossil fuels have the highest carbon footprints, for example, coal emits 820g of CO2e per kWh of electricity produced [30], while electricity generated by wind, solar, hydro, or nuclear power emits lower amounts of carbon footprints, i.e., 12g CO2e /kWh for wind, and 27~48g CO2e /kWh for all types of solar. In different countries and regions, even different electric power companies, the energy structure differs from others, and various energy sources would be used to generate electricity. The location matters as all servers are connected to local grids and they will have different amounts of CO2e emissions when consuming or generating the same amount of electricity, for example, 174g CO2e emission per kWh of electricity for France and 741g CO2e /kWh for Germany (at 12:00 PM on December 1, 2022) [31].

Both tools refer to public data for carbon intensity. They provide data reference of data centers including Google Cloud Platform, Amazon Web Services, and Azure. However, we can see clearly that there is no unified data source, location scope, and effective time due to differences in the data sources of the two tools.

From the preliminary analysis of the above two tools, we know that when evaluating carbon emission impact, both power consumption and carbon intensity should be considered. Parameters like hardware type, PUE, the usage of the core, and memory will contribute to the energy consumption. For carbon intensity, the location matters because of the different energy mix in different countries and regions, but there are various data sources that may provide quite different location scopes and effective time. Also, we notice that the goal of these tools is to make people aware of the carbon emission impact, to provide a quick tool to evaluate the carbon emission during machine learning work and to recommend carbon reduction actions like selecting the cloud provider or server location wisely, buying carbon offsets, choosing clean energy, and improving AI algorithms to be green.

V. EVALUATION EXPERIMENTS

Our objectives are to verify the measurement tools and have some comments and suggestions proposed for measurement tools for AI models through our experiments.

We have developed a systematic methodology to carry out our experiments. First, a cloud-edge platform was set up with heterogenous hardware either X86-based, or ARM-based. These hardware devices have similar levels of computation capabilities to commercial end devices such as LiveBox, controllers on vehicles, etc. Then, we selected an AI application, in our case, we choose Person Re-Identification (Re-ID) which is the task of associating the same person taken from different cameras or from the same

 ISBN: 978-1-68558-050-6

camera on different occasions [32]. Person Re-ID have wide usage in smart building and smart city scenario. Many Person Re-ID open-source models are accessible using various AI architectures, such as CNN, Transformer, or Long Short Term Memory (LSTM). Based on criteria, such as performance, release date, accessibility, etc., we have selected several Person Re-ID models with different AI model architectures. After that, we measured their power consumption during the training stage and inference stage when they are running on various types of hardware in rounds of experiments.

We build a benchmark to compare the results of the on-the-fly and posteriori measurement tools. In the first experiment, the power consumption of Fast-ReID (CNN) model is measured by processing AI inference on a 500s video of a single person. The results of PyJoules and Jtop tools are selected as a baseline of the real-time measured power consumption. MLCO2 Impact and Green Algorithms tools are used to estimate the power consumption afterward, respectively. As is shown in Table III, three types of hardware have been evaluated: a server with GeForce GTX 1080 Ti, Intel Xeon E5-2678 v3 and a memory of 64GB, and two edge devices – one with Intel i7-8559U and a memory of 16GB, and the other with NVIDIA Jetson AGX Xavier, ARMv8 Processor rev 0 (v8l) and a memory of 32GB. For Green Algorithms tool, there is a default CPU usage of 100% and a configurable CPU usage that can be estimated based on the observation of the AI processing experiment. The PUE used in the calculation is 1 because we use local private infrastructure instead of cloud services and ignore the power consumption of cooling or lighting. The memory power draw only depends on the size of memory available (0.3725 W per GB).

In the second experiment, as is shown in Table IV, four different Re-ID models are evaluated at the training stage: Fast-ReID (CNN), st-ReID (CNN), DeepPerson (LSTM), and Trans-ReID (Transformer). Since both GPU and CPU cores will be used in the AI training process, power consumption should be considered for both.

VI. CONCLUSION

We have been carrying out series of experiments to verify the measurement tools. For different types of measurement tools, we found that:

The effectiveness of priori measurement tools relies on their detailed implementation. The application of priori measurement tools is limited. The tools just support one special framework and a subset of types of model layers.

The on-the-fly tools can be used during the processes of AI programs; however, they are limited. PyJoules or JouleHunter can be used to get power consumption (CPU, GPU, RAM) of large AI programs on different x86 architectures devices, while for architectures ARM devices, only Jtop supported. Ideally, it is better to develop and use the same cross-platform tool. However, the comparison of experimental and estimated results shows that the error of the on-the-fly measurement tools is acceptable.

The posteriori measurement tools can be used for power consumption estimation after the AI processing by knowing the runtime and the parameters of hardware (CPU, GPU, memory, etc.). For resource-constrained edge devices, the resources usually tend to be nearly full of use, and the tools with a default configuration are able to make a quick estimation of the power consumption. For servers that have more resources and stronger processing capabilities, if extra information can be given, for example, the real usage of the cores, the Green Algorithms tool will be optimized to make close estimations to real-time measured power consumption. Both tools can provide different CO2e emissions due to different locations where the AI computation is processed. The researchers aim to remind people to carefully select the cloud providers and locations for AI services when carbon impacts should be taken into consideration.

We have selected an AI use case: Re-ID which can be realized by various types of AI architecture: CNN, LSTM, and Transformer. Once the specific AI model for each type is selected, the power consumption of the selected AI models is measured during the training and inference stages when they are running on different edge devices. The experimental results show that the total training power consumption of the AI model is determined by the training algorithm and training time. Training power consumption is in proportion to the training time in general. With the measurement tool, it can quantify the power consumption to make a more accurate assessment for different AI models. The power consumption of AI applications is positively correlated with the complexity of application scenarios when the hardware capability allows.

Our plan includes continuously investigating and improving the measurement tools and verifying them in experiments. With these tools, researchers and scientists may be able to design more power-efficient AI models without sacrificing model performance.

REFERENCES

[1] WiKi climate change. [Online]. Available from: https://en.wikipedia.org/wiki/Climate_change/ [retrieved: 05, 2023].

[2] H. Ritchie, “Sector by sector: where do global greenhouse gas emissions come from?”. [Online]. Available from: https://ourworldindata.org/ghg-emissions-by-sector/ [retrieved: 05, 2023].

[3] Ericsson, “ICT’s potential to reduce greenhouse gas emissions in 2030”. [Online]. Available from: https://www.ericsson.com/en/reports-and-papers/research-papers/exploring-the-effects-of-ict-solutions-on-ghg-emissions-in-2030/ [retrieved: 05, 2023].

[4] Matleena, “Amazing AI Statistics (2022): Stunning Growth of AI”. [Online]. Available from: https://zyro.com/blog/ai-statistic/ [retrieved: 05, 2023].

[5] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green AI”, Communications of the ACM, vol. 63, No. 12, pp. 54-63, Dec. 2020.

[6] OpenAI, http://openai.com/ [retrieved: 05 2023].

[7] E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP”, 57th Annual Meeting of the Association for Computational Linguistics (ACL). Florence, Italy. Jul. 2019, doi: 10.18653/v1/P19-1355.

[8] D. R. So, et al., “Primer: Searching for Efficient Transformers for Language”, 35th Conference on Neural Information

Processing Systems (NeurIPS 2021), virtual, 2021, doi: 10.48550/arXiv.2109.08668.

[9] D. Patterson, et al., "Carbon Emissions and Large Neural Network Training", 2021, doi: 10.48550/arXiv.2104.10350.

[10] D. Patterson , et al., "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink", Computer, vol. 55, pp. 18-28, Jul 2022.

[11] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the Carbon Emissions of Machine Learning", 2019, doi: 10.48550/arXiv.1910.09700.

[12] A. Ligozat and S. Luccioni, "A Practical Guide to Quantifying Carbon Emissions for Machine Learning researchers and practitioners", [Online]. Available from: https://hal.archives-ouvertes.fr/hal-03376391/document/ [retrieved: 05, 2023].

[13] CEN-CENELEC JTC 21 "Artificial Intelligence", [Online]. Available from: https://www.cencenelec.eu/areas-of-work/cen-cenelec-topics/artificial-intelligence/ [retrieved: 05 2023].

[14] CEN/CENELEC, "Standardization landscape for energy management and environmental viability of green data centres", [Online]. Available from: ftp://ftp.cencenelec.eu/EN/EuropeanStandardization/HotTopics/ICT/GreenDataCentres/GDC_report_summary.pdf [retrieved: 05 2023].

[15] Walshe, R., Casey, K., Kernan, J., Fitzpatrick, D., "AI and big data standardization: Contributing to United Nations sustainable development goals", Journal of ICT Standardization, pp. 77–106, 2022.

[16] PowerAPI, http://www.powerapi.org/ [retrieved: 05 2023].

[17] PyJoules, https://github.com/powerapi-ng/pyJoules/ [retrieved: 05 2023].

[18] Keras-flops, https://github.com/tokusumi/keras-flops/ [retrieved: 05 2023].

[19] Torchstat, https://github.com/Swall0w/torchstat/ [retrieved: 05 2023].

[20] torchsummaryX, https://github.com/nmhkahn/torchsummaryX/ [retrieved: 05 2023].

[21] flops-counter, https://github.com/sovrasov/flops-counter.pytorch/ [retrieved: 05 2023].

[22] JouleHunter, https://github.com/powerapi-ng/joulehunter/ [retrieved: 05 2023].

[23] Jtop, https://pypi.org/project/jetson-stats/ [retrieved: 05 2023].

[24] CarbonAI, https://github.com/Capgemini-Invent-France/CarbonAI/ [retrieved: 05 2023].

[25] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: quantifying the carbon footprint of computation". Advanced science, vol 8, issue 12, 2021, doi: 10.48550/arXiv.2007.07610.

[26] Global architecture picture, https://raw.githubusercontent.com/powerapi-ng/powerapi ng.github.io/master/images/powerAPI_archi.png/ [retrieved: 05 2023].

[27] Sandy Bridge generation, https://fr.wikipedia.org/wiki/Intel#Historique_des_microproce sseurs_produits/ [retrieved: 05 2023].

[28] Pyinstrument, https://pyinstrument.readthedocs.io/ [retrieved: 05 2023].

[29] Volta architecture, https://en.wikipedia.org/wiki/Volta_(microarchitecture)/ [retrieved: 05 2023].

[30] https://www.world-nuclear.org/ [retrieved: 05 2023].

[31] https://app.electricitymaps.com/map/ [retrieved: 05 2023].

[32] Person-re-Identification, https://paperswithcode.com/task/person-re-identification [retrieved: 05 2023].

TABLE III. FAST-REID, SINGLE PERSON, RUNNING TIME = 500S, AI INFERENCE

Measurement tools	GPU/CPU type	Power consumption				Carbon emissions[a]
		CPU (W)	*Usage*	*Memory (GB)*	*Total (Wh)*	*CO2e(mg)*
1 On the fly - PyJoules	Server: GPU: GeForce GTX 1080 Ti CPU: Intel Xeon E5-2678 v3 Memory: 64GB				7.2	
2 A posteriori - MLCO2 Impact		120	100%		16.67	633.46
3 A posteriori – Green Algorithms		120	30%/100%	64	8.31/19.98	426.18/ 1002
1 On the fly - PyJoules	Intel machine II: CPU: Intel i7-8559U Memory: 16GB				4.1	
2 A posteriori - MLCO2 Impact		28	100%		3.89	147.82
3 A posteriori – Green Algorithms		28	70%/100%	16	3.55/4.72	182.04/ 241.86
1 On the fly - Jtop	ARM: NVDIA Jetson AGX Xavier CPU: ARMv8 Processor rev 0 (v8l) Memory: 32GB				3.8	
2 A posteriori - MLCO2 Impact		30	100%		4.17	158.46
3 A posteriori – Green Algorithms		30	70%/100%	32	4.57/5.82	234.45/ 298.55

a. The reference location is Europe, France.

TABLE IV. FAST-REID, ST-REID, DEEPPERSON, AND TRANS-REID, AI TRAINING

Measurement tools	Test ML model	Running time(s)	Power consumption						Carbon emissions[a]
			GPU (W)	*Usage*	*CPU (W)*	*Usage*	*Memory (GB)*	*Total (Wh)*	*CO2e(g)*
1 On the fly - PyJoules	Fast-ReID, CNN	4625						125.06	
2 A posteriori - MLCO2 Impact			120	100%	45	100%		211.98	8.06
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	128.16/242.61	7.08/12.44
1 On the fly - PyJoules	st-ReID,	7988						212.26	

2 A posteriori - MLCO2 Impact	CNN		120	100%	45	100%		366.12	13.91
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	238.62/419.01	12.24/21.49
1 On the fly - PyJoules	DeepPerson, LSTM	8326						201.74	
2 A posteriori - MLCO2 Impact			120	100%	45	100%		381.61	30.35
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	248.72/436.74	26.69/46.87
1 On the fly - PyJoules	Trans-ReID, Transformer	17426						451.7	
2 A posteriori - MLCO2 Impact			120	100%	45	100%		798.69	30.35
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	520.55/914.09	26.69g/46.87

a. The reference location is Europe, France.

 ISBN: 978-1-68558-050-6