THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

une méthodologie polyvalente pour évaluer la consommation électrique et l'empreinte environnementale de l'entraînement de l'apprentissage machine : des supercalculateurs aux équipements embarqués

A Versatile Methodology for Assessing the Electricity Consumption and Environmental Footprint of Machine Learning Training: from Supercomputers to Edge Devices

Présentée par :

Mathilde JAY

Direction de thèse :

Denis TRYSTRAM PROFESSEUR DES UNIVERSITES, GRENOBLE INP	Directeur de thèse
Laurent LEFEVRE ENS de Lyon	Co-directeur de thèse

Rapporteurs :

Aurélie BUGEAU
PROFESSEURE DES UNIVERSITES, Université de Bordeaux
Anne-Laure LIGOZAT
PROFESSEURE DES UNIVERSITES, Université Paris Saclay

Thèse soutenue publiquement le **15 octobre 2024**, devant le jury composé de :

Denis TRYSTRAM, PROFESSEUR DES UNIVERSITES, Grenoble INP - Université de Grenoble Alpes	Directeur de thèse
Laurent LEFèVRE, CHARGE DE RECHERCHE HDR, Ecole Normale Supérieure de Lyon	Co-directeur de thèse
Aurélie BUGEAU, PROFESSEURE DES UNIVERSITES, Université de Bordeaux	Rapporteure
Anne-Laure LIGOZAT, PROFESSEURE DES UNIVERSITES, Université Paris Saclay	Rapporteure
Emma STRUBELL, ASSISTANT PROFESSOR, Carnegie Mellon University	Examinatrice
Sylvain BOUVERET, MAITRE DE CONFERENCES, Grenoble INP - Université de Grenoble Alpes	Examinateur
Claude LEPAPE, INGENIEUR DE RECHERCHE, Schneider Electric	Examinateur
Claudia RONCANCIO, PROFESSEURE DES UNIVERSITES, Grenoble INP - Université Grenoble Alpes	Examinatrice

Invités :

Bruno MONNET
INGENIEUR, Hewlett Packard Enterprise

Abstract

The number of Artificial Intelligence applications being developed and deployed is continually increasing. The effects of these activities on the biosphere, particularly on climate change, have attracted attention since 2019, but assessment methodologies still require improvement. More advanced evaluation methods and a deeper understanding of these impacts are necessary to minimize the environmental impacts of artificial intelligence.

With an emphasis on the training phase, this thesis investigates how machine learning (ML) affects the environment.

First, a study is conducted to assess the electricity consumption of IT infrastructures by comparing power meters currently in use with different benchmarks and infrastructures, focusing on Graphic Processing Units (GPUs). The comparison is supported by numerous experiments and is based on classic quantitative criteria, as well as qualitative criteria such as ease of use, configurability, and documentation quality.

These findings are used to analyze the electricity required to train models selected from the MLPerf benchmark on various ML infrastructures, ranging from an edge device to a supercomputer. Fine-grained measurements and reproducible experiments offer distinct perspectives on each computing infrastructure. The proposed methodology enables an equitable comparison of the amount of electricity consumed by different facilities.

Finally, the thesis shifts toward examining the more general environmental impacts of ML, based on an estimation of the embodied impacts of ML infrastructures. These impacts are allocated to each model training, enabling a comparison with the impacts of electricity usage. While numerous ML environmental impact indicators exist, this study focuses on primary energy consumption, global warming potential, and abiotic depletion potential for minerals and metals.

In conclusion, this thesis proposes a methodology that enables a reproducible multi-criteria evaluation of the impact of machine learning training on the environment and can be applied to different ML infrastructures, thus enabling fair comparison and enlightened choices.

Resumé

Le nombre d'applications basées sur l'intelligence artificielle (IA) développées et déployées ne cesse d'augmenter. L'impact de ces activités sur la biosphère, notamment sur le dérèglement climatique, attire l'attention depuis 2019, mais les méthodes d'évaluation nécessitent encore des améliorations. Des méthodes d'évaluation plus avancées et une meilleure compréhension de ces impacts sont nécessaires pour minimiser l'impact environnemental de l'intelligence artificielle.

En mettant l'accent sur la phase d'entraînement, cette thèse étudie l'impact de l'apprentissage automatique sur l'environnement.

Dans un premier temps, une étude est menée pour évaluer la consommation électrique des infrastructures informatiques en comparant les compteurs d'électricité actuellement utilisés, en se concentrant sur les unités de traitement graphique (GPU). La comparaison est étayée par de nombreuses expériences et repose sur des critères quantitatifs classiques, ainsi que sur des critères qualitatifs tels que la facilité d'utilisation, la configurabilité et la qualité de la documentation.

Ces résultats sont utilisés pour analyser l'électricité nécessaire à l'entraînement de modèles sélectionnés à partir du benchmark MLPerf sur différentes infrastructures d'apprentissage automatique, allant d'un appareil embarqué à un supercalculateur. Des mesures fines et des expériences reproductibles offrent des perspectives distinctes sur chaque infrastructure. La méthodologie proposée permet une comparaison équitable de la quantité d'électricité consommée par différentes installations.

Enfin, la thèse s'oriente vers l'évaluation des impacts environnementaux, en se basant sur une estimation des impacts liés à l'extraction des matériaux, à la fabrication, au transport et à la fin de vie de chaque composants des infrastructures de calcul. Ces impacts sont répartis sur chaque entraînement de modèle, permettant une comparaison avec les impacts de la consommation d'électricité. Si de nombreux indicateurs d'impact environnemental de l'apprentissage existent, cette étude se concentre sur la consommation d'énergie primaire, le potentiel de réchauffement climatique et le potentiel d'épuisement abiotique des minéraux et des métaux.

En conclusion, cette thèse propose une méthodologie permettant une évaluation multi-critères reproductible de l'impact de l'entraînement du machine learning sur l'environnement et pouvant être appliquée à différentes infrastructures spécialisées pour l'apprentissage, permettant ainsi une comparaison équitable et des choix éclairés.

Contents

Glossary

ADP Abiotic Depletion Potential. 21, 72, 78

AI Artificial Intelligence. 1, XVI

CNN Convolutional Neural Network. 26, XVIII, XIX

FL Federated Learning. 10

FLOP Floating Point Operation. 7, XVIII

FLOP/s Floating Point Operation per Seconds. 8

FU Functional Unit. 19, 72, 79

GPT Generative Pre-trained Transformer. 27

GPU Graphic Processing Unit. 8, 36

GWP Global Warming Potential. 21, 72

ICT Information and Communication Technologies. 4, 6, 12, 19, XVI

IEA International Energy Agency. 6, 12

LCA Life Cycle Assessment. 13, 72

LCI Life Cycle Inventory. 19, 21

LLM Large Language Model. 22, XIX

LSTM Long Short-Term Memory. 28, XX

ML Machine Learning. 1, 4, 72, XVI

NLP Natural Language Processing. 24

NN Neural Network. XVIII

PE Primary Energy. 21, 72

PUE Power Usage Effectiveness. 12, 22, 36

ResNet Residual Neural Network. 25, XVIII

RNN Recurrent Neural Network. XVIII

TDP Thermal Design Power. 8, 10, 20, 33, 35, 57, XXV

TPU Tensor Processing Unit. 8

General Introduction

Human activities pose a significant threat to the biosphere, upon which society is utterly dependent. Addressing the colossal challenges of transforming society to make it sustainable while facing the effects of climate change has been the objective of many scientific and political international panels or United Nations conferences (COP). The Paris Agreement [UN2015] is an international treaty signed in 2015 that results from such discussions. Its goal is to limit global temperature increase to well below 2 degrees Celsius while pursuing efforts to limit the increase to 1.5 degrees. Reaching this goal requires significantly reducing greenhouse gas (GHG) emissions as soon as possible. In the same year, all the United Nations member states adopted the 2030 Agenda for Sustainable Development including 17 Sustainable Development Goals (SDGs) to fight poverty while improving health and education and tackling climate change - to only cite a few. More than global temperature, scientists have identified 9 planet boundaries that can significantly impact human society if exceeded [Rockström2009, Steffen2015]. Among them, six were more recently estimated to be reached [Richardson2023] including climate change, land system change, and freshwater change.

The 2023 report of the Intergovernmental Panel on Climate Change (IPCC) [Calvin2023] presents digital technologies as an industry that can help mitigate climate change and achieve several SDGs by improving energy management. However, it also states that gains can be counterbalanced by growth in demand for goods and services. Digital technologies can additionally be harmful to other SDGs by increasing inequalities and labor, for example. Artificial Intelligence (AI) exacerbates those effects. It has been used in many industries to increase energy efficiency and productivity but its global environmental footprint is drastically increasing and is not restricted to GHG emissions [Wu2022]. Although digital technologies and artificial intelligence can indirectly affect each planet boundaries, positively or negatively, they directly impact primary energy consumption, climate change, water usage, and rare metal depletion [Benqassem2021].

In response to those observations, the European Union (EU) now requires all large companies to report on the impact of their activities on people and the environment, as part of the European Green Deal to reduce the GHG emissions of the continent. The Corporate Sustainability Reporting Directive (CSRD) [1] entered into force at the beginning of January 2023, defining the information companies have to report. The Energy Efficiency Directives [2] includes obligations on the energy performance of digital infrastructures, for example of data centers [3].

In this context, institutes and companies need standards and tools to evaluate the footprint of their digital services to be able to report, monitor, and reduce it. This thesis propose a methodology to estimate the environmental impacts of training Machine Learning (ML) models.

[1] Directive (EU) 2022/2464 of the European Parliament and of the Council of 14 December 2022 amending Regulation (EU) No 537/2014, Directive 2004/109/EC, Directive 2006/43/EC and Directive 2013/34/EU, as regards corporate sustainability reporting (Text with EEA relevance)

[2] Directive (EU) 2023/1791 of the European Parliament and of the Council of 13 September 2023 on energy efficiency and amending Regulation (EU) 2023/955 (recast) (Text with EEA relevance)

[3] Commission Delegated Regulation (EU) 2024/1364 of 14 March 2024 on the first phase of the establishment of a common Union rating scheme for data centers

1 Introduction

Contents

This introduction presents the current figures on the environmental footprint of Information and Communication Technologies (ICT) (Section 1.1) before focusing on Machine Learning (ML). Section 1.2 is an overview of the material evolution of ML since it was first introduced in the 1950s and Section 1.3 covers the current state of the art on the evaluation of ML impacts. Finally, Section 1.4 presents the research questions and the objectives of this thesis, Section 1.5 its contributions, and Section 1.6 its content.

1.1 ICT environmental impacts

Information and Communication Technologies (ICT) regroup all devices required to access and use digital services such as social media or online shopping. They are usually divided into user interfaces, networks, and data center components. An ICT service is a service based on ICT devices such as a web search engine, a mailbox, or any website. They usually rely on all three ICT categories to deliver a service.

This section covers the environmental impacts associated with ICT, starting by listing the main devices that make up ICT, before providing basic definitions of energy and environmental impacts and their relation to ICT.

1.1.1 The materiality of ICT

Information and Communication Technologies can easily be seen as immaterial since most of the infrastructure is hidden from the user. Understanding its materiality is crucial for appreciating its environmental impacts.

User interfaces include desktop computers, smartphones, wearable devices, and gaming consoles. Most of them require batteries thus were designed to consume low power. Networks consist of fiber optic cables or copper cables but require routers, signal amplifiers and transformers, and antennas to interconnect with each other and the users. They are distributed all around the earth and are mostly shared among internet providers and users.

Data centers are massive buildings storing and processing the data we access online. They house computing nodes, networking devices, cooling systems, and power systems such as distribution units or backup batteries. Computing nodes are responsible for useful work and are more computationally and power-intensive than user devices. They are generally composed of a Central Processing Unit (CPU), Random Access Memory (RAM), storage devices, network interface cards, a motherboard connecting all those components, and a power supply unit to convert electricity into usable power. An AI-specialized compute node can additionally host up to 8 Graphic Processing Units (GPUs).

Such facilities can be broken down into three types of data centers:

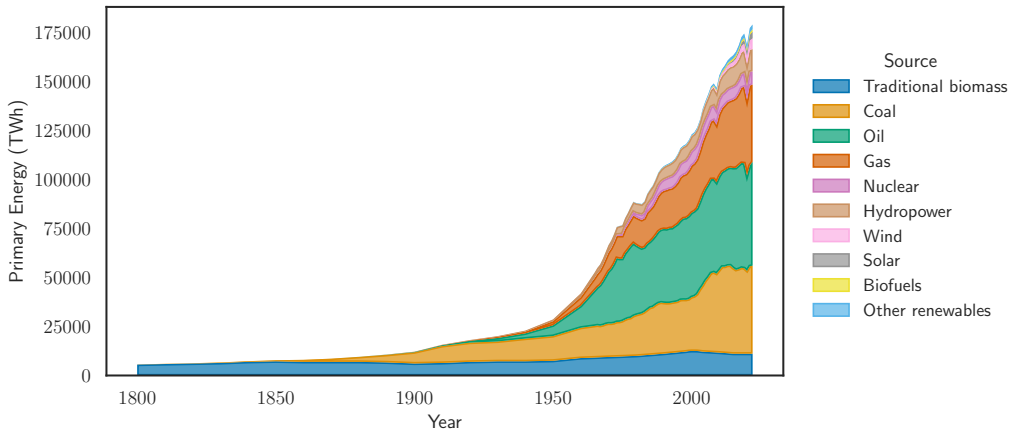
- Cloud data centers are focused on delivering content and services, like websites or streaming platforms. They rely on server nodes with one CPU and are designed to efficiently and reliably scale on demand. Their size and position might vary to minimize user latency and data movement.
- Data centers specialized in High Performance Computing (HPC) handle intensive computational tasks such as scientific research on universe simulation, climate predictions, and now ML workloads. Compute nodes require more powerful CPUs and GPUs and are often customized for specific workloads.
- Edge data centers process data close to the user to reduce the latency for real-time applications. They are usually smaller than other facilities and rely on smaller compute nodes

designed to be power efficient.

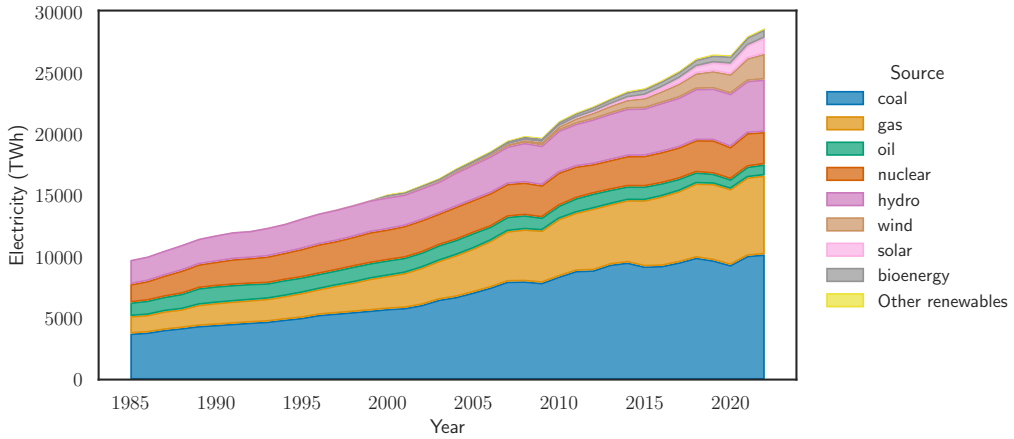
1.1.2 Focus on energy of ICT

Energy can be used to refer to various concepts with significant differences thus proper definitions are needed. Energy - as its most basic definition - is the capacity to do work. **Primary Energy** is the energy as it is available as resources before it has been transformed [Ritchie2022]. Figure 1.1a lists the main types of energy and shows the evolution of primary energy consumption by source since 1800. Fossil energies (coal and oil) represent most of it. **Electricity** is a type of energy resulting from the movement of electrons. The electrical power is expressed in Watt (W) and the electrical energy in Watt-hour (Wh). Electricity is transformed from primary energy. Figure 1.1b presents the evolution of electricity production by source from 1985. In 2022, global primary energy consumption hit a peak of 178,899 TWh, while electricity generation only reached 28,661 TWh, representing around 16%. Electricity is the main energy source for the usage phase of ICT devices.

For simplicity, in this thesis, the terms "energy" and "electricity" will interchangeably refer to "electrical energy", and "power" will refer to "electrical power".



(a) Primary energy consumption by source



(b) Electricity production by source

Figure 1.1: World statistics on energy and electricity [Ritchie2022]

The International Energy Agency (IEA) reports that Data Centers (excluding cryptocurrencies mining) consumed between 240 and 340 TWh of electricity in 2022, while the data transmission network used between 260 and 360 TWh [Agency2024]. Thus they represent at most 2.4% of the world's electricity consumption. The IEA forecasts that data centers will be one of the major contributors to electricity demand growth (+29%) between 2023 and 2026, mainly due to Artificial Intelligence. The Shift Project found that the share of digital technologies in primary energy consumption reached 5% in 2020 and could double during the next 10 years to over 9% [Ferreboeuf2021]. Differences between both estimates are explained by the large scope of the Shift Project analysis. The authors additionally include user devices such as smartphones and TVs.

Such a demand on power grids has significant impacts since production is hardly flexible and is highly influenced by geopolitical crises and weather conditions.

1.1.3 Focus on environmental impacts of ICT

All industries have direct or indirect impacts on the environment, ICT included. Freitag et al. [Freitag2021] conducted a survey of existing estimations and concludes that the share of the digital sector represented between 2.1% and 3.9% of global GHG emissions in 2021. Its significance is in its growth of around 5.5% per year between 2015 and 2019, according to the Shift Project. Freitag et al. don't provide a similar number, but forecast exponential growth, taking into account a slowdown in energy efficiency improvement and the investment in cryptocurrencies and AI. No similar estimation has been produced on other environmental indicators such as rare metal depletion.

The **operational** (or opex) impact of ICT corresponds to the impact related to the energy spent to charge or power devices, while the **embodied** (or capex) impact includes the impact of extracting raw materials, manufacturing, transporting, and dealing with the end of life of the equipment. ICT is an industry characterized by the significance of the impact of the embodied phase. Extracting raw materials and converting them into processors requires a lot of energy and water, consequently emitting carbon and consuming rare metals that can hardly be recycled.

According to the Shift Project, the embodied phase corresponded to 37% of the total impact in 2019. Another study from Gupta et al. [Gupta2022] shows that the carbon footprint of computing has shifted from opex-related activities towards hardware manufacturing. Although it is most significant for battery-fueled equipment such as smartphones and laptops for which the manufacturing cost represents between 60 and 80% of the carbon footprint, it is also true for data center hardware. Indeed, electricity efficiency has improved in the past decades while the average lifetime has remained constant, and as a consequence, the share of opex-related activities has decreased.

ICT can have several levels of impact. Its **direct or first-order impacts** are the impacts due to the IT infrastructure. It is the only category of impacts studied in this thesis. However, ICT can have other impacts. The **second-order impact** refers to the impact of its applications, such as optimizing the energy consumption of a building using an AI algorithm that might compensate for its direct impacts. A **third-order impact** category exists and corresponds to the systemic impacts that the ICT service can have on society [Ligozat2022]. The rebound effect is an example of third-order impact: an improvement in the energy efficiency of a technology can lead to an increase in its total energy consumption through behavior change, as it was stated in the 2023 IPCC report.

Machine Learning is a sub-field of ICT and its recent popularity combined with its growing demand in computations has been contributing to the increasing impacts of IT infrastructures.

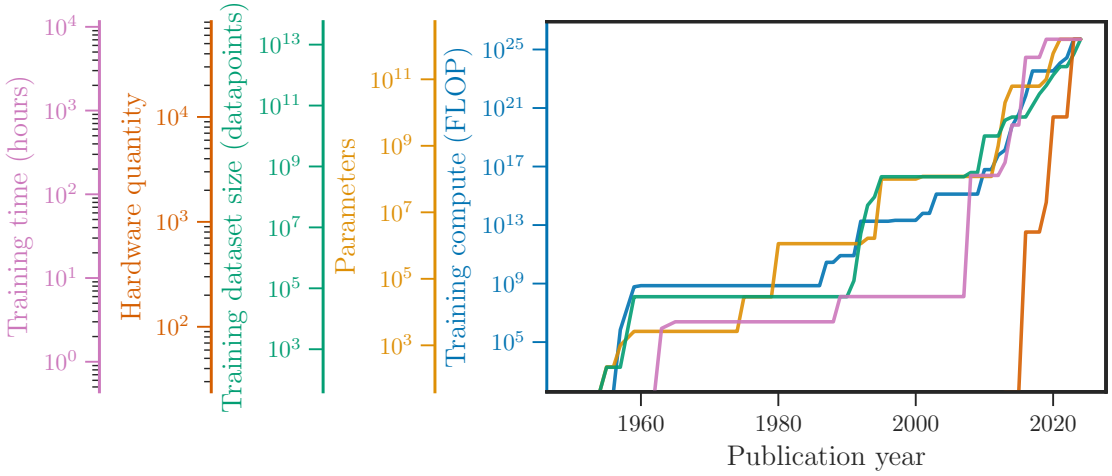


Figure 1.2: Evolutions in Machine Learning models since 1950, as the maximum value up to the given year [AI2024]

1.2 Designing computing infrastructures to handle ML computations

Machine Learning has radically changed since it was first introduced in the 1950s. This growth can be seen from different perspectives. Figure 1.2 shows the exponential growth in the number of trainable parameters of state-of-the-art models since the 1950s, reaching more than 10^{11} in 2023. It was exacerbated by a common practice in ML to overparametrize the models, i.e. choosing a model with more parameters than data points in the dataset. In the Epoch AI database that was used to generate figure 1.2, 47% of the notable models with parameter and data point information have more parameters than data points. 29% of those models have more than 10 times more parameters than data points. Overparametrization means more computations and, according to Thompson et al. [Thompson2020], "the computation required to train an overparameterized model should grow at least as a fourth-order polynomial with respect to performance, i.e. $Computation = O(Performance^4)$". Thus improving the performance of a model requires much more computations. As a consequence, the number of Floating Point Operation (FLOP) required to train model has been growing from 10 in 1957 to 10^{25} in 2023.

The availability of massive datasets has been critical in the advances of the research field. For example, the release of ImageNet in 2012 has unlocked research in computer vision and led to the publication of state-of-the-art models in classification and image recognition. Figure 1.2 shows the evolution in the number of data points required to train the most notable state-of-the-art models as reported by Epoch AI [AI2024]. The biggest datasets reached a thousand billion data points in 2023.

Both the number of data points and the quality of the samples have a significant impact on learning. Collecting and processing data can be expensive. Creating a dataset comes with various challenges: annotating inputs for supervised tasks, fairness between categories (problem of racism in recruitment algorithm), or finding the best processing technique for learning to be more efficient (normalization). Advances in these area enabled the explosion in the number of parameters.

Since the 1950s, the progress in ML has been closely linked to the progress in hardware performances. Unlike most data center workloads, gradient computations are memory-intensive and highly parallelizable. The evolution in hardware specialized for ML has been significant in the past few decades. This section starts by detailing this evolution. The failure of the hardware evolution to match the requirement from ML model training and its cost has led to the multiplication of compute nodes used for training by distributing data parallelism. Finally, new demands such as data privacy and user personalization have pushed the computations toward the user, bringing new

challenges in managing computations and data movement.

1.2.1 Performance of ML-specialized hardware

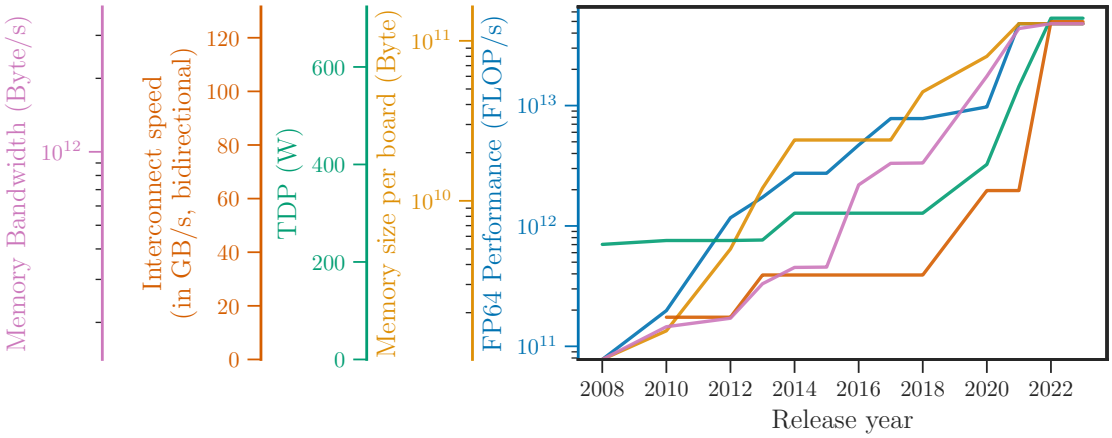


Figure 1.3: Evolutions in Machine Learning specialized hardware (GPU, TPU) metrics, as the maximum value up to the given year [Hobbhahn2023]. The interconnect speed and the TDP are expressed on a linear scale while the other metrics use a logarithm scale.

Moore's law predicted a fast increase in chip performance, stating in 1965 that the number of transistors in an integrated circuit doubles about every two years. In ML, the progress in Graphic Processing Unit (GPU) has marked a shift. Thompson et al. show that GPUs have allowed to reduce the computation time by 35 by 2012 [Thompson2020]. Since then, the performance of ML-specialized hardware has been exponentially improving. Sevilla et al. [Sevilla2022] estimated that a large-scale ML trend had started in 2015 and since then the performance of ML-specialized hardware (FLOP/s) has been doubling every 9.9 months for a selected number of large-scale models. Researchers from Epoch AI have made statistics on the performance of GPUs and Tensor Processing Unit (TPU), publishing more detailed results:

- Computational performance [FLOP/s] has doubled every 2.3 years for both ML and general GPUs.
- Computational price-performance [FLOP per $] has doubled every 2.1 years for ML GPUs and 2.5 years for general GPUs.
- Energy efficiency [FLOP/s per Watt] has doubled every 3.0 years for ML GPUs and 2.7 years for general GPUs

The performance of ML-specialized hardware is classically evaluated in Floating Point Operation per Seconds (FLOP/s), but other metrics impact the performance of training ML models. Managing data movement has become a critical issue. This can be seen in figure 1.3 which is based on the database released by Epoch AI [Hobbhahn2023] and depicts the highest recorded value for each metric throughout the period from 2008 to the given year. It shows the logarithm evolution of the FP64 performance, the memory size per board, and the memory bandwidth. The interconnect speed and the thermal design power are displayed in a linear scale. All metrics have significantly increased since 2008. The FP64 performance in FLOP/s has gained more than two orders of magnitude between 2008 and 2023. The improvement in memory size has been slower but steady. The memory bandwidth was doubled in 2016 and 2022, enabling a boost in FP64 performance that had been stagnating since 2014. Those advancements are partly due to the development of accelerating chips using tensors for matrix multiplication. The most popular and efficient chips

are NVIDIA's V/A/H100 and Google's TPUs. A reduction in the precision of floating point has also led to a significant improvement in FLOP/s. Figure 1.4 shows the improvements due to using tensors and reducing the precision for the most popular and recent hardware. Reducing the precision can increase by 2 orders of magnitude the number of operations per second, but a balance needs to be found with the decrease in learning performance. Using tensor mostly improves the performance.

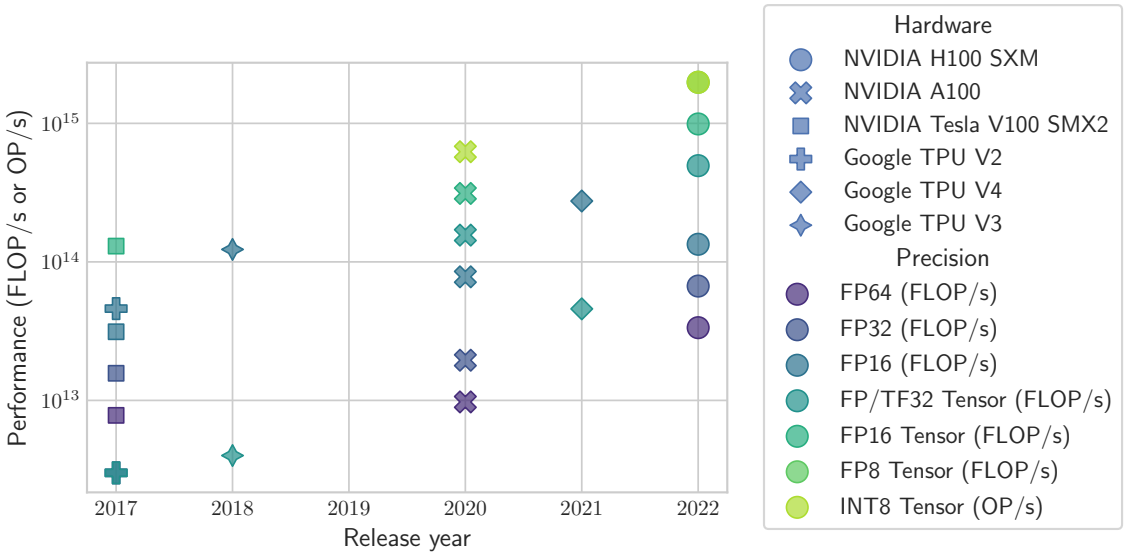


Figure 1.4: Performance of the most recent and popular ML-specialized hardware with various precision [Hobbhahn2023]

One can notice that the best hardware performance reaches 10^{15} FLOP/s (Figure 1.4) when the most recent models require up to 10^{25} FLOP (Figure 1.2) to be trained. That would mean more than 300 years of training. Despite the major improvements in hardware performance, the demands from ML model training have increased more rapidly. ML developers have taken advantage of the parallelization characteristic of ML training computations to compensate for this gap, thus distributing data to parallelize learning across GPUs and compute nodes. Top500 is a project tracking and detecting trends in high-performance computing. It produces a list of the 500 most powerful HPC systems twice a year. The Frontier system is the most powerful system in June 2024. It contains more than 8 million cores and has a peak performance of $1.6 * 10^{18}$ FLOP/s, reducing the number of training days of the most recent model to 72.

1.2.2 Distributed Data Parallelism

Figure 1.2 shows the evolution in the number of machines used for training one model in the past decade, according to Epoch AI database [AI2024]. Distributed learning started to gain attention in 2012 when AlexNet was trained between five and six days on two GTX 580 3GB GPUs [Krizhevsky2012]. The average number of GPUs used to train models went from 2 in 2012 to 3196 in 2024 [AI2024], although the reality is more heterogeneous. Some models still rely on only one GPU while the most expensive model in the database (Gemini 1.0 Ultra) was trained on 55000 Google V4 TPUs in 100 days.

Distributed learning comes with new challenges. It necessitates robust high-speed networking capabilities to share parameters while minimally impacting the training time. ML clusters consist of a few to hundreds of nodes equipped with 8 GPUs. The communication between nodes has improved a lot too. Figure 1.3 shows that interconnect speed has been multiplied by 3 since 2018 thanks to the new generations of NVLink and NVSwitch.

Despite the improvement in hardware performance and the increase in hardware number, the training time continues to exponentially increase. As can be seen in Figure 1.2, the longest training time was multiplied by 10 between 2007 and 2008 and reached months around 2015. The Thermal Design Power, the maximal power supported by the computing components, is increasing too. In Figure 1.3, we can see that the latest GPU has a TDP of 700 W. With typically 8 GPUs and 2 CPUs, an ML compute node can consume up to 8kW which is significantly higher than a classic data center compute node. It means that despite the increase in energy efficiency, the total energy consumed by ML training is soaring. Data center infrastructures have had to be adapted to support such electricity demand [1] and significantly impact the electricity grid [Libertson2021].

1.2.3 Pushing computations towards the edge

The growth of the volume of data generated by user devices and the corresponding threat to privacy has led to the development of new paradigms in learning. Federated Learning (FL) is an example of such evolution that's widespread and well-researched. Instead of parallelizing a unique dataset, it relies on the client's local datasets. It was first introduced by Google [McMahan2017] whose goal was to improve keyboard predictions and voice detection with data coming from clients while preserving client privacy. The data that is generated from user interaction can be used to personalize or fine-tune models. FL and edge computing can be seen as moving the computations closer to the user rather than bringing the data to the compute node. Instead of collecting user data back to data centers, data samples are kept close to where it was produced.

Examples of devices that can benefit from FL are smartphones, smartwatches, smart speakers, or more generally embedded devices. The number of smartphones globally reached 8.36 billion in 2022 [in Data2024]. Although such devices consume less power (under 100 W) than compute nodes, the scale is much larger and devices additionally rely on networks.

1.3 ML environmental impacts

A methodology evaluating the environmental footprint of ML should include all the digital devices and hardware components that it has relied on. On one hand, centralized learning pushes the development of more and more powerful and energy-consuming specialized hardware. On the other hand, edge computing is pushing for more computations located in user terminals, increasing their energy consumption and relying on networks. Such a methodology should also take into account the impact of each phase of the development and deployment of ML models, from collecting and processing data to selecting, testing, training, and fine-tuning models and finally deploying them. Figure 1.5 gives an overview of the model and hardware life cycles.

This section summarizes the most relevant work done on the environmental impacts of ML.

1.3.1 Electricity consumption and carbon emissions of the training phase

The interest in the carbon emissions of training ML models started in 2019 with the work of Strubell et al. [Strubell2019]. The authors evaluate the energy consumption and carbon emission of 4 NLP models based on electricity measurements. They demonstrate that their carbon emissions are significant, comparing them to the lifetime emissions of thermal cars. They show that training

[1] https://www.datacenterdynamics.com/en/analysis/how-meta-redesigned-its-data-centers-for-the-ai-era/

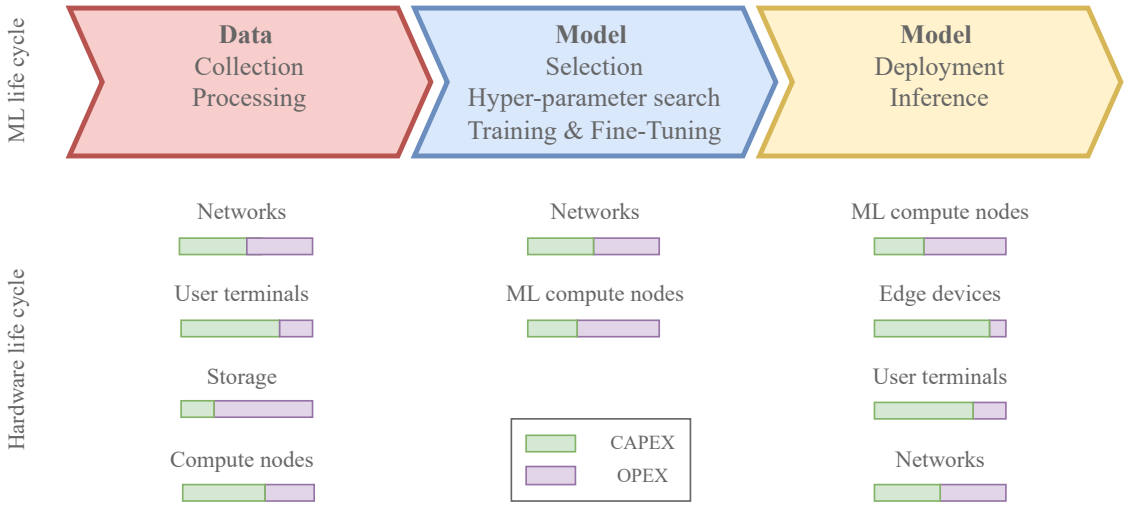


Figure 1.5: Overview of the phases of an ML model life cycle, of the ICT equipment involved in each phases, and their hardware life cycle. Equipment and CAPEX/OPEX shares are arbitrary choices.

a Transformer model with neural architectural search is equivalent to the emissions of 5 cars in their lifetime.

This initial study on state-of-the-art NLP models was followed by similar work on other disciplines or models. [Henderson2020] evaluate the emissions of a few reinforcement learning models, proposing a leaderboard to foster competition to reduce carbon emissions. Researchers from Google [Patterson2021] present an analysis of popular architecture like Transformer 4, GPT-3, Meena, and Gshard. Similarly, researchers from Meta published the operational footprint of the offline training of recommendation models deployed by the company. [Luccioni2023b] analyze the electricity consumption and carbon emissions of training BLOOM, an open-source LLM.

[Schwartz2019] proposed in 2019 the term "Green AI" to promote AI models optimizing energy efficiency in addition to model accuracy. The authors alert on the exponential evolution of ML model size. Those observations are later confirmed by other analysis [Thompson2020, Sevilla2022], showing that this drastic growth includes other metrics too, such as the computation burden (FLOP), the training time, and by consequences the energy consumption, despite the improvement in performance and energy efficiency of specialized hardware. [Patterson2022] balance this result by stating that, at Google, improvements in model, software, and hardware efficiency enabled a stagnation of the share of ML-related workload in the company's total energy consumption. Since the energy spent in Google data centers is significantly growing [2], it doesn't mean that the energy spent in ML is constant, but, according to the authors, its growth is much slower than the growth in model parameters and training computations. It corroborates previous findings from [Henderson2020] that FLOP is not correlated to energy consumption in ML training and should not be used as an estimation as it was previously done.

Measuring and estimation tools [García-Martín2019] reviewed in 2019 the key approaches to estimating the energy consumption of computations and how to apply them on machine learning applications. The first section focuses on energy estimation models and groups the 23 reviewed techniques into four categories: (i) performance counter-based models; (ii) simulations; (iii) architecture or instruction-based models; (iv) real-time power. The second section covers energy measurement and estimation tools. Only five command-line tools are described and none of them

[2] https://www.gstatic.com/gumdrop/sustainability/google-2024-environmental-report.pdf

include GPUs. Three of them are based on RAPL. Finally, two use cases show the usefulness of selected tools but they are not compared with each other nor with power meters. Since then, new tools and methodologies have been proposed to make energy consumption and carbon emission measurement easier for model developers. First, in 2019, two online tools were published to enable estimations based on the model training time and hardware and data center characteristics. [Lacoste2019] provide a website called "MLCO2 impact" for estimating carbon emissions based on GPU type, experiment length, and cloud provider. [Lannelongue2021] have further integrated factors in their online calculator such as core usage and unitary power draw for more hardware precision and the pragmatic scaling factor to take into account successive training tests. Both rely on processor databases to make the tools more accessible to the users. A second wave of Python software published in 2020 [Henderson2020, Anthony2020, Schmidt2021] enabled more precise estimation by retrieving values from RAPL and Nvidia-SMI interfaces. [Bannour2021] compare those energy estimation tools on both qualitative and quantitative criteria. They evaluate the tools on Natural Language Processing (NLP) use cases, focusing on CO2 equivalent emission estimations, and show that the estimates can vary significantly. [Dodge2022] are the first to conduct a study of cloud instances and compare the training of popular models on them. [Jay2023, Heguerte2023] review such tools and evaluate them against power meters on HPC benchmarks and AI trainings, respectively. They show a significant gap between compute node consumption and the tool estimations.

Recommendations Those studies also provide recommendations on how to reduce the emissions of ML training. Most of them urge the community to report the energy consumption [Strubell2019,Lacoste2019,Lannelongue2021,Henderson2020,Anthony2020,Patterson2021,Wu2022, Dodge2022] to increase transparency and better understand the impact of the field. Similarly, they ask conferences to include such reports and reproducibility in their acceptance criteria [3]. They also recommend using efficient models and hardware [Strubell2019, Schwartz2019, Lacoste2019, Henderson2020, Patterson2021, Patterson2022, Wu2022], dimensioning the hardware to the model [Heguerte2023], minimizing the hyperparameter and architectural search [Schwartz2019,Lacoste2019], carefully choosing the cloud provider or data center location to reduce the carbon mix and the PUE and maximize the offset [Schwartz2019,Lacoste2019,Henderson2020,Patterson2021,Dodge2022]. Both Google and Meta evaluate the impact of such optimization on their carbon footprint. At Meta, optimizations including machine, GPUs, and low-precision data format led to an 810x reduction in operational footprint [Wu2022]. At Google, the choice of model, data center, and specialized hardware can reduce the carbon footprint 100 to 1000 times [Patterson2021]. Dodge et al. show that the "flexible start" and "Pause and Resume" scheduling strategies can save up to 27% carbon emissions, averaged over the year.

Global estimations To the best of our knowledge, there is no estimation of the worldwide cost of ML as there is for ICT. The closest number we have is a company-level estimation by Google [Patterson2022]. The authors estimated that 10% to 15% of Google's total energy consumption between 2019 and 2021 was dedicated to ML-related workloads (both training and inference). Considering that the annual energy consumption of Google in 2020 was 15.4 TWh, ML-related workloads consumed between 1.54 and 2.31 TWh. In comparison to the total energy consumed by data centers as provided by the IEA, Google ML-related workloads represent approximately 1%.

The IEA approached the question differently. They evaluate the impact of AI using the number of GPUs sold by Nvidia and estimate that dedicated AI data centers with consume approximately 100 TWh in 2026 [De Vries2023, Agency2024]. They also forecast that the growth in data center

[3] It has been added to NeurIPS Ethics Guidelines: https://neurips.cc/Conferences/2023/EthicsGuidelines

electricity consumption will increase by 29% from 2023 to 2026 of the electricity demand in the European Union.

1.3.2 The carbon footprint of manufacturing and other hardware life cycle phases

In their review of existing estimation tools published in 2021, [Bannour2021] state that emissions resulting from the production and end-of-life phases are unaccounted for or partially accounted for. Since then, several studies have been published.

[Gupta2022] study the Life Cycle Assessment (LCA) of multiple user and data center devices based on sustainability reports. They compare the share of operational and embodied carbon footprint and they show that manufacturing can account for the majority of the hardware and infrastructure impacts, especially since the energy efficiency of digital devices is increasing.

[Ligozat2022] propose a framework to assess the environmental impacts of AI applications and the embodied costs with several criteria. They present how Life Cycle Assessment can be applied to AI services and list the life cycle stages that should be considered according to the ITU recommendation [ITU2014]. Unfortunately, they don't apply the framework to a use case.

A study called "Sustainable AI: Environmental Implications, Challenges and Opportunities" by [Wu2022] is the most advanced analysis by 2023 of the carbon footprint of the development and deployment process of ML at a company level. The authors study various use cases at Meta, mainly recommendation models that run daily. They are the first to project the embodied carbon cost to Large-Scale ML tasks with a time-based allocation. They show that at Meta the manufacturing cost corresponds to half the operational cost thus a third of the global carbon footprint. Data from Meta's research training infrastructure show that the GPU utilization stays between 30% and 50% when the utilization should be maximized to amortize the embodied footprint.

The authors of "Measuring the Carbon Intensity of AI in Cloud Instances" [Dodge2022] include the embodied footprint in their tool.

The analysis of the footprint of training BLOOM [Luccioni2023b] includes the embodied cost, allocated with a time-based approach. The authors find that the embodied cost represents 22.2% of the total carbon footprint. They additionally compute the exact additional consumption due to running the data center infrastructure, encompassing more elements than the PUE would, and discover that it corresponds to 28.9% of the total footprint, leaving a bit less than 50% to the dynamic use of the compute nodes.

Unfortunatly, few recommendations are provided on how to reduce the embodied impact of AI. AI companies still buy large amount of AI-specialized computing nodes and manufacturing companies continue to design new chips.

1.3.3 Impacts of inference phase and deployment

The effect of the inference phase on the environment has long been discarded and dimmed insignificant as regards the effect of training. The large-scale deployment of popular chatbots such as ChatGPT marked a shift in this, attracting scientific interest. Performing one inference has significantly less impact than training. Its the scale in the number of inference that is beginning to have a significant impact, thus the behaviour of user now has an importance.

Both articles from Google [Patterson2022, Patterson2024] and Meta [Wu2022] have provided statistics for their companies. At Google, inference represented 60% of ML energy use for the three years they studied. Meta published that inference accounted for one-third of the company's ML footprint. [Patterson2022] cite statements from Amazon Web Services claiming the share of inference in ML workloads could go up to 90% among their consumers [Leopold2019, Barr2019], but without information on how this would translate in terms of energy footprint.

While analyzing the cost of BLOOM, [Luccioni2023b] conducted an experiment to estimate the energy and carbon impact of deploying the model on Google Cloud. Over 18 days, the compute nodes handled 230,768 requests, consuming 3.960 kWh and 1.4 gCO2eq per request. The service

would have to run for more than 7 years with a similar workload to be equivalent to the training cost.

Improving the energy efficiency of inference is a large area of research, but those articles suggests that the rebound effect might counterbalance its gains.

1.3.4 Electricity consumption and carbon emissions of the training phase in edge devices and Federated Learning

The impact of bringing ML to the edge has also been studied. Application-level analyses propose methodologies on how to estimate the energy consumed by Federated Learning [Savazzi2021]. Studies compare centralized, federated, and distributed learning settings on various applications: image classification models [Qiu2021, Qiu2024], speech detection [Qiu2024], LLMs [Wu2022], reinforcement learning [Savazzi2022a] and continual learning [Savazzi2022a]. In conclusion, bringing ML to the edge doesn't systematically enable saving in terms of energy consumption or carbon emissions. It depends on the number of rounds and the efficiency of hardware and communication [Savazzi_2022b].

On the opposite, a recent study from Patterson et al. [Patterson2024] shows that FL can emits 100 times more carbon than CL, at Google scale. The authors define an equivalent of the PUE for smartphones, calling it the charger PUE (CPUE). They found it was as high as 3.1. Moreover, one case study suggests training on devices also uses 12 more energy than in the data center. From a different perspective, ML energy use was less than 3% than the global energy use of smartphones.

As in learning algorithms, FL brings new challenges in reducing the environmental impacts of ML. Edge devices have a more significant embodied footprint while being under-utilized during their lifetime which makes it harder to amortize. They may have less access to renewable energy. Short lifespans due to planned obsolescence or rapid technological advancements lead to a constant stream of electronic waste.

1.3.5 Adding more environmental impact indicators

When presenting their framework, Ligozat et al. [Ligozat2022] indicate that LCA enables other indicators of environmental damage than carbon emissions. However, none of the previously mentioned studies include another indicator. This focus on carbon emissions can be harmful if it hides other significant impacts such as rare metal depletion or the stress induced by increased water demand at specific locations. While renewable energy and data centers with a lower carbon electricity mix can reduce the carbon footprint of computations, the shift can increase other impacts.

To the best of our knowledge, "TinyML" [Prakash2023] is the only study with a multi-criteria analysis (climate change, water demand, freshwater eutrophication, photochemical oxidant formation), but it's applied to microcontrollers which mostly process data and perform inference, and are rarely used for training. It is interesting to notice that energy consumed during device production is one of the most significant parts of each impact. Extracting raw materials demands the most water and the usage phase represents less than 10% of total impact, except for freshwater eutrophication.

Additionally, none of the existing studies convert electricity into primary energy, omitting the transformation cost.

1.4 Research challenges and objectives

A strong assessment methodology should enable developers to find leverages to reduce the impacts, in the usage phase as well as in the choice of computing infrastructure. Such a methodology should not be specific to an application domain but should apply to any ML model. Focusing on GHG emissions might be important to address climate change, but a more holistic approach that considers the full range of environmental impacts is essential to designing truly sustainable digital services.

This thesis focuses on the training phase of machine learning, as it presents the most significant opportunity to reduce the environmental footprint through careful computing infrastructure design. Compared to the inference phase, where deployment constraints often limit infrastructure choices, the training phase allows for greater flexibility in selecting the most environmentally friendly hardware and software configurations.

This thesis aims to address these challenges with the following objectives:

- Proposing a methodology to assess more environmental impacts of both the usage and embodied phase of training ML models.
- Applying this methodology to state-of-the-art ML models and various sizes of ML infrastructures to compare them.
- Providing a deeper understanding of the electricity consumption of ML training when infrastructures become more diverse and complex.

Experiments presented in this thesis were carried out on several clusters. The Champollion cluster was designed by HPE and made available to us thanks to Bruno Monnet through collaboration with MIAI. The Grid'5000 / Slices testbed is supported by a scientific interest group hosted by Inria that includes CNRS, RENATER, and several Universities as well as other organizations [4]. This work was funded by MIAI (ANR19-P3IA-0003) and the BATE project (BATE-UGAREG21A87) of the Auvergne Rhône-Alpes French region.

1.5 Contributions

The development of the methodology and its application on several models and infrastructures led to the following contributions:

- A methodology to assess the global warming potentiel, the primary energy, and the abiotic depletion potential of the training phase of a model life cycle, including the embodied cost of the computing infrastructure.
- A quantitative and qualitative comparison of existing electricity measuring tools.
- An analysis of the electricity consumption of training ResNet-50, 3D U-Net, Mask R-CNN, RNN-T, BERT-large, and DLRM on an Apollo 6500 Gen10+ node and of training ResNet-50 on an Nvidia Jetson AGX Xavier.
- An analysis of the embodied impacts of both infrastructures, and a comparison of the usage phase with the allocated embodied impacts.

[4] https://www.grid5000.fr/

- A comparison between such infrastructures, in terms of electricity consumption and environmental impacts.

During this thesis, I had the opportunity to publish articles in peer-reviewed conferences and journals. They are detailed in Appendix A.

1.6 Content

This thesis is organized as follows:

- Chapter 2 covers our proposed methodology. It presents the computing infrastructure and the ML models on which we applied the methodology.
- Chapter 3 provides background knowledge on how to measure the electricity consumption of computing infrastructures and a qualitative and quantitative assessment of a selection of software-based power meters.
- Chapter 4 analyses the electricity consumption of training the model we selected. It gives an overview of power profiles and utilization metrics and studies the relationship between the electricity consumption and the model parameters, the accuracy and the number of nodes. Assessing the Apollo node and the Jetson node on the same use case enables a comparison of their energy efficiency.
- Chapter 5 estimates the embodied impact of the computing infrastructures and allocate them to the training phase of the models. A comparative analysis is made.
- Chapter 6 discusses the results presented in the previous chapters and puts them into perspective.

2

Methodology, Infrastructures, and Machine Learning Training Benchmark

Contents

We propose a methodology to evaluate the environmental impact of training Machine Learning models, with the following characteristics:

- Versatile: it can be applied to any computing infrastructure and ML model training.
- Reproducible: Experimental settings ensure that no extra phenomena affect electricity consumption and environmental estimations are based on open-sourced databases.
- Insightful: It enables the developer to find reduction leverages.

To do so, we get inspiration from a methodology standard called Life Cycle Assessment (LCA). LCA is a technique for assessing the environmental aspects associated with a product over its life cycle. It is defined by ISO standards (ISO 14040 [1] and 14044 [2]) and by a specific methodology standard for ICT goods, network, and services from the International Telecommunication Union (ITU) [ITU2014]. It has the advantage of enabling multi-criteria assessment and the inclusion of all the life cycle phases and all the relevant impacts a product or a service can have. Its most important goals are to permit an analysis of phases with improvement potentials and comparisons between products or services. This is especially important to evaluate if the digitalization of services leads to a reduction of impacts [Rasoldier2022].

LCA consists of 4 main stages:

- Define the perimeter, or Functional Unit (FU), of the assessment. For example, it can be performing one inference of an AI model.
- The Life Cycle Inventory (LCI): List the material and energy flows and their interaction with the environment. Determine the most relevant impact categories. In AI, one must consider all development phases (model selection, model training) and the equipment used to perform them.
- Assess every impact category for the FU.
- Review hypothesis with sensibility analysis.

The chapter starts by presenting the scope of methodology (Section 2.1). Section 2.1.2 details the experimental setup and data collection process required for the methodology to be both insightful and reproducible. Section 2.1.3 is focused on LCI modeling and impact categories.

To illustrate the versatility of our approach, we apply it to two distinct computing infrastructures and 6 models from every major discipline. We selected infrastructures designed for AI: an HPC infrastructure and an edge device, and we describe them in section 2.2. Through this comparative analysis, we highlight the methodology's potential to assess the footprint of infrastructure for AI workloads. The ML models we selected are detailed in section 2.3. We picked models from the MLPerf training benchmark suite since it represents state-of-the-art models from every major discipline. Section 2.4 presents the Functional Units for each model and infrastructure that will be assessed in this thesis.

[1] https://www.iso.org/fr/standard/37456.html
[2] https://www.iso.org/fr/standard/38498.html

2.1 Methodology

2.1.1 Scope

The life cycle of an AI model can be decomposed into many phases. First, data needs to be collected, processed, and eventually labeled. Then, the model is selected and trained on the data. Finally, the model is deployed such that users can access it. Each phase has an impact, as it was presented in Chapter 1.

In this thesis, we evaluate the electricity consumption and environmental impact of the training phase of an ML model. This training is performed until a given quality is reached on the validation dataset.

We assume that a model and a dataset have already been selected, that the dataset has been processed, and the hyper-parameter search has already been conducted. Thus data collection, data processing, model selection, hyper-parameter search, and model deployment and inference are outside of the scope of this methodology. Figure 2.1 shows the scope of our methodology in the ML model life cycle.

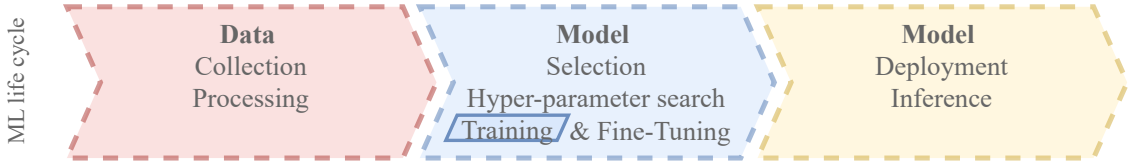


Figure 2.1: Scope of the methodology in the ML model life cycle.

The quality of the model on the validation dataset is the main performance metric of a model. It provides an estimate of how well the model will perform on real-world data and can be used to compare multiple models. Fixing the quality enables fair hardware performance comparisons across infrastructures.

The computing node is also chosen for the task. The scope encompass the computing node that was used to perform the training. That includes the computing node, the rack that contains it, and the power supply but not the networking components, the support devices of the data center, and the cooling infrastructure. Storage is also outside the scope of our methodology. Both the OPEX and CAPEX costs are included in the environmental assessment. The scope is retricted to the direct effects. We consider that the infrastructure is not shared during the execution and that the training uses every component it contains.

The functional units are detailed specifically for each use case studied in the thesis in Section 2.4. To enable fair comparison, FUs should be evaluated on a year of reference.

2.1.2 Experimental setup and data collection for the measure of electricity consumption

The operational phase of training ML models has a growing impact. As shown in Chapter 1, both the Thermal Design Power (TDP) and the training time of training ML models are increasing over time. As a consequence, so does their electricity consumption.

ML nodes consist of several components working together to train models. Storage holds the dataset and periodically sends data batches to local RAM. This data is then transferred to GPUs for intensive computations. The CPU plays a crucial role in managing the training process, including data preprocessing and model compilation. Interconnect refers to the communication pathways connecting these components. Key interconnect technologies include PCIe, NVLink, and InfiniBand, each offering varying levels of bandwidth and latency. High-bandwidth, low-latency interconnects are crucial for accelerating data transfer between CPUs, GPUs, and memory,

significantly impacting training performance. Understanding the electricity footprint of a node requires careful monitoring of all those components.

How the electricity consumption is measured matters. The power consumption should be monitored at a frequency high enough to be able to capture the evolution of the power of each component of the node. Each of those criteria can bring different insights that are necessary to find leverages in energy reduction, especially if they can be correlated to the implementation of the training algorithm or the model characteristic.

Reproducibility is a key ingredient of this methodology because electricity consumption is highly variable and influenced by factors such as external temperature. To accurately study this metric, experiments must be rigorously reproduced multiple times. An idle period of at least 30 seconds between experiments is essential to ensure power consumption returns to a baseline level before starting the next trial. Power profiles can vary slightly between nodes, even for identical systems. The experimental setup should involve a single node or a fixed set of nodes.

In Chapter 3, we compare existing measuring tools on many criteria, and notably on those aspects, to be able to select the one that best suits our needs.

We collect the evolution of validation and train loss and quality from the AI training. The frequency must be set such that their evolution can be studied regarding electricity consumption.

2.1.3 Estimating the embodied cost associated with the model training phase

As it was mentioned in Chapter 1, the embodied (or CAPEX) impact of ICT devices is significant and needs to be included to assess the environmental impact of a service. The embodied impact of computing infrastructures can be estimated from databases like Boavizta [3].

ML nodes that perform training are most of the time used for more than one training or for another ML workload. Associating the whole embodied impact of the node to one training would be exagerated. Attributional Life Cycle Inventory (LCI) is an LCI model that allocates each impactful process from the life cycle of a service. It is known to have fewer uncertainties than the consequential approach while being easier to implement due to the availability of databases. An allocation key must be defined to assign the impacts to a specific service. In the case of ML training, we consider that the computing node is dedicated to the service for the entire execution. Therefore, we use a time-based allocation, as in Equation 2.1.

$$\text{Allocated impact} = \text{Embodied impact} * \frac{\text{Execution Time}}{AUR * \text{Total lifetime}} \tag{2.1}$$

With AUR the Active Utilization Rate being the portion of the equipment lifetime during which it was actively used as opposed to idle or underutilized. The AUR is an hypothesis that might be different depending on infrastructures.

We express the environmental impacts into three categories [Simon2024]:

- Primary Energy (PE), in kilo Watt-hour (kWh). It represents the cumulative energy consumption of a system, including all energy sources used directly or indirectly from extraction to end-use.
- Global Warming Potential (GWP), in equivalent C02 emission (kg CO2 eq), that evaluates the contribution to climate change.
- Abiotic Depletion Potential (ADP) of minerals and metals, in equivalent antimony (kg.Sb.eq), that represents the increase in scarcity of minerals and metals resources.

[3] https://datavizta.boavizta.org/

These 3 categories encompass the most significant environmental concerns associated with digital technologies [Benqassem2021]. Primary Energy (PE) consumption, though not a direct measure of environmental impact, is a critical sustainability indicator. This is because limited energy resources can indirectly contribute to various environmental issues. Increasing the stress on electricity grids can lead to the usage of electricity production site emitting more equivalent CO2 per kWh or to the installation of additional electricity production sites that can have a significant impact of the local environment, biodiversity, and communities. PE also includes industrial heat cost.

Water Usage is another consideration for AI's environmental footprint. However, including a water consumption metric is currently hindered by a lack of reliable data [Li2023]. Additionally, water's impact is highly contextual. "When and Where" water is withdrawn significantly affects its environmental significance. One liter used in a drought-stricken region has a much greater impact than one used during a rainy season with plentiful resources [Li2023].

For the operational impact of ML training, we take into account equipment outside of the computing node using the Power Usage Effectiveness (PUE) and we convert the electricity consumption to PE to account for energy used in the electricity production process. The impact factors of electricity were published in the ADEME database [4]. We recommend using location-based impact factors.

2.2 Infrastructures

To apply our methodology, we selected two infrastructures that were designed for ML training. Champollion is an HPC platform while Jetsons are edge devices. They are representative of typical AI environments. Notably, Champollion is an AI cluster with similar capabilities as clusters that are used to train Large Language Model (LLM)s, in terms of computational power, memory capacity, and network bandwidth. Jetson nodes are designed to be embedded devices thus their electricity power consumption is low while their computational capabilities enable them to run AI applications like object detection in self-driving cars. Both infrastructures have potential to reduce the footprint of AI workload. On one hand, Champollion is energy efficient but on the other hand, a Jetson node has a low power consumption. Depending on the application, it is not clear which infrastructure consumes more energy. Due to this open question, they are relevant choices for this thesis. Additionally, their differences in software stack, computational capabilities, and power consumption show that our methodology can be applied to different infrastructures.

We had the chance to be able to experiment on the Champollion cluster thanks to a collaboration between HPE and MIAI. The Jetson cluster was available on the Grid'5000 testbed.

This section describes the specifications of the Apollo 6500 Gen10+ nodes of the Champollion cluster and of the Nvidia Jetson AGX Xavier node.

2.2.1 HPC: the Champollion cluster and its Apollo nodes

Champollion is a supercomputer designed by HPE. It was ranked 370 on the Top500[5] in June 2023 which ranks supercomputers by their ability to run the Linpack benchmark (solve a set of linear equations) and 13 in the Green500 [6] which ranks them by their energy efficiency. In June 2024, Champollion was ranked 451 on the Top500 and 22 on the Green500. Its specifications can be

[4] https://base-empreinte.ademe.fr/
[5] https://www.top500.org/system/180073/
[6] https://www.top500.org/lists/green500/list/2023/06/

found in table 2.1. The cluster consists of 20 nodes connected with 8 Mellanox HDR Infiniband switches. The inter-GPU communication is operated by the NVLink technology. Appendix C presents the schema of the Champollion organization. The cluster performance theoretical peak is 2.52 PFLOPS (10^{15} FLOPS) and its power consumption was measured to be 60.20 kW on the Linpack benchmark. Its energy efficiency and similarity to computing infrastructures used to train the most well-known models like GPT make Champollion a relevant infrastructure for this study.

Node model	Apollo 6500 Gen10+
Number of nodes	20
GPU model	NVIDIA A100-SXM-80GB
Number of GPU per node	8
GPU TDP (W)	400
CPU model	AMD EPYC 7763 64-Core Processor
Number of CPU per node	2
CPU TDP (W)	280
Memory	1 TB
Switch model	Mellanox HDR Infiniband
Number of switch	8
Switch power consumption (W)	375
Installation year	2022

Table 2.1: Champollion characteristics. TDP: Thermal Design Power; W: Watt

The nodes have access to various storage options. Each node has 3 TB available through local nonvolatile memory express (NVMe) interfaces. A Parallel File System Storage (PFSS) and Clusterstor E300 can be used to store the datasets.

For simplicity, a node from the Champollion cluster will be referred to as an Apollo node in the remainder of this thesis.

2.2.2 Edge: Jetson

We relied on nodes from the Estats cluster of the large-scale test beds for experimental research called Grid'5000 [Balouek2013]. This cluster was selected because its nodes have similar computational capabilities as embedded devices specialized for AI training. On top of that, it was designed to consume low power with power modes from 10 W to 30 W. The energy consumption of the Nvidia Jetson serie and leverages for energy reduction has been studied [S.K2022].

The specifications of this cluster can be found in table 2.2. Appendix C show a picture of one node. Each node has access to a 2 TB Solid State Drive (SSD) available through local NVMe interfaces.

Table 2.3 shows the theoretical performances of a Jetson node.

For all experiments, we used Ubuntu 20.04 as available on the Grid'5000 testbed. In order to increase consumption stability and the consistency of our results, we have set the CPU frequency to the maximum supported. We also installed an Nvidia GPU driver when relevant, with default power management configuration.

For simplicity, an Nvidia Jetson AGX Xavier node will be referred to as a Jetson node in the remainder of this thesis.

Node model	Nvidia Jetson AGX Xavier
Number of nodes	12
GPU model	NVIDIA GV10B, Volta architecture
Number of GPU per node	1
CPU model	Nvidia Carmel (Carmel), aarch64, 8 cores
Number of CPU per node	1
Memory	32 GB
TDP (W)	30
Installation year	2023

Table 2.2: Jetson characteristics. TDP: Thermal Design Power; W: Watt

FP16 (half)	2.820 TFLOPS (2:1)
FP32 (float)	1,410 GFLOPS
FP64 (double)	705.0 GFLOPS (1:2)

Table 2.3: Theoretical performance of a Jetson node

2.3 A Machine Learning Training benchmark: MLPerf

As it was said in the Chapter 1, ML has radically changed since it was first introduced in the 1950s. Appendix B presents the mathematical foundations of ML (Section B.1) and Neural Networks (NN) (Section B.2). It introduces key concepts of ML that are used in this thesis. ML applications are numerous and diverse thus we selected a benchmark representing most of those applications.

MLPerf is a consortium of major commercial and academic organizations created to design a benchmark suite for deep learning to fairly evaluate system performance. The suite has several benchmarks from HPC training to tiny inference. We focus here on the MLPerf training suite. It covers a broad and diverse range of applications and is updated twice a year to stay up to date, which makes it compatible with a study of the power and energy consumption of training state-of-the-art AI models. The benchmark code is open source and the implementation and training procedures are precisely defined to ensure reproducibility and fair comparison of systems, which aligns with our methodology guidelines.

For this thesis, we chose models from three major areas of ML. **Computer Vision** is a field that aims at mimicking the human visual system. This includes processing digital images, understanding their content, and delivering useful and relevant information. The most common tasks in computer vision are classification and detection. Self-driving cars and medical image diagnosis are two of the numerous applications of computer vision. Convolutional Neural Networks are now the reference in image processing and are used for both image classification and semantic segmentation. Appendix B.3 describes such networks. **Natural Language Processing (NLP)** focuses on understanding, manipulation, and generation of natural language by machines. Question answering and machine translation are two examples of NLP tasks. Language is more complex than images in the sense that there is an order and organization in a sentence that needs to be taken into account. Appendix B.4 presents Transformer, a model architecture that is used by most NLP model. **Recommendation** is another significant field whose goal is to suggest content or products based on amassed user data, such as purchase history, clicks, conversions, and

demographics. Recommendation systems are ubiquitous in our daily digital lives, influencing everything from the products we see on e-commerce platforms to the music we listen to on streaming services. The amount of data for recommendation use cases has roughly doubled between 2019 and 2021 [Wu2022].

We selected 6 out of the 8 benchmarks available in April 2022 to represent most applications and models. The list can be found in table 2.4. At the time this thesis was published, the MLPerf Training Benchmark Suite included 16 models.

Area	Benchmark	Model	Dataset	Quality Target
Vision	Image classification	ResNet-50 v1.5	ImageNet	75.90% classification
Vision	Image segmentation (medical)	3D U-Net	KiTS19	0.908 Mean DICE score
Vision	Object detection (heavy weight)	Mask R-CNN	COCO	0.377 Box min AP and 0.339 Mask min AP
Language	Speech recognition	RNN-T	LibriSpeech	0.058 Word Error Rate
Language	NLP	BERT-large	Wikipedia 2020/01/01	0.72 Mask-LM accuracy
Commerce	Recommendation	DLRM	Criteo 1TB Click Logs	0.8025 AUC

Table 2.4: Characteristics of the selected models

The following sections briefly present each model.

2.3.1 Image classification: ResNet

Image classification is the process of assigning a label or category to an entire image based on its visual content. It has a wide range of application, from moderation in social media to detecting diseases in medical imaging.

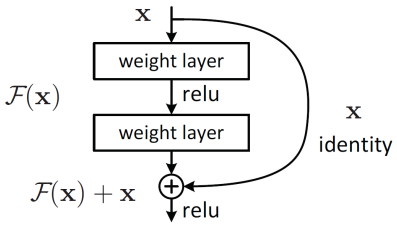
A challenge called ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Russakovsky2015] was held every year from the database creation year in 2009 until 2017 to encourage research in computer vision. A dataset of the same name was made public to support the challenge. ImageNet is a unique database with more than fourteen million images annotated for classification and detection in more than twenty thousand categories. The challenge stopped with its last winner reaching a top-5 error of 2.251%. To put those results in perspective, a human/computer performance comparison was made [Russakovsky2015]. A trained human achieved a top-5 error of 5.1% on a selected dataset.

ResNet was one of the breakthroughs from the ImageNet challenge, achieving a 3.57% error in 2015. Developed by Microsoft, ResNet is based on residual blocks which are convolution blocks with an identity shortcut connection [He2015]. It was thought to deal with the degradation of the training accuracy when adding layers. Previously, one of the greatest challenges when increasing the depth of a neural network was the problem of vanishing gradient. During back-propagation, gradients of parameters are computed thanks to the chain rule. Yet parameters have values between 0 and 1. By multiplication, gradients only decrease. After a certain number of layers, the model stops learning because the gradient update of the parameters is too low.

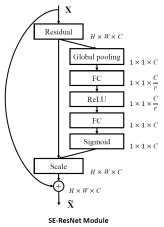
ResNet can learn to bypass a block if it increases its performance, conforming to figure 2.2a. Therefore it can't have lower performance than if there were no blocks. As a consequence, the developers were able to add hundreds of layers, and the model was still learning and improving its accuracy. The increase in the number of parameters means that the computation cost becomes more significant.

2.3.2 Object detection and Image Segmentation: Mask R-CNN

Segmentation is another major area of computer vision. It refers to the task of dividing the image into segments representing different objects. There exist two levels of granularity within the



(a) Residual block [He2015]. Relu (rectified linear unit) is an activation function.



(b) Squeeze-Excitation block [Hu2019].

Figure 2.2: Examples of novel architecture blocks

process of segmentation, namely, instance and semantic segmentation. Instance segmentation will identify each object as one instance, even if they belong to the same class of objects such as pedestrians in a traffic surveillance image, while semantic segmentation will classify all the pixels into a set of previously defined classes.

Convolutional Neural Network (CNN)s, as it was used in the Handwritten digit recognition problem, ended with a linear output that loses the spatial information. However, in semantic segmentation, each pixel of the original image needs a probability of belonging to a certain class. Region proposal networks perform well on segmentation tasks and Mask R-CNN is one of the best versions of this kind of model, outperforming previous state-of-the-art models in instance segmentation, bounding box object detection and person keypoint detection in the COCO (Common Objects in Context) 2016 challenge [He2017].

Mask R-CNN was published in 2014 by Ross Girshick from Facebook AI research [He2017]. It is an extension of Faster R-CNN which is a region proposal network (RPN) method for object detection. Two main steps compose this sort of method: a proposition of a region of interest (ROI) and the evaluation and selection of this proposed ROI. A schema of the architecture can be found in figure 2.3a.

First, a classic CNN such as ResNet produces feature maps that are given to the RPN. Thus instead of going through all the possible regions from raw input, it only scans the features, reducing the complexity. The RPN produces two outputs: the class of the region (object or background) and the offset (coordinate) of the bounding box.

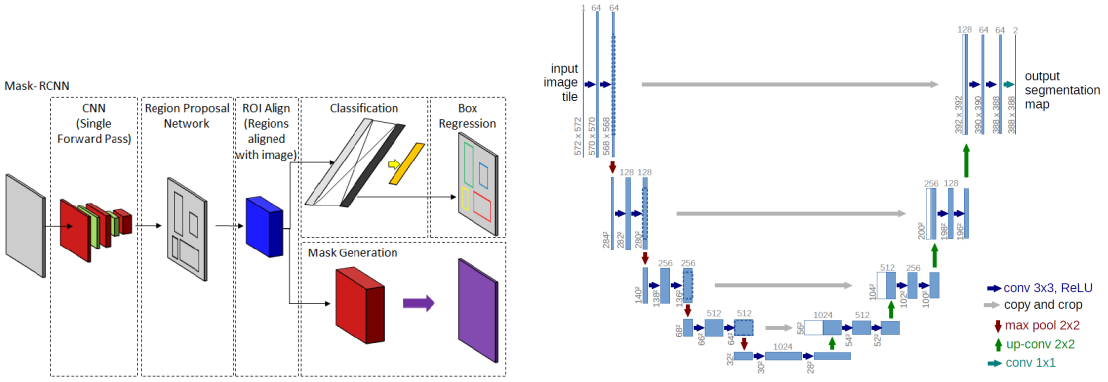
The second stage is a refinement of those outputs. ROI alignment exists because the proposed regions come from various scales when the input of the following fully connected layer needs to have a fixed size. To produce a segmentation mask, the output also needs to have the same shape as the original input. Through a pooling layer and interpolation, the ROI is resized. Then, in parallel to predicting the class and the box offset, Mask R-CNN produces a binary mask thanks to a Fully Convolutional Network (FCN).

2.3.3 Object detection and Image Segmentation: 3D U-Net

One of the issues of Mask R-CNN is the loss of context. Indeed, when an ROI is selected, all the surrounding information is not taken into account to produce the mask. U-Net offers a solution to this challenge. It was published in 2015 [Ronneberger2015]. It is composed of a several-level contracting path and a corresponding expansive path (U shape), as can be seen in Figure 2.3b. The contracting path uses FCN and pooling layers to produce several levels of feature maps. The expending path up-samples features maps and concatenates them with the ones from the contracting path. This is what allows the model to keep context information and to have the best performances in biomedical images where the context is essential.

In biomedical image segmentation, U-Net is the most used model [Ghosh2019].

For 3D U-Net as defined in MLPerf, the filter sizes of the convolutional layers are 256, 128, 64, 32, and 16 in the contracting path order. In the expanding path, the decoder blocks are up-



(a) Architecture of Mask R-CNN [Ghosh2019] *(b) Architecture of U-Net [Ronneberger2015]*

Figure 2.3: Segmentation model architectures

sampled instead of transposed, which means that inputs are resized by interpolation and that no additional weights are learned.

2.3.4 Language understanding: BERT

The first critical step in NLP is to make language understandable in a numerical way. Words are embedded into dense vector representations that words with similar meanings are close in the representation space. This is done by minimizing the context distance between words. Encoding information is an unsupervised task and thus can be used on a large unlabeled corpus. Additionally, resulting models can be easily applied to many tasks relying on natural languages by adding layers and fine-tuning thus reducing the computational cost.

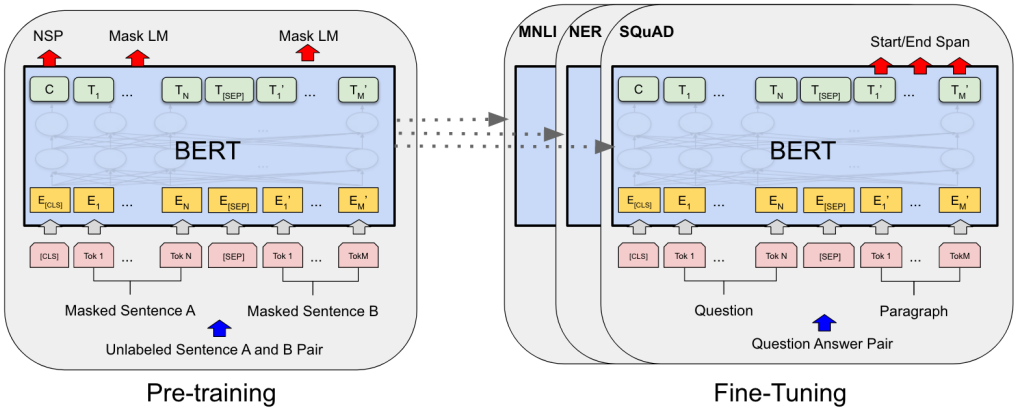


Figure 2.4: Pre-training and fine-tuning procedures for BERT.

BERT was published by Google Brain in 2019 [Devlin2019]. It stands for Bidirectional Encoder Representations from Transformers. It is composed of two parts, as can be seen in Figure 2.4: Pre-Training and Fine-Tuning. Pre-Training is unsupervised and consists of two steps. First, it randomly masks words in the text and trains BERT to predict the masked word based on the surrounding context. This helps BERT understand the meaning and relationships between words. Secondly, it takes pairs of sentences as input and trains BERT to predict if the second sentence follows the first sentence logically. This helps BERT understand the flow and connection between sentences. The other particularity of BERT is that it is bidirectional. Instead of going through the sentence from left to right, it gets context from both the words on the right and the left of the given token. It is one of the main differences between BERT and Generative Pre-trained Trans-

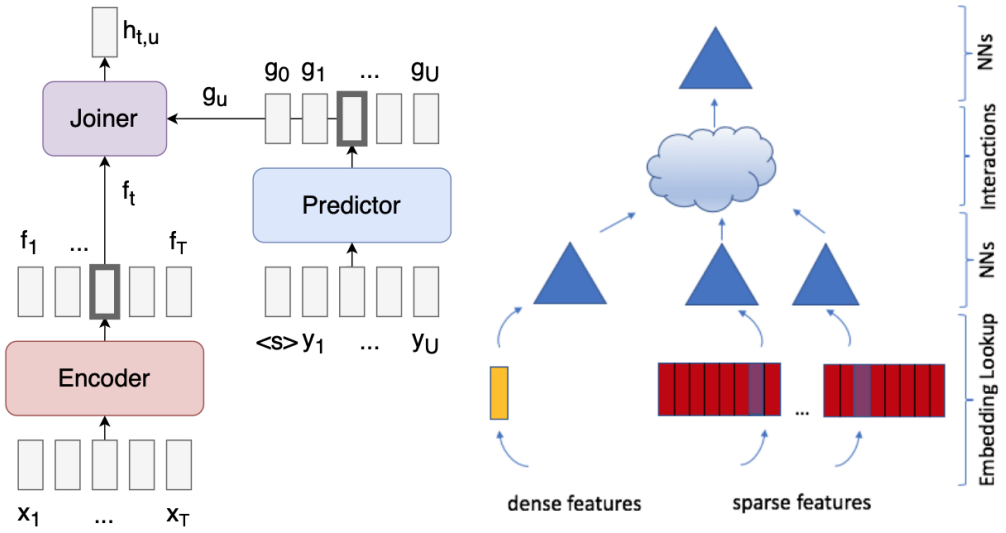
former (GPT) [Radford2018]. BERT is known for its strength in tasks like question answering and sentiment analysis, while GPT-3 excels at creative text generation.

MLPerf includes BERT-Large. BERT-Large consists of 24 layers with 1024 Transformer blocks and 16 self-attention heads. In total, it has 340 million parameters.

2.3.5 Speech Recognition: RNN-T

Speech Recognition allows computers to convert spoken language into text. Examples of applications are virtual assistants and automatic transcription. Although related to language understanding, speech recognition has several characteristics that make it hard to directly use models presented before. First, it is monotonic, meaning that inputs and outputs should somehow be aligned. Speech Recognition models are often run online and need a low latency.

RNN-T stands for Recurrent Neural Network Transducer [Graves2012]. It was published by a researcher from the University of Toronto in 2012. It relies on a Prediction Network and a Transcription Network. The prediction network is a LSTM layer that attempts to model each element of the input given the previous ones while the transcription network is the encoder of the input, as can be seen in Figure 2.5a. A joiner was added in follow-up work [Graves2013] as a fully connected layer that takes both network outputs as inputs. The need to compute probabilities for each possible alignment leads to a significant memory consumption in training [Lugosch2020]. However, this is not the case for inference, making RNN-T faster than attention-based models.



(a) Architecture of RNN-T [Graves2013]. *(b) Architecture of DLRM [Naumov2019].*

Figure 2.5: Architectures of RRN-T and DLRM.

2.3.6 Recommendation: DLRM

DLRM was developed by Facebook (now Meta) and published in 2019 [Naumov2019]. It is currently deployed at the scale of the company and continuously trained with user data. Inspired by matrix factorization, it uses deep learning to enhance its performance. Figure 2.5b shows a schematic view of the different blocks DLRM is composed of. Embedding is used to reduce the dimensionality of user and product data. Interactions between features and clients are inspired by matrix factorization and are referred to as "Factorization machines". This is done by taking the dot product between all pairs of embedding vectors and processing dense features. Several

Multi-Layer Perceptrons (MLP) are added to help process data. DLRM is known for the significant number of learnable parameters, but also its parallelizable characteristics. Meta uses model parallelism for the embeddings and data parallelism for the MLPs, resulting additionally in heavy communication costs. DLRM consists of both a bottom MLP for processing dense features consisting of three hidden layers with 512, 256, and 64 nodes, respectively, and a top MLP consisting of two hidden layers with 512 and 256 nodes. DLRM has approximately 540M parameters.

2.4 Functional units

A functional unit (FU) describes a quantity of a product or the quality of a service based on the expected performance it delivers in its end-use application. The performance criteria in the ML training case is the model's quality on the validation dataset. In this thesis, we define 6 FUs, one for each model described in Table 2.5. For a fair comparison, FUs should be assessed against a single reference year. Models are initialized with random weights before the training starts.

ResNet-50	Train ResNet-50 on ImageNet until it achieves a 75.90% classification accuracy.
3D U-Net	Train 3D U-Net on KiTS19 until it achieves a 0.908 Mean DICE score.
Mask R-CNN	Train Mask R-CNN on COCO until it achieves a 0.377 Box min Average Precision and a 0.339 Mask min Average Precision.
RNN-T	Train RNN-T on LibriSpeech until it achieves a 0.058 Word Error Rate.
BERT	Train BERT-large on Wikipedia 2020/01/01 until it achieves a 0.72 Mask-LM accuracy.
DLRM	Train DLRM on the Criteo 1TB Click Logs until it achieves a 0.8025 Area Under ROC Curve (AUC).

Table 2.5: Functional unit

Each FU is evaluated on an Apollo node. The ResNet-50 FU is evaluated on both an Apollo node and a Jetson node to enable a comparison of both infrastructures. The systems are limited to the computing node used to perform the training.

The BERT FU is assessed when run on 1, 2, and 4 Apollo nodes, to study the effect of multi-node learning on the electricity consumption.

Chapter 3 compares tools that can be used to measure the electricity consumption of a computing node and selects one suitable for our methodology. For each FU, the electricity consumption as defined in Section 2.1.2 is studied in Chapter 4 while the environmental indicators presented in Section 2.1.3 are assessed in Chapter 5.

3

Measuring the electricity consumption of computing infrastructures

Contents

Accurately determining the environmental impact of AI training demands precise measurement of the electricity consumed by the underlying computing infrastructure. This thesis proposes a methodology based on precise power and energy measurements, necessitating a tool capable of high frequency and accuracy. To provide valuable insights, the tool needs to collect and analyze data at a finer level of detail. The purpose of this chapter is to give an overview of computing infrastructure electricity measurement, to compare existing measuring tools, and justify our tool choice.

This chapter builds upon joint work presented in [Jay2023]. While the experimental results and analysis presented here are the author's contributions, the majority of the text was co-written. The study, originally conducted in 2022, has been updated to incorporate the tool employed throughout this thesis. Tools that did not report GPU metrics and Energy Scope, due to its lack of a public version control system and associated reproducibility and transparency concerns, were excluded.

The most mature and less intrusive way to measure the power consumption of a computing node is through the use of physical power meters. However, they require the deployment of an additional measuring infrastructure. In addition, data must be collected and made available using specific software often imposed by the power meter supplier. Finally, a physical power meter only measures the overall consumption of the computing node. It does not detail the consumption of the various components or services launched on this computing node.

On the other hand, numerous power models and internal interfaces have been developed in order to provide consumption metrics at multiple levels. They are able to provide energy consumption data with granularities ranging from the overall consumption of the computing node to the consumption of a single operating system process. Moreover, these technologies are already available or can be implemented on already existing systems, not requiring any additional financial investment or specific hardware setup.

Nevertheless, implementing power models and using these internal interfaces directly to retrieve power consumption metrics requires detailed knowledge of the underlying hardware being used. A multitude of tools has been developed in order to facilitate this task. In this chapter, we call these tools software-based power meters and divide them into three categories: energy calculators, energy measurement software, and power profiling software. Energy calculators estimate energy consumption using thermal design power (TDP)-based modeling. Energy measurement and power profiling software can report respectively the energy and power consumption of the CPU (Central processing unit), DRAM (Dynamic random-access memory), and GPU (Graphics processing unit) as retrieved from the internal interfaces. Some of them additionally implement power models in order to give an estimation of the consumption at the level of a process, a container, or a virtual machine. However, each tool is not equal in terms of the available feature set, the supported sampling rates, or the quality of the estimation. We have found that it can be difficult to choose the right tool for a specific need.

In this chapter, we study a selection of software-based power meters from various angles such as hardware compatibility, underlying technologies used, estimation models used, intrusiveness, quality of estimation, quality of documentation, their strengths, and their limitations. In order to evaluate some characteristics such as the quality of the estimation and the intrusiveness, we executed a set of benchmarks on a computing node equipped with high-precision external power meters.

Section 3.1 details the technologies and modeling behind the tools we selected. Section 3.2 describes the tools we chose to include in the study, the qualitative criteria we compared them with, and a comparative table. Section 3.3 presents our experimental setup. Section 3.4 shows our quantitative results. Section 3.5 concludes the chapter.

3.1 Background

The electricity consumption can be studied with two metrics: the total energy consumed by the execution of a program and the evolution of the power that was consumed during the execution. Power is the time rate of doing work or delivering Energy. It is usually expressed in Watt (W) and energy in Joule (J). Electrical power measurement relies on two metrics: current (in amperes) and voltage (in volts). It is characterized by Equations 3.1 and 3.2.

$$\text{Energy (J)} = \text{Power (W)} \times \text{Time (seconds)} \tag{3.1}$$

$$\text{Power (W)} = \text{Current (A)} \times \text{Voltage (V)} \tag{3.2}$$

Electricity is a well-established scientific field in which current and voltage are fundamental concepts. Current can be measured using a coil whose magnetic field is directly affected by the current. This effect can be evaluated thanks to the movement of a magnet, for example. This amperemeter is in series in the circuit and the coil is placed in parallel to a shunt resistor that enables only a small fraction of the current to be taken from the circuit, thus having close to no impact. This is called a shunt meter. One alternative is to measure the Hall effect, as most external power meters do. Voltage is measured similarly to the current, but the voltmeter is placed in parallel to the circuit. Both are needed to measure the apparent power but it is possible to calculate one from the other if the characteristics of the electrical circuit such as its resistance are known.

Computing devices are designed to work under a given power called the Thermal Design Power (TDP). It is provided by the manufacturer and corresponds to the maximum amount of heat that can be generated by a component under a steady workload. The TDP of a standard CPU is around 100 W and the TDP of GPUs now reaches close to 1000 W, depending on the model. Figure 1.3 shows the evolution of the TDP in the past decades. The power of a computing node is continuously monitored in order to cap the hardware frequency when the power reaches the TDP. This means that most node components are equipped with shunt meters. Values are stored in registers and thus not directly available to the user.

This section starts by detailing how such registers can be accessed. Then it lists other tools that can be used to measure power and energy in computing nodes. It additionally presents a methodology to estimate power consumption based on hardware characteristics and usage measurements. Ultimately, it reveals the additional power required to operate the computing node at the data center level and how it can be calculated.

Measuring the electricity consumption of digital hardware can affect the system on various levels, from measurements to collecting and processing the data. The overhead needs to be taken into account when selecting a tool. The differences between tools are also due to the collection of data. If the equipment is placed outside the motherboard, collection requires precise dating, additional communication devices, and database implementation.

3.1.1 Hardware sensors and software interfaces

Computing node vendors and component manufacturers embed digital sensors, onboard measurement circuits, and interfaces that measure the power consumption of the entire system, the processor socket, the memory, and other computing node components. The scope and the precision depend on the model and the manufacturer. Some of these sensors and interfaces are already integrated into widely used computing nodes and energy consumption data is available through performance counters or vendor-specific APIs, making this method more cost-effective and user-friendly. For example, Intel, AMD, IBM, and Nvidia have released technologies to measure and control the power consumption of computing node components.

Intel CPU RAPL

The RAPL (Running Average Power Limit) interface was introduced by Intel in 2011 [David2010] in the Intel Sandy Bridge architecture. It allows the energy consumption estimation at fine granularity with high sampling rates and the power consumption capping of various parts inside the processor. The first implementation of RAPL used a software power model to estimate energy usage based on "a set of architectural events from each Intel architecture core, the processor graphics, and I/O" [Rotem2012].
The second implementation of RAPL, introduced with the Haswell architecture, is based on fully integrated voltage regulators and enables actual power measurement, improving the accuracy of RAPL measurements [Hackenberg2015].
RAPL reports energy consumption and can limit the power consumption on different levels or power domains: entire CPU socket (PKG), all CPU cores (PP0), integrated graphics (PP1), dynamic random-access memory (DRAM), and entire SoC (PSys). The availability of power domains may vary between architectures and processor models. RAPL energy consumption reports can be accessed through MSR (Model-specific registers). The values in these registers are expressed in energy units and represent the energy consumed in microjoules since the processor was started.
RAPL registers have a high update frequency and low-performance overhead [Khan2018]. Their energy counters are updated approximately every 1 ms (1000 Hz). RAPL is an always-running interface and starts to work when the processor boots.
However, the RAPL interface lacks detailed low-level implementation documentation. Thus the exact methodology of the RAPL calculations remains unknown. The RAPL registers are not updated precisely every 1 ms [Weaver2012], have no timestamps attached [Hackenberg2013], and are updated in a non-atomic way [Khan2018]. The RAPL registers are limited to 32 bits and can overflow. This must therefore be taken into account when reading the RAPL values directly.

AMD APM

In Bulldozer architecture (Family 15h), AMD introduced an on-chip energy consumption estimation called Application Power Management (APM). AMD APM gives an average power for the last time frame of approximately 10 ms on the Bulldozer 4P architecture. The accuracy of APM depends on the system configuration and on the workload. For instance, it shows highly inaccurate power assumptions during sleep modes [Hackenberg2013].

AMD RAPL

In Zen architecture (Family 17h), AMD replaced APM with RAPL (Running Average Power Limit) [AMD2017]. The AMD RAPL has a similar implementation to the Intel RAPL while using different registers. Schone et al. [Schöne2021] studied the functionality and the accuracy of AMD RAPL implementation. The accuracy study shows that this implementation is more similar to the first software model-based Intel RAPL implementation and shows inconsistent results in some cases.
In addition, the AMD implementation does not provide a DRAM power domain and DRAM power consumption is not fully included in the package domain. As, depending on the input data, the power consumption of workload may differ by more than 18% for the whole system [Schöne2019], the lack of full DRAM power consumption reporting heavily impacts AMD RAPL accuracy for data-dependent workloads.

Nvidia GPU NVML

NVIDIA provides users with an API called NVIDIA Management Library (NVML) [Corporation.2019]. It provides access to GPU device metrics such as current utilization, temperature, and power draw. According to the official documentation, the method `nvmlDeviceGetPowerUsage()` returns the current power draw within an accuracy of 5% of the GPU and its associated circuitry (e.g. memory). Retrieval of current power draw is supported by Nvidia GPUs of the Fermi generation and newer. Even if there is no information on whether the power is measured or estimated, several works mention that it is measured from onboard sensors [Arafa2020, Sen2018].

3.1.2 Intra-node devices

Intra-node devices can be defined as equipment placed in the motherboard. These types of devices include Baseboard Management Controllers (BMC) embedded in computing nodes, devices placed between a computing node's power supply and main board as PowerMon2 [Bedard2010], and devices for component-level instrumentation as PowerInsight [Laros2013]. These devices can provide accurate consumption of individual computing node components [El Mehdi Diouri2013, Diouri2014] but need an additional financial investment and lack user-friendliness.

3.1.3 External Devices

External devices, commonly known as power meters, are not embedded in computing nodes. They are generally positioned between the wall socket and the power supply unit of a computing node. These devices include external power meters and Power Distribution Units (PDUs) with measuring capabilities. Nowadays, numerous external power meters are available on the market, such as OmegaWatt or HPE iLO intelligent power distribution units (iPDUs).

External measurement has been used for a long time, provides users with accurate power consumption values, and is the only way to measure the consumption of the entire computing node. As the measurement is purely external, this method has little impact on the measured system. This method has drawbacks. Firstly, external devices report only the consumption of the entire computing node, without any details at the level of different computing node components or running processes. Secondly, measuring the energy consumption of large-scale systems using this solution can be very cost-ineffective as multiple expensive devices need to be purchased. Thirdly, most devices available in the market have low measurement rates, which gives too coarse-grained results that may not be suitable for some use cases, for example profiling the power consumption of the various phases of an application. Finally, these devices rarely provide data in a user-friendly form as they usually use serial interfaces and need additional software to get values.

3.1.4 Usage-based modeling

With minimum information on the system, the power consumption can be modeled using the Thermal Design Power (TDP). Components such as Intel processors may exceed the maximum TDP for a limited amount of time when running with Turbo Boost or using Intel Advanced Vector Extensions [Intel2023]. Despite this, the TDP value can be used as a reasonable approximation of component power consumption at maximum use.

So, if the TDP, average CPU usage, and total execution time are known, we can use the following simple equation to estimate the total CPU power consumption.

$$\text{Energy} = \text{TDP} \times \text{Avg. Utilization} \times \text{Execution Time} \tag{3.3}$$

This type of modeling has many advantages. It is very simple, can give a coarse-grain idea of power consumption, and can be done offline. However, this model assumes that component consumption will never exceed its TDP value and tends to be inaccurate for inconsistent or non-intensive workloads.

3.1.5 The extra power spent by computing infrastructures

Much electricity is lost from production to where it is used from loss in transportation and transformation.

Digital devices rely on Alternative Current (AC) meaning that both current and voltage oscillate between their positive maximum and their negative symmetric. An offset between both waves leads to a decrease in instant power and is due to the inductive or capacitive qualities of the devices. This offset depends on devices and is called the **power factor**. Manufacturers provide it. The resulting power is referred to as the **real power** as opposed to **apparent power**. Most circuits have a power factor higher than 0.9 which can be a significant factor for energy measurement. It is worth noticing that digital devices internally operate in continuous current thanks to a converter in the power supply unit, leading to power loss as well. At the aggregated level of a data center, [Ahmed2021] shows that the internal power supply system can consume more than 10% of the rated IT load.

Additionally, many devices are needed for the processing unit and other useful components to be used. This energy consumption should be taken into account as well.

The power consumption of data centers mainly comes from the compute nodes, the cooling system, and the power supply system. The consumption of non-computing components in data centers is represented by the Power Usage Effectiveness (PUE). It is the ratio between the total energy consumed and the energy consumed by IT infrastructure, the perfect value being 1. According to the Uptime Institute Global Data Center Survey Results 2024, the average world PUE was 1.56. However, the main data center companies report a much lower value, as can be seen in Table 3.1.

Google	Amazon	Meta	Microsoft
1.06-1.10	1.07-1.15	1.09	1.18

Table 3.1: Examples of PUE from data center companies in 2023, as reported by their 2023 sustainability reports.

This extra power should be taken into account by measurement tools to provide accurate metrics.

3.2 Software-based power meters

Most of the methods previously introduced require either specific hardware equipment or detailed knowledge of the underlying hardware being used. They can't be used as they are and often require additional implementation. For example, integrated technologies need to be queried and processed in order to get energy metrics. Fortunately, software packages or programs were developed to simplify the process for the users.

3.2.1 Selected tools

We selected tools able to measure or estimate energy consumption with varying levels of granularity that are suited for AI workloads relying on Graphic Processing Units. The tools are grouped into four categories: (i) External and intra-node devices; (ii) Power profiling software; (iii) Energy measurement software packages; (iv) Energy calculators. Each group corresponds to a specific

purpose therefore to different requirements and setups. This section describes the most important characteristics of those tools. The version of each studied tool is indicated in parentheses.

External and intra-node devices

We selected two power measuring devices that are available at large-scale test beds for experimental research called Grid'5000 [Balouek2013], and that we used for our experiments. They are sending data to a database using the Kwollect metric collection system [Delamare2021].

OmegaWatt has a maximum sampling frequency of 50 Hz and a precision of 0.1 W. It is placed between the node and the power delivery device.

Baseboard Management Controller (BMC) reports the power consumption of the entire computing node with a sampling frequency of 0.2 Hz.

Power profiling software

Such tools are able to return the power profile of a program.

Alumet (Preliminary version) [Raffin2023] is a Rust program collecting data from NVML and RAPL for each node component and handling register overflows. The preliminary version used in this thesis was later updated to include more features [1]. A plugin API allows to extend the capabilities of the tool. The user can build his measurement tool by selecting the plugins that suit his needs, or by creating a new plugin with a couple of lines of code. The measurement pipeline can be reconfigured on the fly, without restarting the agent. For instance, it is possible to adjust the acquisition frequency in real time. It was developed to be lightweight and easily configurable.

Energy measurement software packages

Energy measurement software packages report the total energy consumed by the computing node during the execution of the program. The three software packages we selected are all Python packages based on Intel RAPL and Nvidia NVML interfaces. They monitor the energy consumption between a start and an end pointer to insert in the code. In addition to returning the energy, they fetch the CO2 signal from the machine location and its IP address so it can find the energy efficiency and compute the carbon footprint. They also include the PUE if its value is available.

Carbon Tracker (version 1.1.6) [Anthony2020] was developed in 2020. It uses powercap interface files in order to access RAPL counters and NVML python library in order to get CPU and GPU consumption metrics. It can be integrated into a machine learning model training to predict the energy consumed by the whole training from the training of one epoch. It assumes that the energy consumed in one epoch doesn't change through training. The resulting logs are readable and can be processed by a code interface. It can be used in a Jupyter notebook.

Code Carbon (version 2.0.0) [Schmidt2021] retrieves consumption information from RAPL files of powercap interface, Power Gadget, and NVML python library between checkpoints in the code. It also automatically retrieves the TDP coefficient from an internal database if Intel RAPL and Power Gadget are not available. A dashboard is available with Comet to visualize the energy of various experiments.

Experiment Impact Tracker (version of June 4, 2021) [Henderson2020] distinguishes itself by separating the consumption of the Python script execution process from the rest of the system. The resulting data can be accessed by a code interface or a JSON file.

[1] https://alumet.dev/

Energy calculators

Energy calculators are software-based power meters available on the web relying on TDP usage-based modeling to compute the energy consumed by equipment.

ML CO2 impact (version of Jul 5, 2022) [Victor Schmidt] was developed for cloud-based experiments and only supports one piece of equipment at a time. The required parameters are hardware type, hours used, cloud provider, and region of computing. Only the most common hardware type is in its database. By selecting "private provider", you can fill in the carbon efficiency of your equipment and region and the percentage of carbon offset. However, you can only select one piece of equipment at a time.

Green Algorithms (version 2.2) [Lannelongue2021] relies on more parameters and provides the user with more metrics than the carbon footprint: runtime, type of cores, number of cores, model, memory available (in GB), platform used for the computations (cloud provider, personal computer, local server), region, usage factor of the CPU, power usage effectiveness (PUE) if local server was selected, and pragmatic scaling factor. The last item is the number of times the experiment was run, to account for example for hyper-parameter search in machine learning model training. This calculator provides the user with various metrics such as the carbon footprint, the energy needed, the number of tree months to sequester the carbon emitted, and the equivalent number of kilometers in driving a car or flying in a commercial plane.

3.2.2 Other available tools

perf, **PowerAPI** [Spirals2020] and **Scaphandre** (version 0.4.1) [Petit2020] are power profiling software that were included in the study but don't report GPUs metrics. **nvidia-smi**, **Likwid**, [Gruber2021] **PowerTOP**, and **powerStat** are command line tools difficult to use to understand the energy consumption of a program. **powermetrics** and **Intel Power Gadget** are supported only by MacOS or Windows operating systems. **PAPI** [Weaver2012] is an interface to power-related performance counters. Other tools such as **PyJoules** [Spirals2019] and **Cumulator** are Python packages highly similar to those studied in this work. **Kepler** is another promising software-based power meter developed to work exclusively in the Kubernetes environment, which is beyond the scope of this work.

3.2.3 Comparison criteria

We describe the various criteria we used to qualitatively compare the selected software-based power meters in Table 3.2. Those criteria were selected to best differentiate the tools.

We divided the criteria into four categories: (1) Development, by whom and when the tool was developed; (2) Environment, what are the hardware requirements and in which environment they can be used; (3) Functionality, how each tool works internally; (4) User-friendliness, how easy it is to use and configure.

Development

Both metrics give information on the developer and when the tool was developed.

- **Origin** Who developed it.
- **First / latest release date** When the tool was released for the first time and when was made the latest release (as of 6 September 2022).

Environment

The environment metrics provide details on the hardware requirements and in which environment they can be used.

- **Hardware compatibility** Technologies that must be available at the computing node to run tools.
- **Scope** The granularity at which the solution provides consumption values. Levels: machine, resource (CPU, GPU, DRAM(CPU)), cgroups, processes.
- **Job management system support** Whether the tool has built-in job management system support (OAR, SLURM).

Functionality

Each tool has different functionalities and relies on different mechanisms.

- **Hardware technology used** The technology used to measure or estimate energy consumption. For example, RAPL and NVML.
- **Software power model used** If a model is used on top of the hardware technology. For example, to estimate the energy consumed at the process level.
- **Default sampling frequency** The frequency at which the solution samples energy consumption values by default.
- **Online reporting** If the data are available in real-time or if it is provided only at the end of the execution.
- **Power profiling** If the solution has power profiling capabilities in the form of time series or if the solution reports only total resulting energy consumption.

User-friendliness

The user-friendliness evaluates how easy it is to use and configure a tool.

- **Configurability** "Poor", "Fair" or "Good" depending on the number of configurable parameters supported among the following list: acquisition frequency, result data form, used model, and result data (power, energy, or carbon emission).
- **Availability of source code** Whether the source code of the tool can be found online and with what license the tool is distributed.
- **Ease of use** How easy it is to install and use the solution. "Poor" if one needs an understanding of the architecture of the tool or its environment to be able to configure it and collect its results. "Fair" if the tool doesn't need any architecture-dependent configuration but an additional mechanism is required to retrieve results. "Good" if no configuration or additional mechanism is needed to retrieve results. "Very good" if no installation or software skills are required.
- **Quality of documentation** "Poor" if the documentation is not sufficient to use and configure the tool. "Fair" if the documentation is sufficient, but an effort is needed to understand how it works. "Good" if the available documentation addresses usage questions such as parameter settings [Bannour2021].
- **Resulting data format** The format of the result data that is provided. For example, as a value stored in a code variable (Code) or written into a database back-end (Prometheus, InfluxDB, MongoDB, Riemann, Warp10), sent via a socket connection (Socket), a text file (JSON, CSV, Latex), available on a web page (Web) or just printed in the standard output (Stdout).

- **Data visualization possibilities** Dashboards (Grafana, Comet, custom) or online web resources.

3.2.4 Qualitative comparison

In table 3.2, we compare the selected tools on the criteria introduced in the previous section.

External and intra-node devices offer broad compatibility and real-time metrics but are limited to machine-level data and lack configuration flexibility. Among the remaining tools, a common trade-off emerges between user-friendliness and configurability, with simpler interfaces typically providing fewer details or features. None of the software-based power meters analyzed support job management, a critical feature for large-scale experiments. Furthermore, data formats and visualizations exhibit significant heterogeneity across tools.

Some tools have unique features. Notably, Alumet is the sole option for real-time data collection. Code Carbon uniquely provides TDP-based estimation as a fallback when NVML and RAPL are unavailable. Unfortunately, Experiment Impact Tracker and ML CO2 Impact are no longer actively maintained.

3.3 Experimental setup

After qualitatively comparing the software-based power meters, we conducted a group of experiments to verify how the selected tools work, evaluate the quality of their outputs, and quantitatively compare them.

3.3.1 Environment

Infrastructure We executed all experiments on a machine from the Gemini cluster of large-scale test beds for experimental research Grid'5000 [Balouek2013]. This cluster was selected because it contains nodes with multiple recent GPUs supporting power metric retrieval with Nvidia NVML, two CPUs with the second implementation of Intel RAPL, and high-performance external power meters.

The nodes in this cluster have the following specifications:

- System model: Nvidia DGX-1
- CPU: 2 x Intel Xeon E5-2698 v4 (Broadwell, 2.20GHz, 20 cores/CPU)
- Memory: 512 GB
- GPU: 8 x Nvidia Tesla V100-SXM2-32GB (32 GB)

External power meter The power consumption of each node in the Gemini cluster is individually monitored by an Omegawatt [OmegaWatt2018] power meter. This power meter has a maximum sampling frequency of 50 Hz and a precision of 0.1 W. We used them with a sampling frequency of 1 Hz.

BMC In the Gemini cluster, each node has a BMC that reports the power consumption of the entire host system with a sampling frequency of 0.2 Hz.

Table 3.2: Qualitative comparison of selected software-based power meters.

	External and intra-node devices		Power profiling software	Energy measurement software packages				Energy calculators		
	OmegaWatt	*BMC*	*Alumet*	*Code Carbon*	*Experiment Impact Tracker*	*Carbon Tracker*	*Green Algorithms*	*ML CO2 Impact*	
Developement									
Origin			Eviden, LIG	MILA	University of Standford	University of Copenhagen	University of Cambridge	MILA	
First (latest) release date			Mar. 2023 (May 2024)	Nov. 2020 (Jun. 2024)	Dec. 2019 (Jan. 2020)	Apr. 2020 (Sept. 2023)	Jul. 2020 (Apr. 2023)	Aug. 2019 (Jul. 2022)	
Environment									
Hardware compatibility	Any	Any	Intel RAPL, Nvidia NVML	Any	Intel RAPL, Nvidia NVML	Intel RAPL, Nvidia NVML	Any	Any	
Scope	Node	Node	CPU, DRAM, GPU	CPU, DRAM, GPU	CPU, DRAM, GPU, process	CPU, DRAM, GPU	Node	Node	
Job management support			No	No	No	No			
Functional									
Hardware technology used			RAPL, NVML	RAPL, NVML, TDP	RAPL, NVML	RAPL, NVML	TDP	TDP	
Software power model used					GPU, CPU and RAM usage-based				
Default sampling frequency (Hz)	1	0.2	2	1/15	1	0.1			
Online reporting	Yes	Yes	Yes	No	No	No	No	No	
Power profiling	Yes	Yes	Yes	No	No	No	No	No	
User-friendliness									
Availability of source code (License)			Yes (EUPL 1.2)	Yes (MIT)	Yes (MIT)	Yes (MIT)	Yes (CC-BY-4.0)	Yes (MIT)	
Ease of use	Poor	Poor	Good	Good	Good	Good	Very good	Very good	
Quality of documentation			Good	Fair	Fair	Good	Good	Fai	
Configurability	Fair	Poor	Good	Poor	Quite good	Poor	Poor	Poor	
Resulting data format	HTTP end-point	HTTP end-point	CSV	CSV	JSON, Code	File, Code	Web	Web, Latex	
Data visualisation possibilities	Grafana (Kwollect)	Grafana (Kwollect)		Comet			Graphs on the web page		

Operating system and configuration For all experiments, we used the minimal variant of Ubuntu 20.04 available on the Grid'5000 testbed. In order to increase consumption stability and the consistency of our results, we have disabled Hyper-Threading and Turbo-Boost technologies and set the CPU frequency to the maximum supported. We also installed an Nvidia GPU driver when relevant, with default power management configuration.

3.3.2 Selected benchmarks

In order to evaluate software-based power meters, we executed benchmarks representative of typical workloads, well-known by the community, and implemented for GPUs. For this purpose, we chose the NAS parallel benchmarks implemented for GPU in CUDA [Araujo2021].

We selected three NAS benchmark kernels in order to simulate an intensive use of different computing node components. The first kernel is the EP (Embarrassingly parallel) kernel that generates pairs of Gaussian random deviates thus making intensive use of the CPU (and respectively GPU). The second kernel is the MG (Multi-Grid) kernel. This kernel performs a V-cycle multigrid algorithm and tests both short and long-distance data communication, thus is memory intensive. The third kernel is the LU (Lower-Upper Gauss-Seidel solver) kernel. This kernel is a pseudo-application that performs a synthetic computational fluid dynamics (CFD) calculation. LU is less CPU intensive than EP but also uses memory.

Every parallel NAS benchmark has a class that can be thought of as a problem size. For instance, for the MG benchmark, classes from A to E will have increasing grid size, iteration number, and therefore execution time. The class for each benchmark kernel was chosen empirically for our experiments in order to have suitable execution times.

3.3.3 Experimental protocol

We evaluated the consumption results given by the software power meters listed in Section 3.2.1 while executing the NAS benchmark kernels presented in Section 3.3.2. We added a one-minute interval between each benchmark execution to let the computing node component cool down after each benchmark run and prevent the power consumption of subsequent executions from being impacted.

Tools configuration Most of the software-based power meters studied in this work were used with the default configuration. However, some of them have multiple configuration possibilities or the default values are not suitable for our experiments. **Experiment Impact Tracker** and **Carbon Tracker** multiply the computed energy by a Power Usage Effectiveness (PUE) ratio by default. It was systematically removed.

Reproducibility In order to make the results more reliable, all the experiments were carried out ten times. We have automated the execution of all the experiments as well as the processing of their results.

3.4 Results

3.4.1 Computing node components power profile

Alumet is the sole tool with the additional feature of reporting power profiles of components of the computing node: CPU, DRAM, and GPU.

Figure 3.1 shows the power profiles for every CPU, DRAM, and GPU as provided by the above-mentioned software. We can also mention that the DRAM consumption reported by each tool seems to reflect the actual DRAM usage by each benchmark. For memory-intensive benchmarks such as MG, we see the change in DRAM consumption reports while the benchmark is running. Whereas, for the EP benchmark which uses no memory, the DRAM consumption data remains unchanged. It confirms that the benchmarks are GPU intensive, except for MG which has a phase when only the CPU and the DRAM are working. Within this phase, the intensity in memory seems to prevent the GPUs from working at maximum utilization.

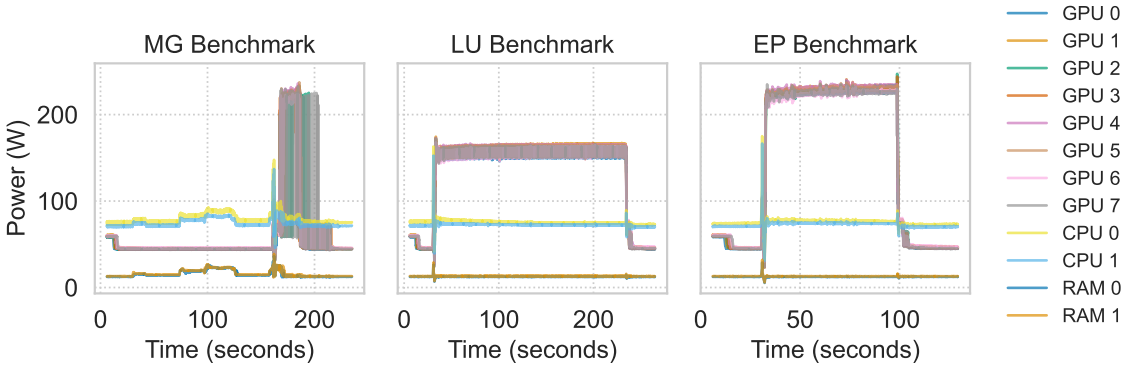


Figure 3.1: Power profiles of specific components: DRAMs, CPUs, and GPUs.

3.4.2 Comparing total computing node power profile

Figure 3.2 shows the GPU benchmark profiles as reported by the external power meter, the BMC, and Alumet.

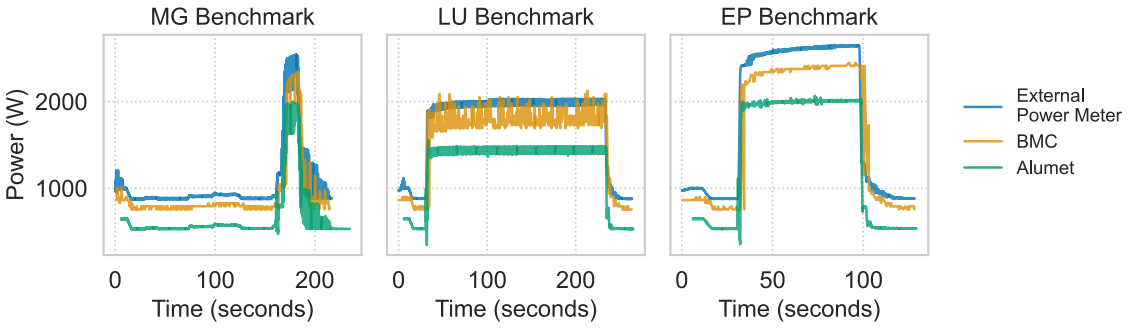


Figure 3.2: Power profiles provided by Alumet, BMC, and the External power meter on the GPU benchmarks.

We can see that the evolution of the Alumet power profile is visually similar to the evolution of the external power meter and BMC profiles, despite a non-negligible offset between the three instruments. This offset depends on which component of the computing node is included in the reports of each tool. The external power meter is installed between the computing node and the wall socket. Thus, it reports the overall consumption of the computing node with all its components. The BMC is installed internally after the power supply unit (PSU) unit and the reporting scope differs according to its implementation. The values given by BMC are therefore predictably lower. Alumet is based on Intel RAPL and Nvidia NVML, which only includes the consumption of the CPU, DRAM, and GPU components. The consumption of other components, such as fans, storage, and network interfaces, is not included.

The evolutions of the benchmarks are well captured by each tool. For the LU and MG benchmarks, the tool gives a more variable power profile than for EP. This variability is due to the lower power consumption of the GPU during the waiting times introduced by the memory operations performed by the LU and MG benchmarks. However, each tool captures them differently. Tools with a higher acquisition frequency like Alumet and the external power meter better capture consumption changes when running the benchmark, resulting in a more precise power profile. Tools with low acquisition frequency like the BMC are not able to capture all consumption details and will therefore give a less accurate power profile.

In Figure 3.2, we notice that the power reported by the external power meter does not go down instantly at the end of the computation phase, notably for the EP and LU benchmarks. The power takes some time to return to idle values. This additional consumption could be due to the fans running at high speed in order to cool the components after the execution of the benchmark. Since the power profile given by software-based power meters excludes fan consumption, this phenomenon is not observed by Alumet.

3.4.3 Correlation and offset with external power meter

We previously observed that the software-based power profiles are similar and are visually strongly correlated with the external power meter and the BMC profiles. We will study this correlation in more depth in this section.

The tools do not have exactly the same timestamps and sampling frequencies. To do a point-by-point correlation study between power profiles, we had to fit the higher frequencies data points to the lower ones by averaging. To compute the correlation, we used the library Pandas [Reback2022] and the default Pearson correlation.

The Pearson correlation coefficient between Alumet and the external power meter is around 0.99 for all benchmarks, which is a highly strong correlation. Furthermore, the more stable the execution, the higher the correlation.

It can be noticed in Figure 3.2 that the offset between the software-based power meter and the external power meter is not constant. To verify this assumption, we computed the regression using the linear regression method of scikit-learn [Pedregosa2011]. We have found that the regression slope was 1.17, with the external power meter values being the response variable. The resulting linear regression can be seen in Figure 3.3. The higher the power, the bigger the offset. We suppose that this offset increase is related to the components whose consumption is not included by the tools, such as the power supply unit and the fans.

We believe it can be generalized that the relation between the power reported by the external power meter and the software-based power meters is not constant. Thus, estimating the total power consumption from the power reported by the tools can't be done by simply adding a constant offset. Furthermore, this relation is specific to each computing infrastructure and must be evaluated for each compute node architecture or even for each individual compute node. To do so is necessary if the goal is to report the exact energy consumed by a workload.

3.4.4 Quantitative comparison

After studying the power profiles, we looked into the total energy consumed during the execution of the benchmarks. For the external power meter, the BMC and the software-based power meters only supporting the power profile as an output, the total energies are calculated by integrating the power time series.

Figure 3.4 shows the total energy in joule spent per benchmark as provided by the tools, the external power meter, and the BMC. The error lines on top of the bars indicate the standard deviation of the energy. The external power meters report higher energy than the tools and the higher variability of the values provided by BMC leads to a greater standard deviation compared

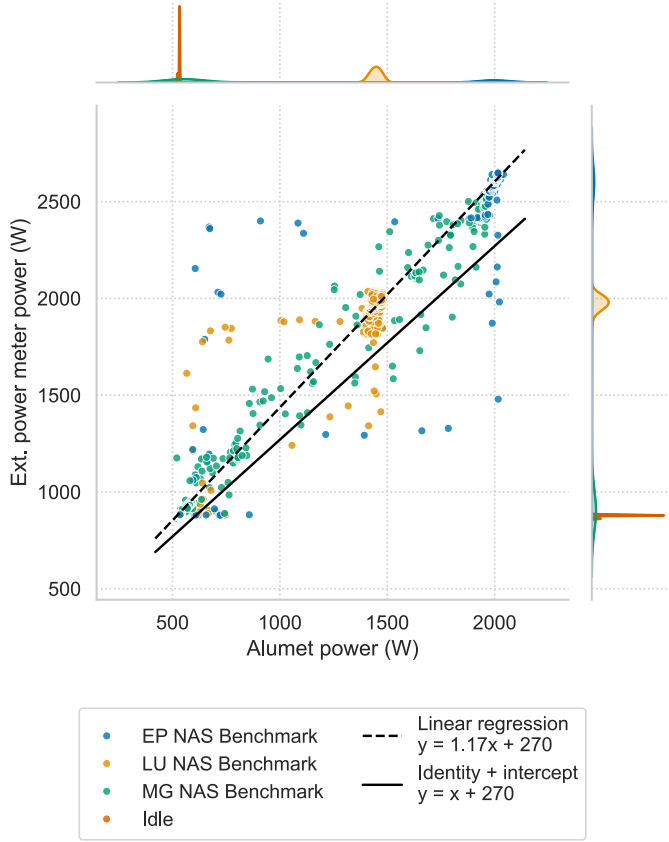


Figure 3.3: Power-power plot between the external power meter and power profiling tools.

to the external power meter. The standard deviation of the energy reported by the tools is not significant in general which means that the tools are consistent.

The differences between tool reports are large, which is due to a higher difference in sampling frequency (see Table 3.2). Experiment Impact Tracker has a higher variability on all benchmarks but especially on the MG benchmark. For all benchmarks, the online calculators (Green Algorithm, ML CO2 Impact) tend to report energies closer to the external power meters than the tools based on RAPL and NVML, but not on the MG NAS benchmark. This suggests that those calculators work better for constant workloads too, and when the average usage is known.

3.5 Conclusion

This chapter compares six software-based power meters: Alumet [Raffin2023], Carbon Tracker [Anthony2020], Code Carbon [Schmidt2021], Experiment Impact Tracker [Henderson2020], Green Algorithms [Lannelongue2021], and ML CO2 Impact [Victor Schmidt]. We detail the existing methods to measure or estimate the energy consumption of a computing node or an application execution. Then we present and compare the selected tools qualitatively. An experimental study investigates the quality of the power profiles and the energy estimations under computation and/or memory-intensive benchmarks. Those experiments allow us to evaluate other characteristics of the software-based power meters, such as the quality of their reports. We experimentally validate the consistency of the power or energy consumption reported by the software-based power meters by comparing them with high-performance external power meters. We find that the profiles are

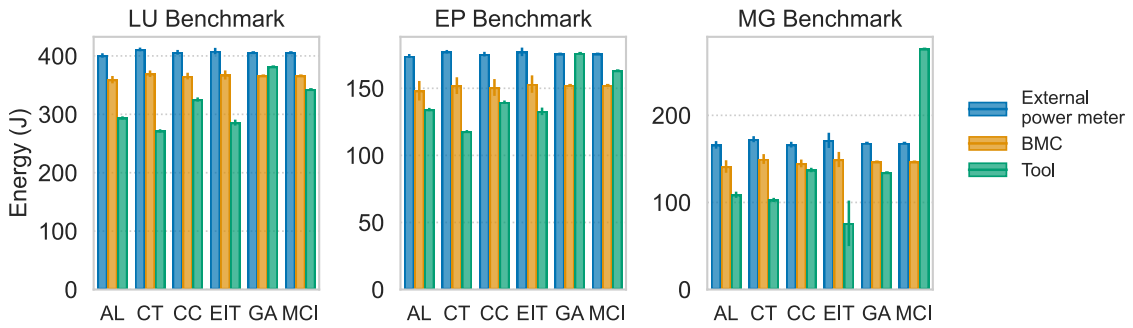


Figure 3.4: Total energy consumed by the benchmarks as reported by the power meters. Tools: Alumet (AL), Carbon Tracker (CT), Code Carbon (CC), Experiment Impact Tracker (EIT), Green Algorithm (GA), ML CO2 Impact (MCI)

highly correlated. We show and explain why the offset between the external power meter and the software-based power meters is significant and not constant. We have found that the software-based power meters based on Intel RAPL and Nvidia NVML can be used to estimate consumption at a fine granularity. All studied tools give relatively similar consumption values. The main differences between them are the supported sampling frequencies, the user-friendliness, the environment in which they can be used, and the ability to estimate the power at various granularities.

Goals in measuring the power or the energy can be numerous. We identify a few scenarios and give recommendations on which tool to select based on our study.

Whether the workload to study is a long-running job or short execution is of importance. Monitoring short executions demands a high sampling frequency. But when looking for a tool easy to use and when the precision is not critical, we would recommend Green Algorithms as they don't require any additional development or tool installation.

When the objective is to reduce the amount of energy needed to execute a workload, most of the tools can be used since they all report the energy although, for some of them, you might need to compute the energy from the power time series. We have found that the energy calculators Green Algorithm and ML CO2 Impact can give very good estimations on the conditions of having a constant workload and an in-depth knowledge of the execution (duration and average usage), with the disadvantage that there is no guarantee on the quality of the estimation.

The three energy measurement Python libraries are highly similar in their energy reports. Code Carbon has the lowest offset in all experiments. Carbon Tracker computes the energy on the fly to be able to make predictions on the whole program. It is also well documented in our opinion. Experiment Impact Tracker has higher variability in the estimations.

Alumet is the sole tool for monitoring the power consumption in real-time and reporting data at the component level.

No software-based power meter allows one to exactly measure the complete energy or the power consumed by the computing node while executing a workload, as we have shown that the relationship between the external power meter and the tool power is not constant.

In this Chapter, we showed that using a tool able to monitor the power consumption at the component level can bring valuable insights, as opposed to energy-measuring or node-level tools. The limit of such a tool is the significant gap between component and node-level consumption that can't be estimated without an external power meter.

In Chapter 4, we use this tool to better understand the power consumption of each component of the Apollo and the Jeston nodes on the functional units presented in Chapter 2, as well as the total electricity spent to achieve the FU. Their analysis provides us with insights and leverages.

4

Understanding the electricity consumption of training

Contents

In this Chapter, we present the electricity consumption of the Functional Units (FUs) described in section 2.4 on an Apollo 6500 Gen10+ node and an Nvidia Jetson AGW Xavier (Specifications in section 2.2. Sections 4.1 and 4.2 focus on each computing infrastructure. Section 4.3 puts both sections in perspective by comparing them.

4.1 On Apollo, a node from HPE AI supercomputer Champollion

All 6 FUs were executed on an Apollo node. This section starts by presenting the settings specific to Apollo before presenting the analysis of the results.

4.1.1 Settings

Hyperparameters were selected from a random grid search. The only parameter differencing the execution is the random seed, as expected for any MLPerf submission. Models are trained until they reach the quality target, as defined in Table 2.4.

Monitoring energy and power

Apollo nodes are equipped with the HPE iLO 5 management processor which monitors the power consumption thanks to iPDU attached to the power supplies, according to the user guide. The specifications states that it has a 99% precision and measures power consumption below 100 mW [1]. Measurements are displayed every 10 seconds and are five-minute averages. To analyze more finely the evolution of power, we additionally collect power and energy information from the Nvidia Management Library (NVML) which monitors the power consumption of each GPU and from the Running Average Power Limit (RAPL) register of each CPU at a frequency of 2 Hz.

Reproducibility

The frequencies and power caps of the CPUs and GPUs are set to the maximum. Hyper-threading is disabled. For single-node experiments, we executed the benchmark on one fixed node to avoid variability in the energy consumption of nodes. Experiments are repeated at least 7 times to ensure generalization.

4.1.2 Results

The Apollo node was able to achieve the 6 FUs.

Global statistics

Table 4.1 presents the average and standard deviation of the energy consumption, training time, GPU utilization, and CPU utilization of each FU on one Apollo node, as reported by RAPL and NVML.

The performance (training time) is comparable with what HPE submitted for the v2.1 round. The GPU utilization percentage is high on average (88%) while the CPU utilization is low (22%) which shows that the infrastructure is well-dimensioned for the majority of the selected models and that the implementation allows it to benefit plainly from the infrastructure.

[1] https://www.hp.com/hpinfo/newsroom/press_kits/2010/techforum2010/pdf/TF_IPD_DataSheet.pdf

	Energy (kWh)	Time (min)	Utilization (%)	
			GPU	CPU
FU				
ResNet-50	1.61 ± 0	29.42 ± 0.19	94.84 ± 0.71	28.57 ± 0.72
3D U-Net	1.73 ± 0.7	31.72 ± 12.6	96.94 ± 0.89	8.2 ± 0.07
Mask R-CNN	2.16 ± 0.09	43.6 ± 1.69	89.87 ± 0.2	8.84 ± 0.02
RNN-T	1.97 ± 0.11	36.12 ± 2.21	95.46 ± 0.28	68.47 ± 0.89
BERT-large	1.13 ± 0.01	20.83 ± 0.25	96.88 ± 0.11	6.88 ± 0.01
DLRM	0.14 ± 0	4.18 ± 0.02	57 ± 0.46	5.36 ± 0.03

Table 4.1: FU performance statistics on Apollo (average pm standard deviation).

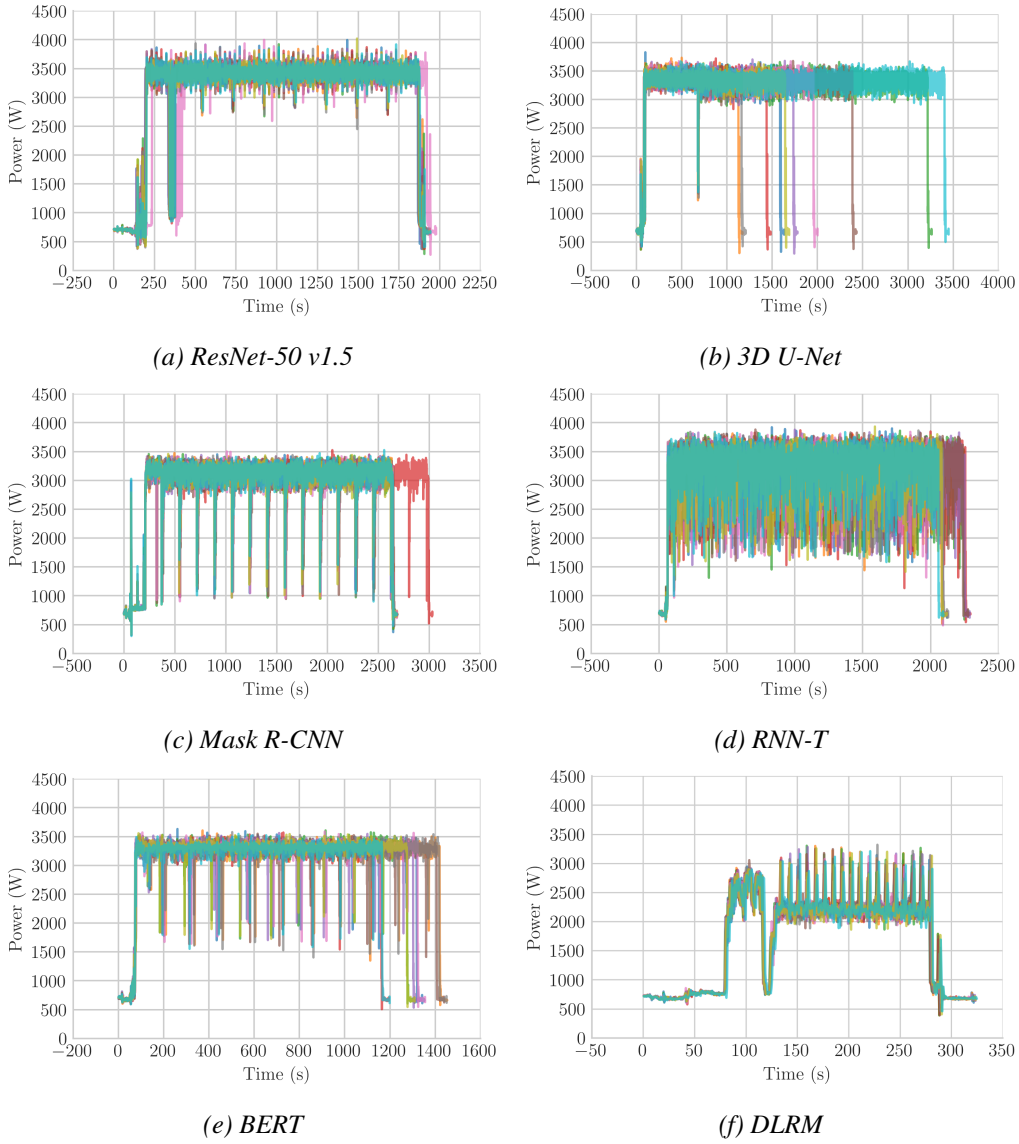
DLRM trains the fastest (less than 5 minutes) while the other model trainings require between 20 and 43 minutes. It also has a lower GPU utilization than average, as opposed to 95% for the other models. 3D U-Net stands out with a highly variable training time, which means that the quality of training is highly influenced by training randomness. Finally, RNN-T has a high CPU utilization, which suggests a larger amount of data processing than other models.

Power profiles

The evolution of power in Figure 4.1 offers more insights. The execution includes several phases: initialization, training steps, and evaluation phases. The initialization is globally not significant, except for DLMR for which it lasts almost at half the execution. The evaluation phases are shared out among the training and correspond to the low power peaks. Peaks are more or less low or frequent depending on the models, likely depending on the size of the data to load. RNN-T is characterized by highly frequent evaluation peaks. We can additionally notice that the power drawn during the most intensive training phase is different from one model to another. It ranges from 2100 W to 3500 W.

Correlation between model parameters and electricity consumption

To better understand those differences, we explore the parameters that can impact electricity consumption. The number and precision of the parameters as well as the size of the samples and the batches seem like the most significant factors that might influence the training. Table 4.2 shows the size of model parameters, datasets, and data batches. We include the size of the dataset and batches in gigabytes (GB) since sample types are highly diverse - from short texts to large images. The differences between models are quite significant. The dataset size ranges from 20 to 500 GB, the batch size from 5 to 2763, and the number of parameters from 19 to 540 million. It is interesting to notice that the batch size in GB that was optimal to train RNN-T is significantly higher than for other models which leads to believe that time is wasted in loading data. It explains why RNN-T training consumes more energy than BERT-large and DLRM when it has fewer parameters to train. Surprisingly, DLRM is the fastest model to train when it has the largest number of parameters and the heaviest database. It is the model with the lowest instant power consumption. This suggests that the model fills most of the memory thus forcing the batch size to be small. In conclusion, the memory capacity of the node prevents the GPU from being intensely used during training.



(a) ResNet-50 v1.5

(b) 3D U-Net

(c) Mask R-CNN

(d) RNN-T

(e) BERT

(f) DLRM

Figure 4.1: Power evolution of training models.

Model	Parameter number (M)	Dataset size		Batch size	
		Sample	GB	Sample	In GB
ResNet-50 v1.5	25.6	1.28e+6	167	408	53
3D U-Net	19	6.72e+4	40	56	229
Mask R-CNN	25.6	4.00e+4	20	96	48
RNN-T	29.8	2.78e+5	500	1536	2763
BERT-large	345	3.00e+6	400	384	51
DLRM	540	3.78e+9	342	55296	5

Table 4.2: Model implementation details. The batch size is the ratio between the dataset size and the number of samples in a batch.

From Table 4.2 and Table 4.1, Figure 4.2 shows the correlation[2] between the implementation details (Number of parameters, Batch size, and Dataset size) and the performance (Electricity consumption, Execution time, CPU and GPU utilization, and Average power consumption). Over the 6 FU, the energy consumption and training time are highly correlated (higher than 99%) as can be expected since the workloads have a similar power profile. The GPU utilization is mostly correlated with the average power consumption, which is not surprising since the GPUs represent most of the power consumption and the power of a component is directly impacted by its utilization. The CPU utilization is highly correlated with the batch size in gigabytes which might be linked to pre-processing data tasks performed by the CPU. Surprisingly, the energy consumption is inversely correlated with the number of parameters and the number of samples per batch. We remove DLRM from the analysis since it can be considered an outlier, but it didn't change those conclusions.

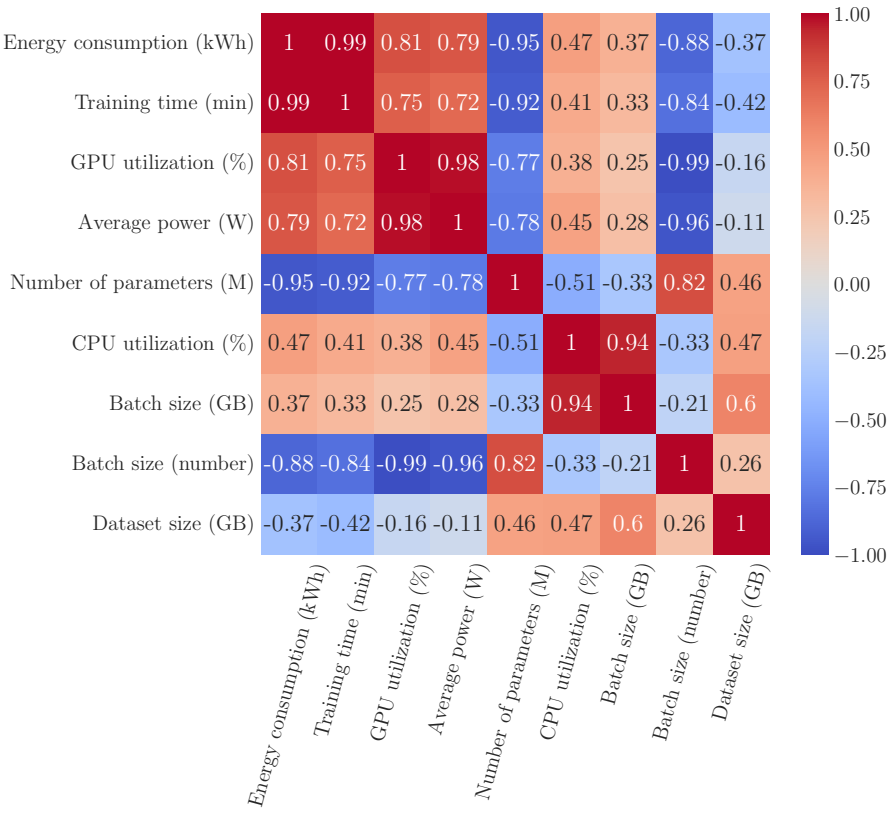


Figure 4.2: Correlation between model characteristics and training performances.

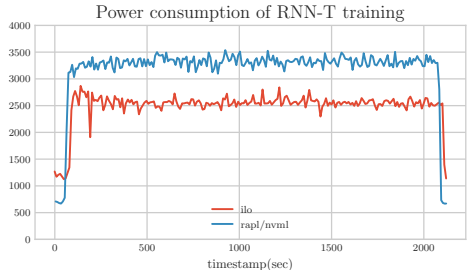
The impact of data transfer can also be important. As the size of the dataset increases, the storage needs to be distributed and the choice of storage system affects the data loading latency. When training ML models, the computations are made on a batch of data in parallel and if this batch doesn't fit in memory or even locally, it needs to be loaded regularly. The latency becomes significant if this loading frequency is high and the loading time is significant. Unfortunately, we didn't monitor the amount of data transferred from the various ports. The datasets are by default stored in the Clusterstor. We noticed it had a real impact on ResNet training throughput. By moving the dataset to the local storage, the training time was divided by 7 and the energy consumption by 3. The energy consumed by the interface ports is not included in the energy monitoring. Monitoring the total energy consumed by the node might bring insights into the

[2] https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

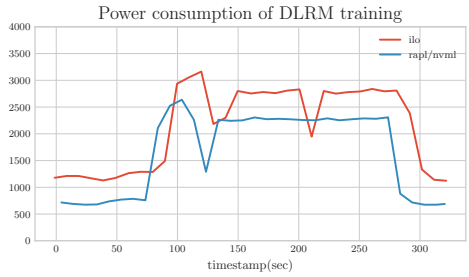
impact of data transfer on the total energy consumption.

Comparing iLO and software-based power meters

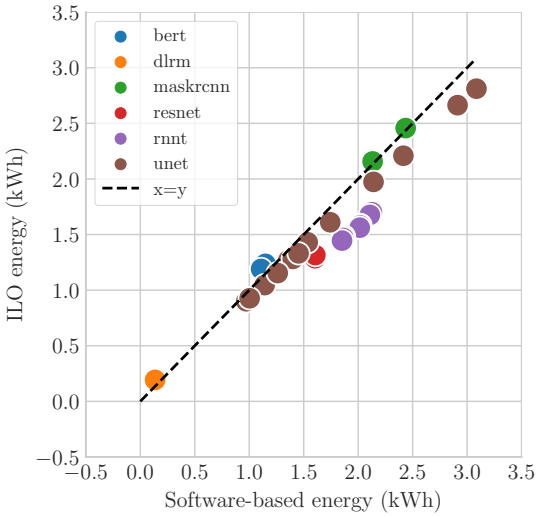
As we presented in Chapter 3, the gap between the software-based power meter and an external power meter can be significant and corresponds to the consumption of the fan, the data transfer, and network cards. In this section, we study the gap between a software-based power meter and an iPDU called iLO and installed on each Apollo node for each FU. Figure 4.3c shows the power evolution for the RNN-T FU and the DLRM FU. We can see that the gap is not consistent, although the evolution is very similar. iLO reports are the average over the datapoints of the last 5 minutes which explains the time offset between both power meters.



(a) RNN-T



(b) DLRM



(c) Evolution of the power as reported by iLO and the software-based power meter on the FU.

(d) Comparing the total energy consumed for every conducted experiments.

Figure 4.3: Comparing iLO and software-based power meter monitoring.

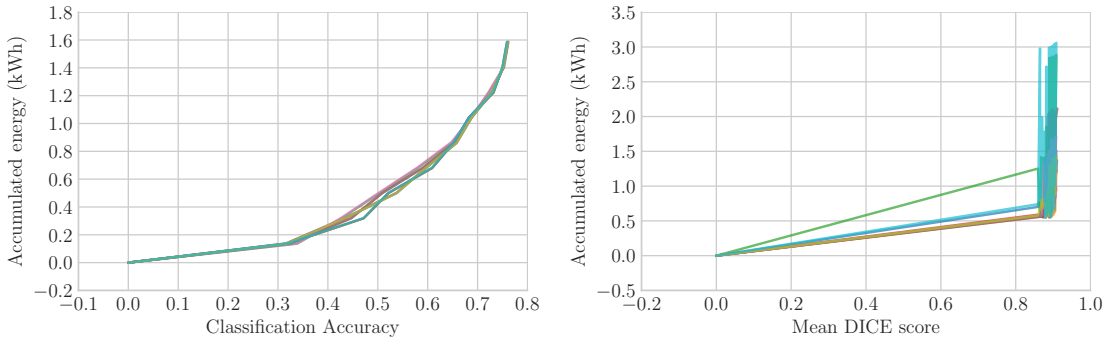
Figure 4.3d shows the relationship between the total energy of each repetition of the FUs as reported by iLO and a software-based power meter. One data point corresponds to one FU. We would have expected the data points to be at the left of the $x = y$ line since iLO should have reported a higher energy consumption. This is not the case, although it is different depending on the model. On average, the energy reported by the software-based power meter is higher than the one reported by iLO, but that is not the case for BERT.

Those findings are not coherent with previous results. In Chapter 3, we found that reports from RAPL and NVML were reasonable based on a power meter placed between the plug and the node. An iPDU should show the same behavior and we were not able to explain the incoherence.

Impact of quality target on training energy consumption

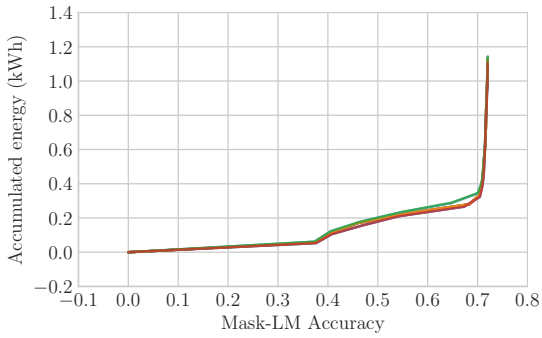
The quality of the model on the validation dataset is a measure of the performance of the model. Training an ML model systematically follows the same pattern. The quality increase is slowing batch after batch. In terms of time and energy, it means that each quality percentage consumes more to be reached.

The quality metric depends on the model and the application. It can be a metric to minimize or to maximize. Figure 4.4 shows that the accumulated energy has an exponential relationship with the quality metric., for models whose quality metric is maximized. Figure 4.4e presents this relationship for models whose quality metric is maximized, and Figure 4.4h for models whose quality metric is minimized. Except for ResNet and Mask R-CNN, most of the energy is consumed by the very last percent of the quality metric. This suggests that the quality target should be carefully defined and could be a significant energy reduction leverage.

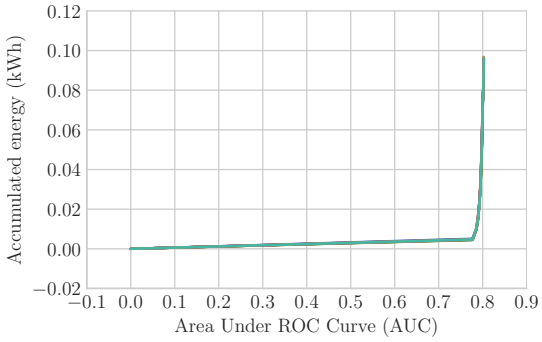


(a) ResNet

(b) U-Net

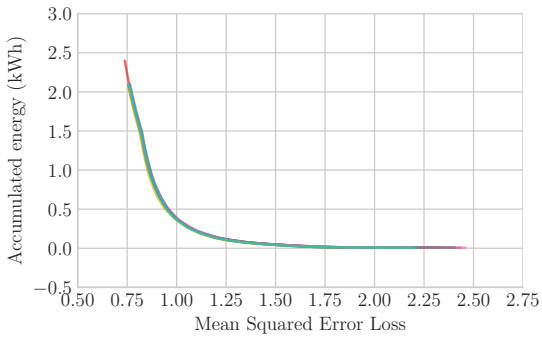


(c) BERT

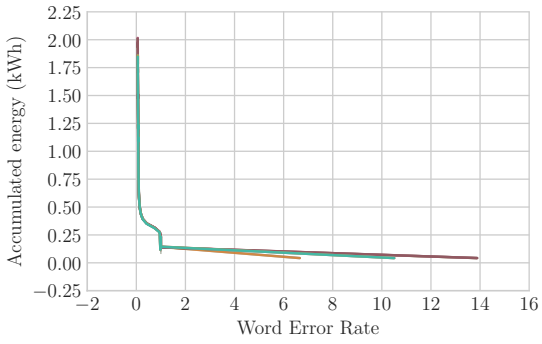


(d) DLRM

(e) Models whose quality metric is maximized.



(f) Mask R-CNN



(g) RNN-T

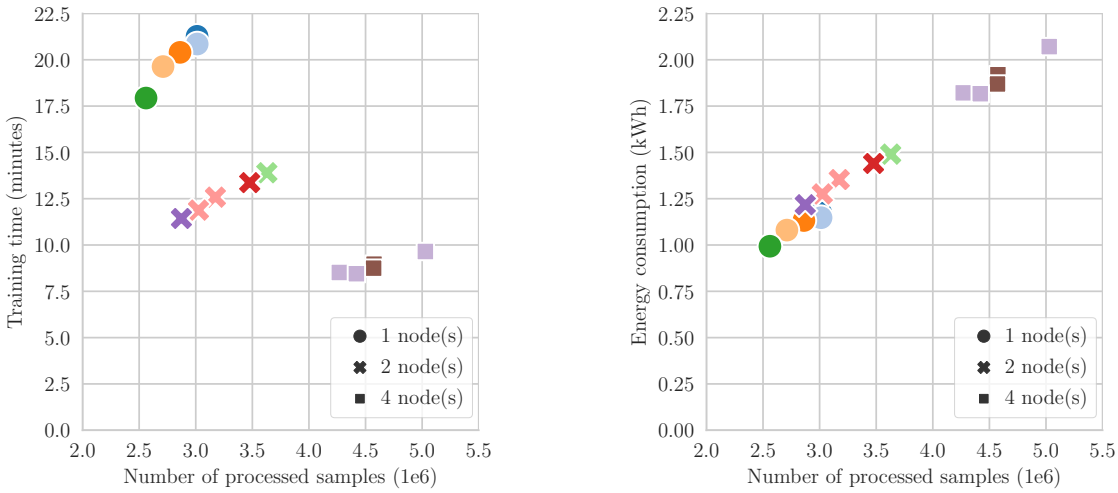
(h) Models whose quality metric is minimized.

Figure 4.4: Energy required to reach each quality metric point for the Apollo node for each FU.

Impact of sizing up the infrastructure

We conducted experiments to study the impact of increasing the number of nodes used for the training. Figure 4.5 shows the relationship between the energy consumption and the training time

and the number of processed samples for BERT FU on 1, 2, and 4 nodes. Nodes are selected randomly among the 20 nodes of Champollion at each experiment.



(a) *(b)*

Figure 4.5: Training time and energy relationship with the total number of processed samples for the BERT FU. The color indicates the set of nodes that was used.

For a given number of nodes, five experiments were conducted, each one characterized by a seed and a set of nodes. The color indicates the set of nodes, as some of them overlap particularly when the experiments require 4 nodes. It can be seen that the variability in time and energy is quite significant, due to the instability of multi-node training. It means that the target was attained with fewer steps in some experiments than in others, and as a consequence, fewer samples were processed. The number of processed samples was selected to compare different numbers of nodes since more nodes are able to process more samples in the same amount of time. However, we could expect the energy consumed to be similar for identical numbers of processed samples.

In Figure 4.5a, it can be noticed that when the number of nodes is multiplied by two, the time is not divided by 2 but by a factor of 1.5. This is due to the regularisation effect of processing more samples in parallel which slows down the training. As a consequence, the algorithm has to process more samples thus requiring more time.

Similarly, Figure 4.5b shows that training with twice as many nodes consumes more energy (by a factor of 1.31). This means that there is a tradeoff to find between minimizing the energy consumption and minimizing the training time.

It seems that the energy depends more on the number of samples that were needed to reach the quality target than on the number of nodes. For the few data points for which the number of nodes is different and the number of processed samples is equivalent, it seems that more nodes consume more energy. This is likely due to the time overhead of transferring and aggregating data across nodes.

Distributing data and learning across nodes is a common practice to reduce the training time. However, it does not seem like a good leverage to reduce energy consumption. It also has a significant impact on the learning thus a hyperparameter optimization is needed to find the best trade-off.

4.1.3 Discussion

Studying the power and electricity consumption of computing nodes in ML training comes with many challenges. Existing measurement tools are unreliable and prevent insights from components other than the CPUs and the GPUs that are monitored by software-based power meters.

Unfortunately, components like networking interfaces and switches play a significant role in training and their impact on the total energy cost would be interesting to study.

Not many leverages can be used to reduce the electricity consumption of the FUs on one Apollo node. Our study does not find a strong correlation between implementation parameters and energy consumption. The computing nodes need to be dimensioned for the model that needs to be trained. Each model has its memory and computation characteristics as well as its software and optimization libraries and those libraries need to be adjusted to the computing node. Integrating software information in the analysis might be interesting. Optimizing the classic performance criteria like the training time leads to a reduction in energy consumption, for a given node and model. For our set of experiment, the choice of quality target is the most impactful leverage. Increasing the number of nodes can reduce the training time, but not the energy consumption, and this conclusion was established without taking switches and networking interfaces into account, which might significantly increase the overall consumption.

4.2 On Jetson, an embedded AI device

As previously explained, only the ResNet-50 FU was executed on a Jetson node. The majority of the other models exceeded the Jetson node's computational capacity for training. Additionaly, the training time of the models that fit was too significant to allow the study of several models.

4.2.1 Settings

We conducted a grid search to find the best hyperparameters for the model and the hardware. The parameters are listed in Table 4.3. The learning rate is divided by 10 every 40,000 steps and the training is stopped if the accuracy does not improve for 20,000 steps. The number of workers is set to 2 to reduce the CPU load.

Parameter	Value
Image resolution	224x224
Batch size	192
Learning rate	0.001
Loss	Negative log likelihood
Optimizer	Adam
Early stop (step)	20,000
Accuracy target	75.9%
Accuracy metric	Averaged

Table 4.3: Hyperparameters for the ResNet-50 FU on Jetson AGX Xavier

Monitoring energy and power

Nvidia processors are equipped with power meters that can control and monitor the instant power consumed by the GPU, the CPU, and the memory. The power mode is set to 30W. At the beginning

of each experiment, the Jetson-stats application[3] is launched to monitor the CPU and GPU power and usage of each host. The acquisition frequency is 1Hz. Energy is computed as the sum of the product of instant power and the average time interval between two acquisitions.

Reproducibility

The frequency of the CPU and the GPU are fixed to their maximum value. The cache is emptied every 500 steps to avoid OOM errors. Considering the duration of the experiments, it was more difficult to perform a hyper-parameter search and the FU were only repeated twice with different random seeds.

4.2.2 Results

The training was stopped because it had not improved. The maximum reached accuracy on the train dataset was 74% and 55% on the evaluation dataset. Thus we define ResNet-50* FU as in table 4.4.

ResNet-50*	Train ResNet-50 on ImageNet until it achieves a 55% classification score.

Table 4.4: Functional unit

Global statistics

Table 4.5 shows the performance of Jetson on the ResNet-50* FU. We can notice that the training time is significant, reaching almost 88 hours. This explains the large energy consumption of 3.15 kWh. The high GPU utilization proves that we were able to push the component to its limits. Similarly, the RAM averages at 69%.

	Energy (kWh)	**Time (min)**	**Utilization (%)**	
			GPU	**CPU**
FU				
ResNet-50*	3.15	5332	97.93	15.56

Table 4.5: FU performance statistics on Jetson.

Power profiles

The evolution of the instant power consumption can be seen in Figure 4.6. The displayed power is an average of the last 30 seconds to improve the figure readability.

The GPU consumes on average 19.60 W during training while the CPU only consumes 2.66 W. The average total power consumption is 37.04 W, which is surprising since the TDP of the Jetson AGX Xavier is 30 W. The significant gap between the sum of the CPU and GPU and the total consumption comes from the consumption of RAM, storage, and network components.

With a closer look at Figure 4.6, we can notice regular power peaks. It corresponds to evaluations on the test set which leads to intense In/Out (I0) operations when the machine is preprocessing samples, loading them to the GPU, and performing inference. Those steps require less GPU

[3] https://rnext.it/jetson_stats/reference/jtop.html#jtop.jtop.power

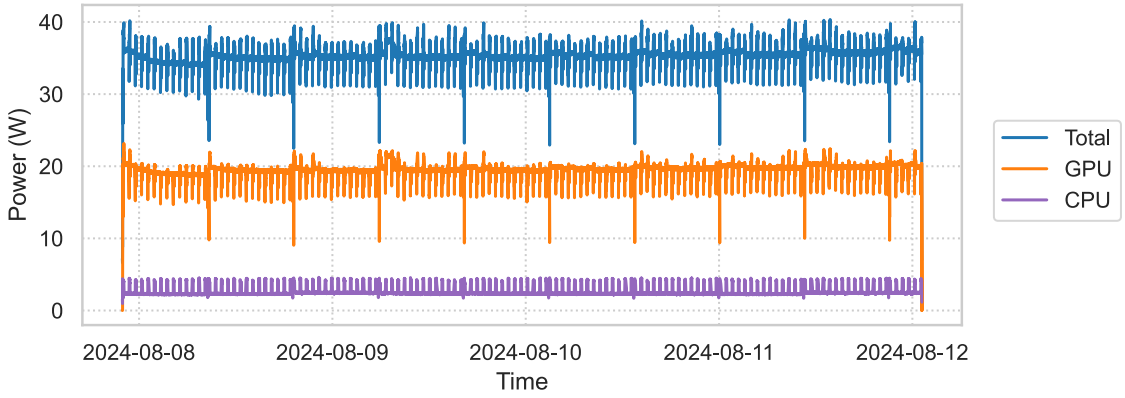


Figure 4.6: The power consumption of ResNet-50 FU on Jetson AGX Xavier.*

and more CPU usage which explain both upward and downward peaks. Larger peaks are due to learning rate change and checkpointing.

Impact of accuracy on training energy consumption

Figure 4.7 shows the energy required to reach each point gain in accuracy. As before, each accuracy point is harder to reach than the previous, except for a peak around 50% accuracy.

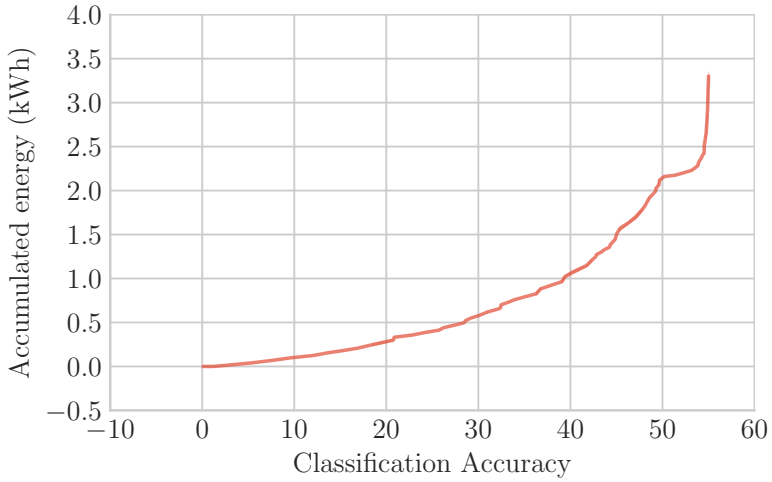


Figure 4.7: Energy required to reach each accuracy point for the Jetson node on the ResNet-50 FU.*

4.2.3 Discussion

It seems that a batch size of 192 was the limiting factor to reach the target accuracy. Other than optimizing the performance of the training, it is also the highest batch size before getting Out Of Memory (OOM) errors. This suggests that the hardware is under-dimensioned for the model. Thus a memory constraint prevents the training from reaching the accuracy we know can be attained with the given model and dataset.

4.3 Comparing the electricity consumption of deep learning training across ML infrastructure

On Apollo, ResNet-50 took 1.61 kWh to reach a 75.9% classification score. On Jetson, the model stopped learning after reaching 55% on the test dataset, and it required 3.15 kWh. In summary, it took twice as much energy to reach 60% of the target.

Figure 4.8 shows the energy required to reach each accuracy point while training ResNet-50 on ImageNet for both the Apollo and the Jetson nodes. This graph enables us to compare the energy efficiency of the training on both infrastructures.

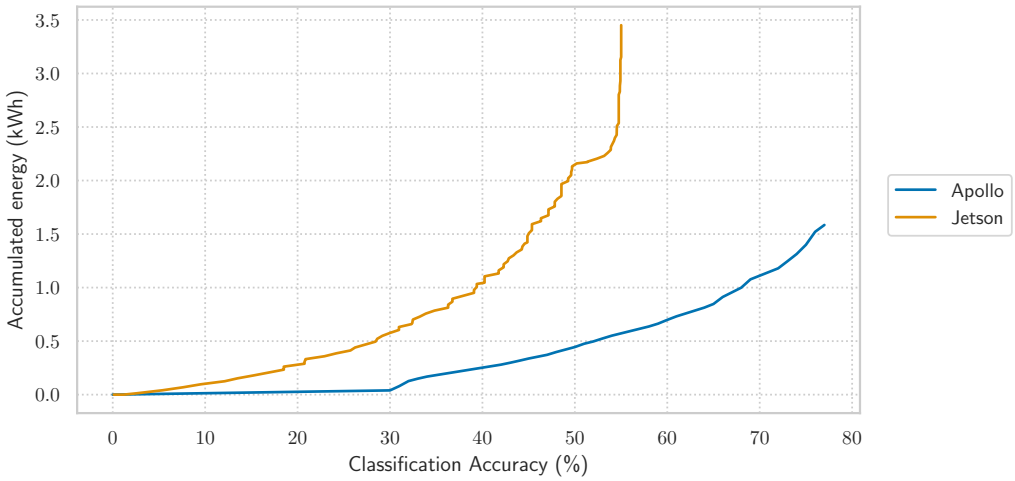


Figure 4.8: Energy required to reach each accuracy point for the Jetson node and the Apollo node while training ResNet-50 on ImageNet.

We evaluate Apollo on the ResNet-50* FU to enable a fair comparison. Apollo required 0.57 kWh and 11.26 minutes to achieve this FU. Reducing the target accuracy by 20.9% divided the metrics by more than 2.

Figure 4.9 compares on several criteria the ResNet-50* FU on the Jetson node and the Apollo node and the ResNet-50 FU on the Apollo node. Original criteria (electricity consumption, training time, and accuracy) were normalized between 0 and 1 such that the objective for each criterion is to be closer to 1. Thus, the larger the surface, the better the infrastructure. The electricity consumption and training time were converted to electric efficiency and speed which are criteria we want to optimize for each FU. The normalization values were selected to highlight the comparison between computing infrastructures and FUs. As a consequence, the value should not be taken into account as absolute values but relatively between traces. Minimum normalization values are systematically zeros.

It can be seen that Apollo dominates Jetson on the three criteria. The speed is the most differentiating criterion. Compared to the amount of time required to train on a Jetson node, the difference between both FU on an Apollo node is not significant. However, the electric efficiency was improved by reducing the target accuracy.

In Chapter 5, we convert electricity consumption into environmental impacts and estimate the embodied cost of the hardware. Considering the difference in the size of both nodes, it might be possible that the more significant embodied cost of an Apollo node compensates for the additional electricity cost of the ResNet-50 FU on the Jetson node.

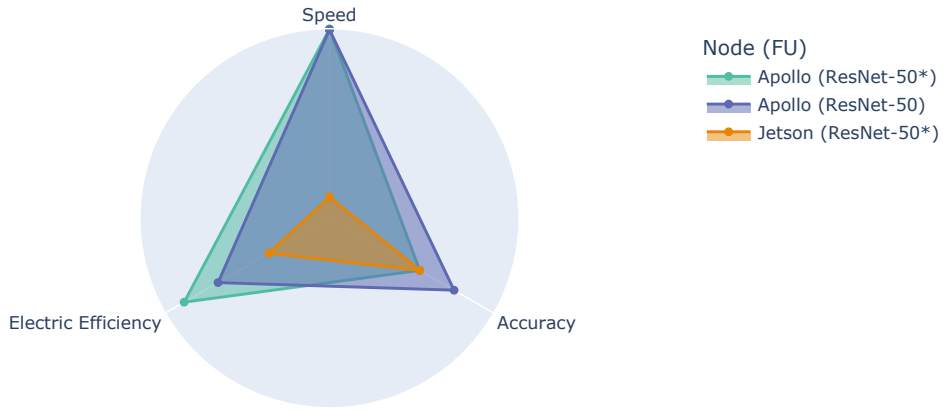


Figure 4.9: Multi-criteria comparison of the ResNet-50 and ResNet-50 FUs on Jetson and Apollo. Values were normalized between 0 and 1, with 1 being the target of the corresponding criteria. For criteria with values outside of the [0,1] interval, values were normalized between 0 and 1, with 1 being the target of the corresponding criteria. Minimum and maximum values are set to (5, 0), (100, 0) for electricity consumption (kWh) and duration (hours), respectively, thus converting them into efficiency metrics.*

5

The environmental impacts of ML infrastructures and ML training

Contents

ICT has numerous environmental impacts including but not exclusively primary energy consumption, rare metal resources depletion (or abiotic depletion), or carbon emissions. Those impacts come from various phases in the life cycle of products.

In this chapter, we describe how we estimate the embodied impacts of the Apollo and Jetson nodes (Section 5.1) and allocate them to the Functional Units defined in Chapter 2. We analyze the effects of each component of both nodes on the total impact (Section 5.2), study the relative usage and embodied impacts (Section 5.3), and compare the impacts of training ResNet-50 on Apollo and Jetson nodes on the three indicators (Section 5.4). Section 5.5 provides a sensitivity analysis of the environmental impacts on the hypothesis.

5.1 LCA, databases, and hypothesis

Environmental data on manufacturing computing nodes is scarce, especially for recent compute nodes such as Apollo. We rely on the API of Datavizta which collects data from the Green Cloud Computing database [Gröger2021] for most of the compute node embodied impacts. However, the impacts of Nvidia GPUs are not open-sourced and have to be estimated. For the electricity mix impact factor and PUE parameters, we decided to use the geographical region averages, as we aim to produce conclusions uninfluenced by the temporal dynamics of the electricity mix and data center efficiency differences.

Table 5.1 summarizes the notations that will be used to describe the methodology.

Notations	
I	: Environmental Impact (expressed in kg CO2 eq, kg Seb, and MJ)
I_{capex}	: Embodied Impact (manufacture, transport, and end of life)
I_{opex}	: Operational Impact (usage)
IF_{elec}	: Electricity mix Impact Factor
E	: Electricity consumption
AUR	: Active Utilization Rate
PUE	: Power Usage Effectiveness of the data center
T	: Use time of the equipment
s	: Surface or area (of the GPU die or of the board)
c	: Memory capacity
d	: Memory density
w	: Node weight

Table 5.1: Definition of variables used for the environmental equations

5.1.1 Estimating the embodied impacts of GPUs

Without open-source LCA of GPUs, we follow a methodology proposed by Boavizta[1]. While it was not yet peer-reviewed, the proposition is well-documented and based on environmental evaluation of chips and circuits [Liu2014,Ozkan2018,Pirson2021,Nassajfar2021]. As an example, [Luccioni2023b] used an arbitrary value of 150 kgCO2eq for an Nvidia GPU. Instead, Boavizta relies on a study by the German Federal Environment Agency (Green Cloud Computing), which developed a model of the impacts of CPUs using as inputs their die area size (s_{die}), memory density (d) and capacity (c), the printed circuit board (PCB) area (s_{PCB}). We can use their model if we assume that GPUs are based on similar semiconductors. Equation 5.1 lists the equations we used to estimate the impacts of each GPU from the specifications previously mentioned. Constant values can be found in Boavizta Documentation[2].

$$
\begin{aligned}
&I_{compute}(s_{die}) = s_{die} * I_{compute,manufactoring} + I_{compute,transport} \\
&I_{memory}(c,d) = \frac{c}{d} * I_{memory,manufactoring} + I_{memory,transport} \\
&I_{board}(s_{PCB}) = s_{PCB} * I_{board} \\
&I_{GPU,capex} = \\
&\quad I_{compute}(s_{die}) + I_{memory}(c,d) + I_{board}(s_{PCB}) + I_{HeatSink} + I_{PCIEConnector}
\end{aligned}
\tag{5.1}
$$

Another advantage of this method is that it enables a comparison between the impact of each component of the GPU. Its flexibility enables it to be applied to different chip designs, and notably various memory capacities, while being based on easily accessible design parameters.

5.1.2 Estimating the embodied impacts of the node

We use the Boavizta API [3] to estimate the impacts of CPUs, RAM, storage, and other components. The storage is not included due to the lack of knowledge of the Apollo infrastructure.

CPU For both computing nodes, we selected the number of CPU, cores, and amount of RAM according to their specification. However, the models of CPU were not available in the database. We chose the Milan architecture for Apollo and the ICE lake architecture for Jetson to be the closest to their respective architectures.

Other components The impacts of the power supply and the node case are averaged across models. We consider one power supply for one node. Considering the difference in the size of the node cases we are studying and the significance of the case impacts in the total impacts of nodes, we didn't want to use the same values for the case impacts of Apollo and Jetson. Thus we assume that the impacts of the node case (I_{case}) are proportional to the node weight (w), as in Equation 5.2.

$$
I_{case} = \frac{w}{\bar{w}} * \bar{I}_{case} \tag{5.2}
$$

[1] https://github.com/Boavizta/boaviztapi/issues/65

[2] https://boavizta.org/en/blog/empreinte-de-la-fabrication-d-un-serveur,https://github.com/Boavizta/boaviztapi/issues/65

[3] https://dataviz.boavizta.org/serversimpact

With $\bar{w}$ and $\bar{I}_{case}$ the weight and the case impacts of a classic server. We assume a classic server weighs 20 kg, an average value from the products Boavizta relied on to estimate the case impacts. According to the specifications, Apollo weighs 96.27 kg and Jetson 1.548 kg.

5.1.3 Allocating the embodied impacts of the node to training

As explained in chapter 2, section 2.1, we chose a time-based allocation, as in Equation 5.3.

$$I_{training,capex} = \frac{T_{training}}{AUR * T_{lifetime}} * (I_{GPU,capex} + I_{CPU,capex} + I_{RAM,capex} + I_{Other,capex}) \tag{5.3}$$

We assume a lifetime ($T_{lifetime}$) of 4 years and an average utilization rate (AUR) of 50% for the Apollo and Jetson nodes.

5.1.4 Estimating the impacts of the electricity consumption

The electricity consumption is multiplied by the electricity impact factor and the PUE of the data center to obtain the opex-related impacts of the training.

$$I_{training,opex} = PUE * IF_{elec} * E_{training} \tag{5.4}$$

We used a PUE of 1.56, the world average in 2024 according to the Uptime Institute Global Data Center Survey Results of 2024, and the French electricity mix impact factors[4].

5.1.5 The environmental impacts of training

The total impacts of training are the sum of both operational and embodied impacts.

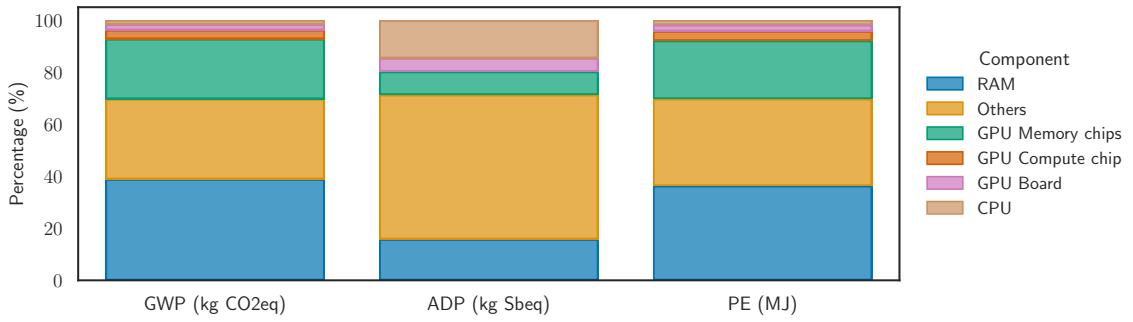
$$I_{training} = I_{training,opex} + I_{training,capex} \tag{5.5}$$

5.2 Node embodied impacts

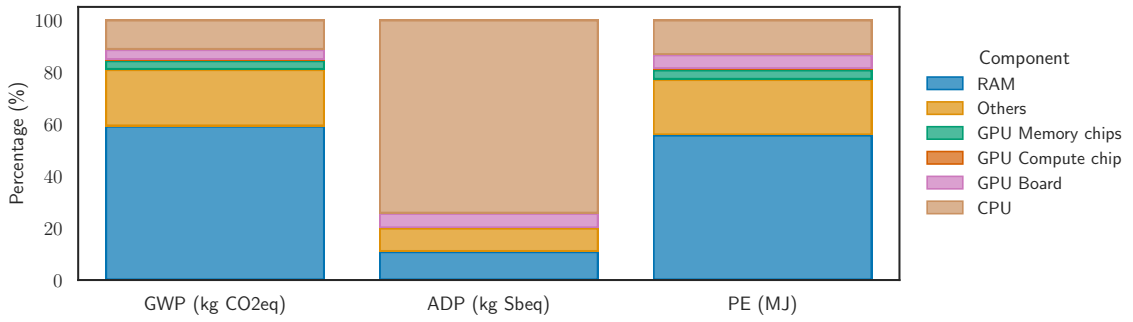
Applying the proposed methodology on an Apollo node and a Jetson node results in the embodied impacts presented in table 5.2. The embodied impacts of Apollo are around an order of magnitude higher than the impact of Jetson, as can be expected since an Apollo node contains 8 GPUs and 2 CPUs and Jetson only one of each thus the amount of metals and manufacturing is significantly higher for Apollo.

Figures 5.1a and 5.1b show the part of each compute node component in the total manufacturing impacts of an Apollo node and a Jetson node, respectively. It can be seen that for both nodes, the memory chips are the most impactful components overall, although the CPU and the GPU board have a more significant ADP impact than GWP and PE impacts. The share of CPU in the total impact is higher for Jetson, which can be explained by the higher proportion of GPU and RAM components in Apollo for a relatively similar computing capacity to the CPUs. The impact of other components (power supply, node case) is more significant for Apollo than for Jetson which is due to the allocation in weight we performed from the average node impacts.

[4]From ADEME V2.02 Base Impact and the IRENA 2022 report. PE impact factors are not available in open access thus we used the consumption of fossil resources along with the percentage of produced renewable energy (23,5%).



(a) Apollo



(b) Jetson

Figure 5.1: Share of each compute node component in the total manufacturing impacts of each node.

	Apollo	**Jetson**
GWP (kg CO2eq)	3.86E+03	8.79E+01
ADP (kg Sbeq)	2.79E-01	2.74E-02
PE (MJ)	4.97E+04	1.25E+03

Table 5.2: Total embodied or capex-related impact of Apollo and Jetson

5.3 Allocating the embodied impact to training

This section focuses on the impacts of training by analyzing the usage and allocated embodied impacts of the FUs. It also compares the ResNet-50* FU on Jetson and Apollo. A sensitivity analysis shows the effect of the hypothesis on the results. Calculations were done on a spreadsheet and published on the web [5].

5.3.1 Impacts of training on Apollo

Table 5.3 summarizes the impacts of training each model (See Table 2.5 for details), after applying Equation 5.4 on the electricity consumption and the allocation equation 5.3 on the embodied impacts of Apollo.

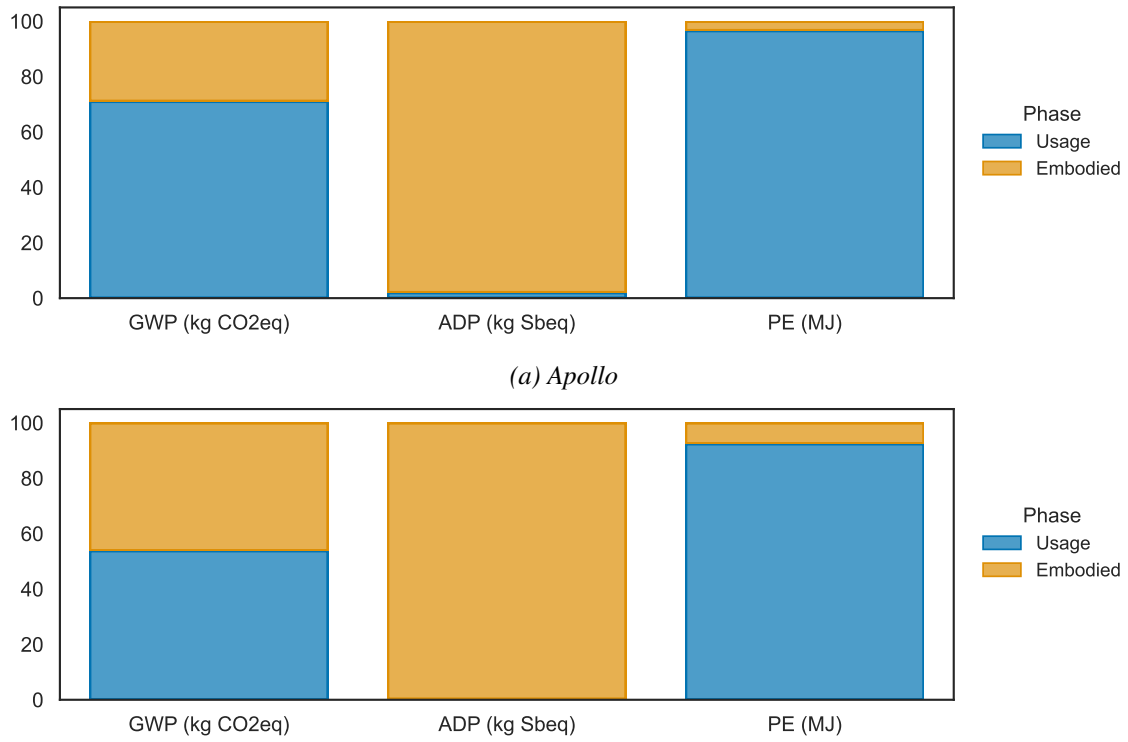
Figure 5.2a compares the usage and manufacturing parts in the impacts allocated to training ResNet. The usage dominates the PE and GWP impacts while manufacturing and transports

[5] https://docs.google.com/spreadsheets/d/e/2PACX-1vTVrRxa05Wb5JTqr3L7Uzq25G9YGoYz1EEqS6oz2_2eu1dBBUr7VtWKiw6H3G0h0MeWe5C4_3Ej5SBY/pubhtml

	ResNet-50	3D U-Net	Mask R-CNN	RNN-T	BERT	DLRM
GWP (kg CO2eq)	2.87e-1	3.08e-1	3.97e-1	3.52e-1	2.02e-1	2.90e-2
ADP (kg Sbeq)	6.12e-6	6.60e-6	9.06e-6	7.52e-6	4.34e-6	8.63e-7
PE (MJ)	3.16e+1	3.40e+1	4.26e+1	3.89e+1	2.23e+1	2.73e+0

Table 5.3: Total impacts of each FU on Apollo, including both manufacturing and usage.

dominate the ADP impact.



(a) Apollo

(b) Jetson

Figure 5.2: Share of usage and embodied phase in the total impacts of the ResNet-50 FU on the Apollo node and on the Jetson node.*

5.3.2 Impacts of training on Jetson

Similarly, Table 5.4 shows the total impacts of the ResNet-5* FU on a Jetson node, and Figure 5.2b the share of usage and manufacturing and transport in those total impacts. It can be noticed that the share of the embodied phase is more important for Jetson than for Apollo, which can be explained by the difference in training time, considering that we set the same lifetime for both infrastructures.

	ResNet-50*
GWP (kg CO2eq)	7.43E-01
ADP (kg Sbeq)	1.07E-04
PE (MJ)	6.49E+01

Table 5.4: Total impacts of the ResNet-50 FU on Jetson, including both manufacturing and usage.*

5.4 Comparison of the environmental impacts of training on Apollo and on Jetson

The larger embodied impacts of Apollo don't compensate for the inferiority of the computing capacity of Jetson. Table 5.5 shows the total impacts of executing the ResNet-50* FU on the Apollo node and the Jetson node.

	Apollo	**Jetson**
GWP (kg CO2eq)	1.04E-01	7.43E-01
ADP (kg Sbeq)	2.34E-06	1.07E-04
PE (MJ)	1.13E+01	6.49E+01

Table 5.5: Total impacts of the ResNet-50 FU, including both manufacturing and usage.*

The global warming potential (GWP) of the ResNet-50* FU on Jetson is 7 times as high as on Apollo. The abiotic depletion potential (ADP) is 46 times higher, and the Primary Energy (PE) is 5.7 times more important. Figure 5.3 adds those three impacts to the comparison started in Chapter 4. Instead of reducing the gap between both computing nodes, taking into account the impact of electricity and the allocated embodied impacts exacerbates the dominance of Apollo. The criteria most differentiating the nodes are the speed and the ADP efficiency. In conclusion, the larger embodied impact of an Apollo node doesn't counterbalance the longer training time on Jetson.

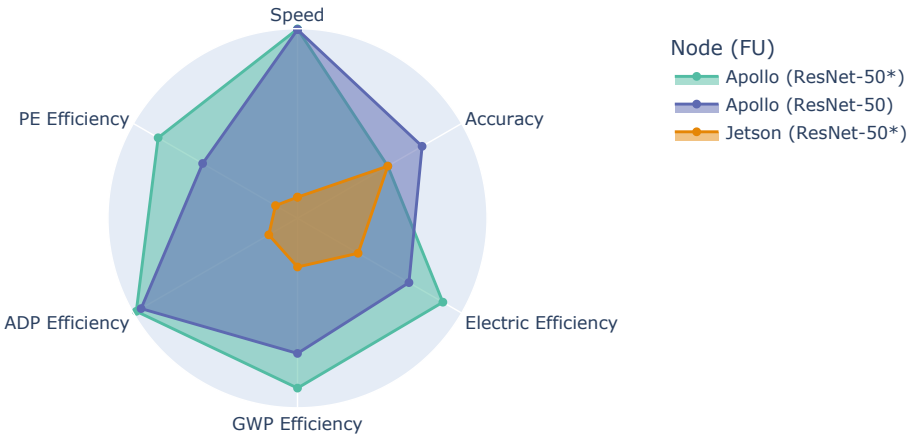


Figure 5.3: Multi-criteria comparison of the ResNet-50 and ResNet-50 FUs on Jetson and Apollo. Values were normalized between 0 and 1, with 1 being the target of the corresponding criteria. For criteria with values outside of the [0,1] interval, values were normalized between 0 and 1, with 1 being the target of the corresponding criteria. Minimum and maximum values are set to (5, 0), (100, 0), (1, 0), (0.00013, 0), (75, 0) for electricity consumption (kWh), duration (hours), GWP (kg CO2eq), ADP (kg Sbeq), and PE (MJ), respectively, thus converting them into efficiency metrics.*

5.5 Discussion and sensitivity analysis

This section highlights the importance of using an accurate software-based power meter to measure electricity consumption. The estimation of the usage phase is more reliable and the comparison has greater value.

We decided to use an average electricity impact factor to simplify the process, but this methodology could be used with a dynamic electricity impact factor to compare - for example - carbon aware computing techniques, whose benefits have been proven [Madon2022, Vasconcelos2023]. ML model training workloads are suitable for such techniques since they are less impacted by delays.

This analysis could be improved by taking into account storage and network. Doing so would require defining another allocation key since such equipment is not shared in a time-based approach.

The Average Utilization Rate, the Power Usage Effectiveness of the Data Center, and the lifetime of the node have significant impacts on the total impacts of training ML models on both Apollo and Jetson. For the previously presented results, we assume an AUR of 65%, a PUE of 1.56, and a lifetime of 4 years for both nodes. But depending on where the workload is performed, those values can be highly different. If Apollo was placed inside a Data Center with a PUE of 1.1 which is closer to the values published by cloud companies, the GWP and the PE of training ResNet would be reduced by 26% and 40%, respectively, since those impacts mainly come from the usage that the PUE directly impacts. Considering the current demand for AI hardware, it is fair to assume that ML nodes are more efficiently used and the AUR reached 100%. If that were the case, the ADP would be reduced by 52%. If the lifetime was multiplied by 1.5, there would be a 48% decrease in ADP. The lifetime of specialized clusters such as Champollion is harder to predict than classic server nodes since they require more resources to build and use. It would be reasonable to assume that they are used for longer than 4 years. In those three scenarios, the other impacts are reduced by less than 15%.

We could additionally assume that the Jetson node is placed at the extreme edge thus having a PUE of 1. This would reduce the GWP and the PE by 24% and 50%. Multiplying the lifetime by 1.5 would decrease the ADP by 50% and the GWP by 18%.

Considering the gap between environmental metrics and the scope of our methodology, there is no realistic scenario where Jetson is more advantageous than Apollo on the ResNet-50* FU.

Going beyond those results, Chapter 6 discusses the limits of this methodology and perspectives for future work.

6

Conclusion, discussion, and perspectives

Contents

In this thesis, we propose a methodology to assess the environmental impacts of the training phase of a Machine Learning (ML) model, with a focus on electricity consumption and how to account for the embodied impacts of the manufacturing phase of the hardware life cycle. We define 3 environmental impact indicators and an allocation key to attribute the embodied cost to the training phase, relying on the Life Cycle Assessment (LCA) standard methodology. We developed this methodology with two objectives. First, it enables an analysis of the electricity usage of the computing infrastructure adequate to find leverages to reduce the electricity consumption. Secondly, it can be used to compare computing infrastructures and to make a decision on the infrastructure that is optimal for a given use case. As a consequence, this methodology is versatile, insightful, and reproducible. To highlight those qualities, we selected 6 models from different areas of ML from the MLPerf benchmark and we trained them on two computing infrastructures specialized for AI: a supercomputer, the Apollo 6500 Gen10+ node from the HPE Champollion cluster, and an edge device, Nvidia Jetson AGX Xavier. Thus we defined 7 Functional Unit (FU), one for each model and an additional one, ResNet-50*, with a reduced quality target to enable the comparison between the infrastructures since the Jetson node couldn't achieve the target of the ResNet-50 FU.

To be able to apply our methodology, we first had to select an electricity measuring tool suitable for computing nodes. To accomplish this, in Chapter 3, we conducted an extensive comparison of software-based power meters to understand how electricity measuring tools work, which one best suits the need of the methodology, and what limits we must consider when assessing the electricity consumption of a computing program.

As a first analysis, we studied the electricity consumption of training each FU on an Apollo node and the ResNet-50* FU on a Jetson node. We study the power profile and the correlation between the characteristics of models and the FU electricity consumption. Additionally, we evaluate the BERT FU on up to 4 Apollo nodes and show that increasing the number of nodes is not a leverage to reduce the electricity consumption. An analysis of the energy required to reach each accuracy point shows that this metric can be used as an early stop criterion.

Secondly, we estimate the embodied costs for each component of the Apollo and the Jetson nodes and show that the memory (RAM, GPU Memory chip) is the most impactful component. Then, we allocate them to the FUs and found that the usage phase dominates the Primary Energy (PE) and the Global Warming Potential (GWP) impacts while the embodied phase represents most of the Abiotic Depletion Potential (ADP) indicator.

A comparative analysis of both nodes using the ResNet-50* FU demonstrates the effectiveness of our methodology in assessing not only the electricity consumption but also broader environmental impact indicators. While the Jetson node exhibited a lower embodied carbon footprint, its computational efficiency was significantly outperformed by the Apollo nodes, resulting in an at least five-fold efficiency advantage for the Apollo node across all impact categories.

Reproducibility is a significant aspect of our methodology. Section 6.1 presents the steps we followed to ensure reproducibility. Experiments we conducted and their outcome have shown some limitations to our methodology for assessing the environmental impacts of ML training. Section 6.2 provides an overview of those limits. Section 6.3 explores perspectives to improve our methodology.

6.1 Reproducibility

Reproducibility has many advantages. Transparently documenting experimental protocols, datasets, and analytical methods facilitates independent verification thus enhancing the reliability and validity of the research. Additionally, open access to these materials fosters collaboration

and accelerates scientific progress. In this thesis, we control the experimental setup by fixing the environment and parameters affecting electricity consumption, as outlined in each chapter. Specifically, CPU and GPU frequencies, as well as idle time before execution, were held constant to minimize power fluctuations.

The code that was used to conduct the experiments presented in this thesis can be found in the following repositories:

1. Repository for Chapter 3: https://github.com/vladostp/an-experimental-comparison-of-software-based-power-meters (folder GPU Benchmarks), Persistent identifier: https://hal.inria.fr/hal-03974900. Those artifacts allowed [Jay2023] to be awarded a reproducibility badge.

2. Repository of the software-based power meter we used in Chapter 4: https://github.com/TheElectronWill/nvml-sensor, Persistent identifier: https://hal.science/hal-04664358.

3. Repository for Chapter 4 and Section 6.2.3: https://github.com/mjay42/Assessing-the-electricity-consumption-of-ML-training.

4. Spreadsheet for Chapter 5: https://docs.google.com/spreadsheets/d/e/2PACX-1vTVrRxa05Wb5JTqr3L7Uzq25G9YGoYz1EEqS6oz2_2eu1dBBUr7VtWKiw6H3G0h0MeWe5C4_3Ej5SBY/pubhtml.

6.2 Discussion

First, if the goal is to choose between a supercomputer and an edge device for training a given model, many more criteria need to be considered on top of environmental footprint. Section 6.2.1 proposes a more comprehensive comparison. We based our results on software-based power meter monitoring which fails to encompass the consumption of the whole node, as it is proved in Chapter 3. Section 6.2.2 summarizes challenges we had with measuring electricity consumption. Our methodology also assumes that it is possible to replicate training. However, in many use cases, it can be too expensive in terms of resources and time. Section 6.2.3 shows that it is possible to estimate the total training cost from observations. We identified other limits we didn't have time to explore. The correlation analysis for the Apollo node wasn't conclusive, and more experiments would have been required to collect data from more components like the networking interfaces or the type of storage. LCA databases come with significant uncertainties which we failed to take into account and doing so would strengthen our analysis. We also didn't include the differences in power supply units and storage between the nodes, when both have a significant influence on the total impact of the nodes.

6.2.1 Extending the comparison criteria set for Apollo and Jetson on the ResNet-50* Functional Unit

It was inherently unfair to compare an Apollo node to a Jetson node on the ResNet-50* Functional Unit (See Table 4.4). While a Jetson might outperform a typical web server in terms of computational power, its primary function is edge inference, not training cutting-edge models. At best, it can personalize these models using local data. However, model size is typically tailored to the processing power of the device, favoring smaller, more efficient models.

Our methodology fails to fully capture the strengths of the Jetson as an edge device. It can perform computations at the network's edge, significantly reducing latency and data transfer for

the user. Additionally, it's far more accessible in terms of both development expertise and cost. An Nvidia Jetson AGX Xavier 32GB can be purchased online for a few thousand dollars, while a cluster like Champollion likely costs 100 times more to build and necessitates experienced engineers to operate. In Figure 6.1, we include those new criteria in the comparison.

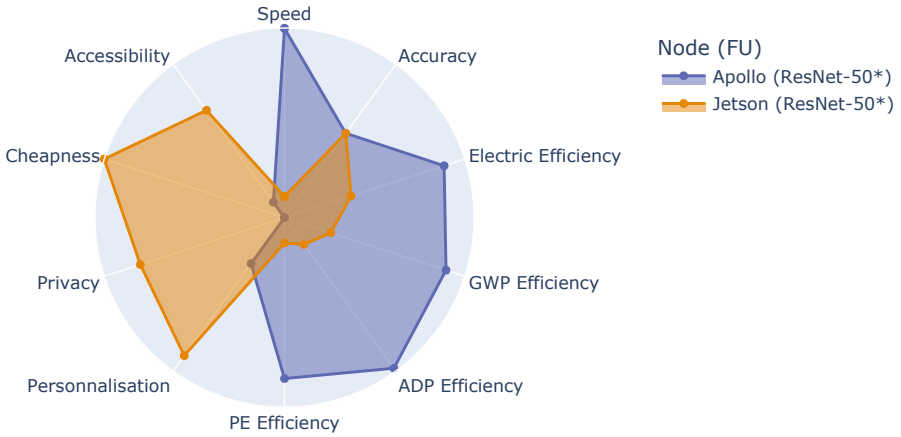


Figure 6.1: Multi-criteria comparison of the ResNet-50 on Jetson and Apollo. For criteria with values outside of the [0,1] interval, values were normalized between 0 and 1, with 1 being the target of the corresponding criteria. Minimum and maximum values are set to (5, 0), (100, 0), (1, 0), (0.00013, 0), (75, 0) for electricity consumption (kWh), duration (hours), GWP (kg CO2eq), ADP (kg Sbeq), and PE (MJ), respectively, thus converting them into efficiency metrics.*

This overview of both infrastructures can help decision-makers choose the best option depending on their needs. It is proof that our methodology can be used to effectively compare computing infrastructures on such functional units.

From our use case, we can conclude that edge devices are not a solution to reduce the electricity consumption of training ML models. In most use cases, they are used in addition to HPC infrastructures thus increasing the total environmental footprint of a model training phase. Alternatively, edge computing can be an opportunity for constraining the learning and inference phases, since it has access to less data and computations. This approach encourages the development of applications focused on sufficiency or lower quality expectations rather than solely performance, which we found to be the most effective strategy for energy reduction.

6.2.2 Challenges in measuring the electricity consumption of computing nodes

In this thesis, we were able to experiment with multiple power meters which came with various challenges.

In Chapter 3, we conducted extensive experiments to study the offset between the power consumed by the computing components (CPUs, GPUs, RAM) and the total power consumed by the computing node. We show that the gap is significant and depends on the node thus it can hardly be predicted for different computing nodes such as Apollo or Jetson nodes.

In Chapter 4, we studied two computing nodes equipped with different power meters than in Chapter 3 and than each other. On Apollo, we had access to the AMD version of RAPL in addition to NVML as software-based power meters and to iLO, an iPDU. We found inconsistencies between them that we were not able to explain. We based all our results on RAPL and NVML since they enable component power consumption reports. On Jetson, the software-based power meter called Jetson-stats or TegraStats reports the power consumption of the CPU and the GPU as well as the total consumption of the node. Surprisingly, the total consumption is higher than the TDP of the node and of the power mode that we set. Before migrating on the Estats cluster of Grid'5000, we made tests on a Jetson development kit that we were able to monitor with an external power

meter called PowerSpy2 [1] that has a resolution of 20 ms and a 99% precision. Figure 6.2 shows the evolution of the power as reported by both power meters on training ResNet-18 on an Nvidia Jetson AGX Xavier 32 Go development kit. We can see that the external power meter reports a power of 30W, which corresponds to the configured power mode. The offset with TegraStats is significant and corresponds to almost a third of the external power meter reports. Despite identical specifications, the development kit and the node don't have the same energy patterns.

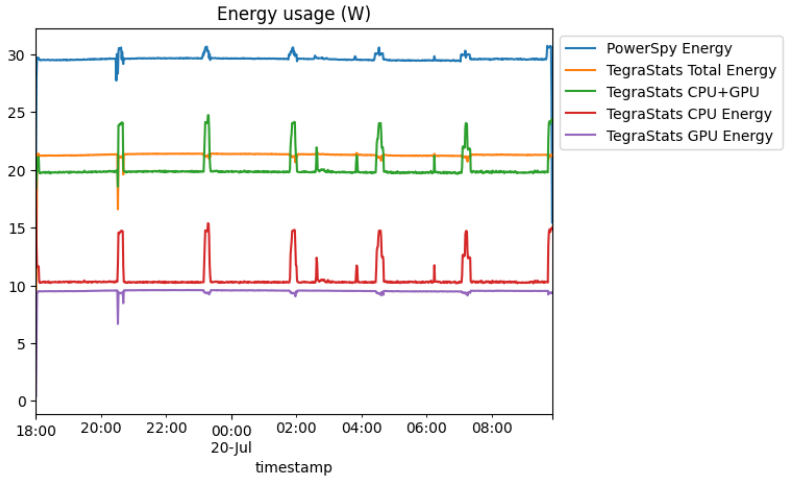


Figure 6.2: Comparison of the power consumption reported by TegraStats and PowerSpy2 on training ResNet-18 on an Nvidia Jetson AGX Xavier 32 Go development kit.

In conclusion, the offset between tools and total node consumption depends on the tool and should be considered when studying the electricity consumption of a node. There are many opportunities to improve the quality and reliance of existing power meters. Additionally, it is necessary to add tools to be able to take into account the entire node as well as the networking components to better understand the total impact of training ML models.

6.2.3 Trainings too expensive to replicate

Most existing methodologies presented in Chapter 1 require replication of the training, as we did in this thesis, to evaluate the electricity consumption of training. In many use cases, it can be too expensive in terms of resources and time.

As an example, training a generative AI model such as Stable Diffusion demands weeks of computations using 32 nodes. In this section, we replicate a fraction of the training while monitoring the electricity consumption and estimate the total training based on those observations. [Anthony2020] had already proven that the electricity consumption of epochs is constant and could be predicted from the first epoch. We show that this characteristic can be used to estimate the total training electricity cost by replication after the training is completed, assuming sufficient information from the original training. Our approach is open-sourced and reproducible.

This section is based on a joint work [Berthelot2024]. It corresponds to my contribution to the article.

We illustrate our approach on Stable Diffusion [Rombach2022], an open-source text-to-image generative deep-learning model. Stable Diffusion was developed by researchers from the CompVis Group at Ludwig Maximilian University of Munich and Runway with a compute donation by Stability AI and training data from non-profit organizations. We selected Stable Diffusion because it is popular, its model is open-sourced, and its successive versions can be downloaded on Hugging

[1] https://www.alciom.com/en/our-trades/products/powerspy2/

Face [2]. Additionally, image generation is the most energy-consuming LLM task [Luccioni2023a]. Several versions of the model exist from successive training phases from v1-0 to v1-5. The first model version used by the service at the start of August 2022 was v1-4, which was replaced by v1-5 two months later. The number of steps that were required for versions v1-1, v1-4, and v1-5 are displayed in Table 6.1.

We executed experiments on nodes from the Sirius cluster, whose specifications are described in Appendix C.1, of the large-scale experimental Grid'5000 platform [Balouek2013]. This cluster was selected because of its similarity with the resources used by developers for the training and inference of the Stable Diffusion model.

For all experiments, we used Ubuntu 20.04 and we installed an Nvidia GPU driver with the default power management configuration. The power consumption of the Sirius cluster is monitored by an Omegawatt [OmegaWatt2018] power meter, which has a precision of 0.1 watts (W). We used it with a sampling frequency of 1 Hz. Additionally, we gathered power metrics from Nvidia NVML and Intel RAPL at a sampling frequency of 2 Hz. To ensure reproducibility, all results are averaged from seven experiments. We based our experiments on the Diffusers library and the Accelerate optimizer framework.

We were able to train the v1-1 Stable Diffusion model on Sirius with the same gradient accumulation, batch size, and optimizer as originally. The learning rate was kept constant. The original training was distributed across 32 nodes. Assuming that the energy consumed by each node is equivalent, we carried out the experiments on a single node. We used the Pokemon BLIP captions dataset [3] which contains 833 images with captions. A linear regression was trained on data points gathered from 61 training experiments with 7 to 3500 training steps, and we tested it on 6 experiments with 5000 to 6500 training steps. Two resolutions of images were used for the original training, 256x256 and 512x512. Thus we conducted the experiments and built a regression for each resolution (Equation 6.1 and 6.2, respectively). Those regressions were validated with a score higher than 99%.

$$\text{Energy (kWh)} = 5.26e^{-04} \times N + 2.01e^{-02} \tag{6.1}$$

$$\text{Energy (kWh)} = 1.78e^{-03} \times N + 1.64e^{-02} \tag{6.2}$$

Where N is the number of training steps.

Table 6.1 presents the estimated energy consumed by the model versions that are pertinent for this work, based on the number of steps provided by the developers of Stable Diffusion and our regression models. The estimated energy was multiplied by the number of nodes originally used (32).

In conclusion, we show that it is possible to estimate the total training electricity consumption from the replication of a few training steps. To do so, we need the model to be open-sourced, a cluster with similar computing capacity equipped with a power meter, and some basic knowledge on the training such as the number of steps and the specification of the cluster that was used.

[2] https://huggingface.co/runwayml/stable-diffusion-v1-5

[3] https://huggingface.co/datasets/lambdalabs/pokemon-blip-captions

Version	Image size	# steps	Estimated energy (kWh)	
			1 node	32 nodes
v1-1	256	$2.37e^{+05}$	$4.70e^{+02}$	$1.50e^{+04}$
	512	$1.94e^{+03}$		
v1-4	512	$2.25e^{+05}$	$4.01e^{+02}$	$1.28e^{+04}$
v1-5	512	$5.95e^{+05}$	$1.06e^{+03}$	$3.39e^{+04}$

Table 6.1: Estimated electricity consumption of training Stable Diffusion (number of steps provided by the developers)

6.3 Perspectives

We explore three perspectives to improve our methodology. Extending the scope to include the data collection and processing phase as well as the model deployment could be done using an LCA methodology. This is discussed in Section 6.3.1. Our methodology can be used to assess the direct impact of model training, but it fails to evaluate its indirect impact on society thus in Section 6.3.2, we propose a few solutions to enable the evaluation of the indirect effects of the development of models. Finally, Section 6.3.3 lists other impact categories that would need to be considered to assess the sustainability of ML models.

6.3.1 A more complete LCA of model development and deployment

The ML development phase that was mostly studied in this thesis - training - is only one of the numerous phases of an ML model life cycle (Figure 1.5). The collection of data and its preprocessing is often discarded because it is assumed that it is only done once in the life cycle of a model. However, in many cases, data is a continuous stream that needs to be processed and the collection phase becomes more significant.

The impact of model deployment might be more difficult to assess since it depends on user behavior. In [Berthelot2024], we replicate the inference of Stable Diffusion v1-1 to measure the electricity consumption of using the model. Stable Diffusion has the particularity of being deployed online as a service[4] and can be freely accessed from user terminals. The Functional Unit we consider is the cost of the service for one year, covering the activity periods of the v1-4 and v1-5 versions of the model (2 and 10 months) before a new one is proposed on the site.

We use web measurement tools to evaluate the number of users that used this AI-based web service in one year (2022), the average time they spend on the website, and the amount of data that is downloaded and uploaded from the data center to the user terminal. Those measurements can be used to allocate the world average impacts of user terminals and networks to the service. The training is allocated considering the time during which the version is deployed in the year we study.

Figure 6.3 shows the share of each digital infrastructure involved in the service in the total impacts of the FU. We found that in the case of Stable Diffusion deployed online, the share of inference in the total environmental cost of the development and deployment of the model for one year dominates the share of training. The training accounts for 10% of the GWP impact and 8% of the PE impact.

[4] https://stablediffusionweb.com/

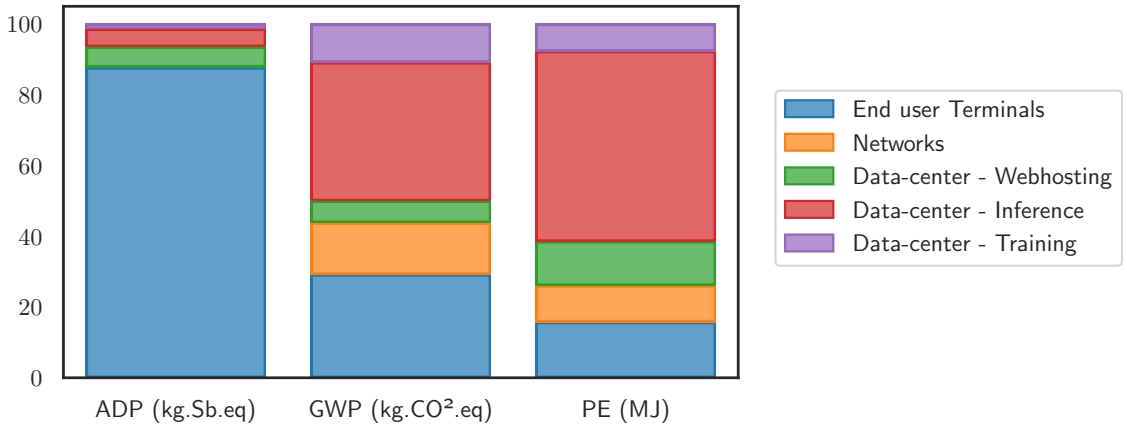


Figure 6.3: Share (in percentage) of each digital infrastructure involved in the service in the total impacts of the FU.

We were able to conduct this study because Stable Diffusion had already been deployed. To use this methodology prior to the deployment, we would need to estimate the amount of users and their average time on the website. The potential behavior of users can be very different from one use case to another. However, this methodology can be used to evaluate various scenarios.

Edge computing can be seen as an opportunity to reduce the network part of the deployment, which is significant for this use case. It is not clear how it might affect the Inference and the End User terminals part. In this thesis, we saw that edge devices are not systematically more energy efficient than HPC infrastructures. Simultaneously, HPC infrastructure can be shared among services thus their active utilization rate is expected to be higher. Edge computing has also the potential to increase the embodied impact of end-user terminals. The MLPerf benchmark and dashboard could be used to assess such scenarios since they cover nodes designed for inference with tiny, mobile, edge, and data center categories. Adding the electricity consumption to the list of performances would facilitate such assessment and the inclusion of environmental criteria into decision-making.

Our methodology could also be used to assess different learning paradigms such as federated learning, transfer learning, fine-tuning, and scenarios combining those concepts and different computing infrastructures. Most of the time, a model is not trained only once but it is continually improved or personalized to a use case. Typically, a model is intensively trained using an HPC infrastructure on a large but generalist dataset. Then it is deployed to user devices where it is continuously improved on local data - which is called fine-tuning or personalization - or via federated learning.

As a consequence, instead of being seen as an alternative solution to data center computing, edge computing could be seen as a complementary field, bringing ML to applications and industries where it would not be possible otherwise due to privacy or latency limitations. For example, Federated Learning is being used in the medical field to improve models without centralizing highly sensitive patient data. The use case of this thesis suggests that the environmental impact of such scenarios can be significant and should be assessed.

6.3.2 Consequential Approach in LCA

Accounting for the embodied impact of the infrastructure is an important challenge. The attributional approach we used in this thesis has many limits. It suggests that computing infrastructures exist already and that services share them and can be used to analyze the impacts of a product or service at a micro-level. For example, in Chapter 5, the sensitivity analysis enabled us to say that multiplying the lifetime by 1.5 could decrease the Abiotic Depletion Potential (ADP) by 48% for

the Apollo node on the ResNet Functional Unit (FU) if other parameters are kept constant. However, doing so could have impacts outside of the service level. Increasing the lifetime of clusters means that more infrastructures are available, but it can have various impacts on the economy. For example, it might lead to producing less hardware, or, at the opposite, to more ML models being developed and deployed.

The attributional approach fails to assess those impacts thus it is not suitable to assess the impact of the service at the macro or economic level. In other words, it doesn't take into account the indirect effects of the service. However, AI workloads have a significant impact on the economy. As it was presented in Chapter 1, more powerful and efficient hardware is designed and put into the market every few years to allow ever bigger models to be trained and deployed at a larger scale.

Another approach, called consequential LCA, estimates the consequences of changing a part of the economy. Its goal is to answer the questions "What environmental impact product X is responsible for? What are the consequences of consuming X?" instead of "What environmental impact can be attributed to product X?". It is better for decision-making at the macro scale or strategic level. However, it is more challenging to model. It relies on marginal life cycle inventory databases that are less common and the impact of the choice of service might not be linear. It requires modeling the relationship between every actor influencing the sector: the capacity of chip manufacturers to produce chips and how it is influenced by demand, the investment of ML companies in hardware, the advances in research, the impact of ML workloads on productivity and society, etc.

The discussion on edge computing in the previous Section suggests that edge computing could be seen as an indirect effect of ML training thus it would be interesting to integrate it into a consequential approach. Many factors might influence the advancement of edge computing and its environmental impacts and a consequential approach has the potential of shedding light on the still open question of whether edge computing can help reduce the impacts of the sector or exacerbate them.

6.3.3 Assessing the sustainability of an ML model

Environmental indicators used in this thesis are not enough to evaluate the sustainability of a service. Models were proposed to assess sustainability as a fair allocation of planet boundaries to each human [Hjalsted2021]. In other words, each inhabitant could have a budget for each indicator that, scaled up, would respect planet boundaries. And if the budget remaining after considering food, housing, or transport is enough to support the service, then it might be called sustainable. This is only an example of how sustainability can be assessed, and this aspect could be explored to improve the methodology.

While environmental considerations are crucial, sustainability encompasses more criteria. In 2015, the United Nations established the 17 Sustainable Development Goals addressing issues like health, poverty, and education. The EU AI Act, which came into effect in August 2024, regulates the development and use of AI. Beyond environmental impact, it emphasizes security risks, transparency, traceability, and ethical considerations[5]. Therefore, our proposed methodology should be used in conjunction with other assessment frameworks to comprehensively evaluate an ML model's impact on society.

[5] https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

General conclusion

Computing infrastructures handling Machine Learning (ML) trainings are consuming more electricity than ever when the carbon intensity of electricity is still significant. Simultaneously, the growing amount of computations required by ML workloads pushes for more powerful hardware with an impactful manufacturing process. More advanced evaluation methods and a deeper understanding of these impacts are necessary to minimize the environmental impacts of the field.

This thesis introduces a novel methodology for assessing the environmental impact of ML model training, with a particular emphasis on electricity consumption and the embodied impact of the computing infrastructure. A study is conducted to compare power meters on their capacity to assess the electricity consumption of IT infrastructures and by extension understand how electricity measuring tools work, identify their limits, and select the one that best suits the needs of the methodology. To validate our methodology, we selected 6 models from different areas of ML from the MLPerf benchmark and we trained them on two computing infrastructures specialized for ML: a supercomputer, the Apollo 6500 Gen10+ node from the HPE Champollion cluster, and an edge device, Nvidia Jetson AGX Xavier. We demonstrate the versatility, reproducibility, and insightfulness of the proposed approach by comparing both infrastructures. The environmental impact of the manufacturing and usage phases of the training are assessed through 3 indicators: the global warming potential, the primary energy, and the abiotic depletion potential.

Key findings include significant variation in energy consumption among models and hardware platforms, a dominance of memory components on embodied environmental impact, and a superiority of the high-performance computing infrastructure over the edge device despite a higher embodied impact. Recommendations for future research encompass expanding the analysis to include the data collection and model deployment phases, refining power measurement techniques, and developing methods to evaluate the broader societal implications of ML.

We hope this work will contribute to a better and broader assessment of ML workloads thus promoting transparency and responsible practices in this industry.

Bibliography

[Agency2024] International Energy Agency. *Electricity 2024 - Analysis and forecast to 2026*. Rapport technique, 2024.

[Ahmed2021] Kazi Main Uddin Ahmed, Manuel Alvarez et Math H. J. Bollen. *A Novel Reliability Index to Assess the Computational Resource Adequacy in Data Centers*. IEEE Access, vol. 9, pages 54530–54541, 2021.

[AI2024] Epoch AI. *Parameter, Compute and Data Trends in Machine Learning*, 2024. https://epochai.org/data/epochdb/visualization.

[AMD2017] AMD. *Processor Programming Reference (PPR) for AMD Family 17h Model 01h, Revision B1 Processors*, April 2017.

[Anthony2020] Lasse F. Wolff Anthony, Benjamin Kanding et Raghavendra Selvan. *Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models*. arXiv:2007.03051 [cs, eess, stat], July 2020. arXiv: 2007.03051.

[Arafa2020] Yehia Arafa, Ammar ElWazir, Abdelrahman ElKanishy, Youssef Aly, Ayatelrahman Elsayed, Abdel-Hameed Badawy, Gopinath Chennupati, Stephan Eidenbenz et Nandakishore Santhi. *Verified Instruction-Level Energy Consumption Measurement for NVIDIA GPUs*. Proceedings of the 17th ACM International Conference on Computing Frontiers, pages 60–70, May 2020. arXiv: 2002.07795.

[Araujo2021] Gabriell Araujo, Dalvan Griebler, Dinei A. Rockenbach, Marco Danelutto et Luiz G. Fernandes. *NAS Parallel Benchmarks with CUDA and beyond*. Software: Practice and Experience, vol. n/a, no. n/a, November 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3056.

[Balouek2013] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Perez, Flavien Quesnel, Cyril Rohr et Luc Sarzyniec. *Adding Virtualization Capabilities to the Grid'5000 Testbed*. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann et Tony Shan, editeurs, Cloud Computing and Services Science, pages 3–20, Cham, 2013. Springer International Publishing.

[Bannour2021] Nesrine Bannour, Sahar Ghannay, Aurélie Névéol et Anne-Laure Ligozat. *Evaluating the carbon footprint of NLP methods: a survey and analysis of existing tools*. In Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, pages 11–21, Virtual, November 2021. Association for Computational Linguistics.

[Barr2019] Jeff Barr. *Amazon EC2 Update*, 2019. aws.amazon.com/blogs/aws/amazon-ec2-update-inf1-instances-with-aws-inferentia-chips -for-high-performance-cost-effective-inferencing/.

[Bedard2010] Daniel Bedard, Min Yeol Lim, Robert Fowler et Allan Porterfield. *PowerMon: Fine-grained and integrated power monitoring for commodity computer systems*. In Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon), pages 479–484, March 2010. ISSN: 1558-058X.

[Benqassem2021] Sofia Benqassem, Frederic Bordage, Lorraine de Montenay, Julie Delmas-Orgelet, Firmin Domon, Etienne Lees Perasso, Damien Prunel et Caroline Vateau. *DIGITAL TECHNOLOGIES IN EUROPE: an environmental life cycle approach*. Rapport technique, 2021.

[Berthelot2024] Adrien Berthelot, Eddy Caron, Mathilde Jay et Laurent Lefèvre. *Estimating the environmental impact of Generative-AI services using an LCA-based methodology*. Procedia CIRP, vol. 122, pages 707–712, May 2024.

[Calvin2023] Katherine Calvin, Dipak Dasgupta, Gerhard Krinner, Aditi Mukherji, Peter W. Thorne, Christopher Trisos, José Romero, Paulina Aldunce, Ko Barrett, Gabriel Blanco, William W.L. Cheung, Sarah Connors, Fatima Denton, Aïda Diongue-Niang, David Dodman, Matthias Garschagen, Oliver Geden, Bronwyn Hayward, Christopher Jones, Frank Jotzo, Thelma Krug, Rodel Lasco, Yune-Yi Lee, Valérie Masson-Delmotte, Malte Meinshausen, Katja Mintenbeck, Abdalah Mokssit, Friederike E.L. Otto, Minal Pathak, Anna Pirani, Elvira Poloczanska, Hans-Otto Pörtner, Aromar Revi, Debra C. Roberts, Joyashree Roy, Alex C. Ruane, Jim Skea, Priyadarshi R. Shukla, Raphael Slade, Aimée Slangen, Youba Sokona, Anna A. Sörensson, Melinda Tignor, Detlef Van Vuuren, Yi-Ming Wei, Harald Winkler, Panmao Zhai, Zinta Zommers, Jean-Charles Hourcade, Francis X. Johnson, Shonali Pachauri, Nicholas P. Simpson, Chandni Singh, Adelle Thomas, Edmond Totin, Paola Arias, Mercedes Bustamante, Ismail Elgizouli, Gregory Flato, Mark Howden, Carlos Méndez-Vallejo, Joy Jacqueline Pereira, Ramón Pichs-Madruga, Steven K. Rose, Yamina Saheb, Roberto Sánchez Rodríguez, Diana Ürge Vorsatz, Cunde Xiao, Noureddine Yassaa, Andrés Alegría, Kyle Armour, Birgit Bednar-Friedl, Kornelis Blok, Guéladio Cissé, Frank Dentener, Siri Eriksen, Erich Fischer, Gregory Garner, Céline Guivarch, Marjolijn Haasnoot, Gerrit Hansen, Mathias Hauser, Ed Hawkins, Tim Hermans, Robert Kopp, Noëmie Leprince-Ringuet, Jared Lewis, Debora Ley, Chloé Ludden, Leila Niamir, Zebedee Nicholls, Shreya Some, Sophie Szopa, Blair Trewin, Kaj-Ivar Van Der Wijst, Gundula Winter, Maximilian Witting, Arlene Birt, Meeyoung Ha, José Romero, Jinmi Kim, Erik F. Haites, Yonghun Jung, Robert Stavins, Arlene Birt, Meeyoung Ha, Dan Jezreel A. Orendain, Lance Ignon, Semin Park, Youngin Park, Andy Reisinger, Diego Cammaramo, Andreas Fischlin, Jan S. Fuglestvedt, Gerrit Hansen, Chloé Ludden, Valérie Masson-Delmotte, J.B. Robin Matthews, Katja Mintenbeck, Anna Pirani, Elvira Poloczanska, Noëmie Leprince-Ringuet et Clotilde Péan. *IPCC, 2023: Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, H. Lee and J. Romero (eds.)].*

IPCC, Geneva, Switzerland. Rapport technique, Intergovernmental Panel on Climate Change (IPCC), July 2023. Edition: First.

[Corporation.2019] NVIDIA Corporation. *Nvidia management library (nvml)*, 2019. https://docs.nvidia.com/deploy/nvml-api.

[David2010] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna et Christian Le. *RAPL: Memory power estimation and capping*. In 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED), pages 189–194, August 2010.

[De Vries2023] Alex De Vries. *The growing energy footprint of artificial intelligence*. Joule, vol. 7, no. 10, pages 2191–2194, October 2023.

[Delamare2021] Simon Delamare et Lucas Nussbaum. *Kwollect: Metrics Collection for Experiments at Scale*. In IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 1–6, Vancouver, BC, Canada, May 2021. IEEE.

[Devlin2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee et Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, May 2019. arXiv:1810.04805 [cs].

[Diouri2014] Mohammed El Mehdi Diouri, Manuel F. Dolz, Olivier Glück, Laurent Lefèvre, Pedro Alonso, Sandra Catalán, Rafael Mayo et Enrique S. Quintana-Ortí. *Assessing Power Monitoring Approaches for Energy and Power Analysis of Computers*. Sustainable Computing: Informatics and Systems, vol. 4, no. 2, pages 68–82, June 2014.

[Dodge2022] Jesse Dodge, Taylor Prewitt, Remi Tachet Des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario et Will Buchanan. *Measuring the Carbon Intensity of AI in Cloud Instances*. In 2022 ACM Conference on Fairness, Accountability, and Transparency, pages 1877–1894, Seoul Republic of Korea, June 2022. ACM.

[El Mehdi Diouri2013] Mohammed El Mehdi Diouri, Olivier Gluck, Laurent Lefevre et Jean-Christophe Mignot. *Your cluster is not power homogeneous: Take care when designing green schedulers!* In 2013 International Green Computing Conference Proceedings, pages 1–10, Arlington, VA, USA, June 2013. IEEE.

[Ferreboeuf2021] Hugues Ferreboeuf, Maxime Efoui-Hess et Xavier Verne. *Environmental impacts of digital technology: 5-year trends and 5G governance*. The Schift Project, 2021.

[Freitag2021] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S. Blair et Adrian Friday. *The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations*. Patterns, vol. 2, no. 9, page 100340, September 2021.

[García-Martín2019] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley et Håkan Grahn. *Estimation of energy consumption in machine learning*. Journal of Parallel and Distributed Computing, vol. 134, pages 75–88, December 2019.

[Ghosh2019] Swarnendu Ghosh, Nibaran Das, Ishita Das et Ujjwal Maulik. *Understanding Deep Learning Techniques for Image Segmentation*. arXiv:1907.06119 [cs], July 2019. arXiv: 1907.06119.

[Graves2012] Alex Graves. *Sequence Transduction with Recurrent Neural Networks*, 2012. Version Number: 1.

[Graves2013] Alex Graves, Abdel-rahman Mohamed et Geoffrey Hinton. *Speech recognition with deep recurrent neural networks*. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 6645–6649, Vancouver, BC, Canada, May 2013. IEEE.

[Gruber2021] Thomas Gruber, Jan Eitzinger, Georg Hager et Gerhard Wellein. *LIKWID*, 2021. https://doi.org/10.5281/zenodo.4275676.

[Gröger2021] Jens Gröger, Ran Liu, Lutz Stobbe et Jan Druschke. *Green Cloud Computing: Lebenszyklusbasierte Datenerhebung zu Umweltwirkungen des Cloud Computing*. Rapport technique, 2021.

[Gupta2022] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks et Carole-Jean Wu. *Chasing Carbon: The Elusive Environmental Footprint of Computing*. IEEE Micro, vol. 42, no. 4, pages 37–47, July 2022. Conference Name: IEEE Micro.

[Hackenberg2013] Daniel Hackenberg, Thomas Ilsche, Robert Schone, Daniel Molka, Maik Schmidt et Wolfgang E. Nagel. *Power measurement techniques on standard compute nodes: A quantitative comparison*. In 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pages 194–204, Austin, TX, USA, April 2013. IEEE.

[Hackenberg2015] Daniel Hackenberg, Robert Schone, Thomas Ilsche, Daniel Molka, Joseph Schuchart et Robin Geyer. *An Energy Efficiency Feature Survey of the Intel Haswell Processor*. In 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, pages 896–904, Hyderabad, India, May 2015. IEEE.

[He2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385 [cs], December 2015. arXiv: 1512.03385.

[He2017] Kaiming He, Georgia Gkioxari, Piotr Dollar et Ross Girshick. *Mask R-CNN*. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 2980–2988, Venice, October 2017. IEEE.

[Heguerte2023] Lucia Bouza Heguerte, Aurélie Bugeau et Loïc Lannelongue. *How to estimate carbon footprint when training deep learning models? A guide and review*, June 2023. arXiv:2306.08323 [cs].

[Henderson2020] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky et Joelle Pineau. *Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning*. Journal of Machine Learning Research 21, vol. 1-43, 2020.

[Hjalsted2021] Anjila Wegge Hjalsted, Alexis Laurent, Martin Marchman Andersen, Karen Holm Olsen, Morten Ryberg et Michael Hauschild. *Sharing the safe operating space: Exploring ethical allocation principles to operationalize the planetary boundaries and assess absolute sustainability at individual and industrial sector levels*. Journal of Industrial Ecology, vol. 25, no. 1, pages 6–19, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jiec.13050.

[Hobbhahn2023] Marius Hobbhahn, Lennart Heim et Gökçe Aydos. *Trends in Machine Learning Hardware*, 2023.

[Hu2019] Jie Hu, Li Shen, Samuel Albanie, Gang Sun et Enhua Wu. *Squeeze-and-Excitation Networks*. arXiv:1709.01507 [cs], May 2019. arXiv: 1709.01507.

[in Data2024] Our World in Data. *Data Page: Mobile phone subscriptions*, 2024. Data adapted from International Telecommunication Union (via World Bank). https://ourworldindata.org/grapher/mobile-cellular-subscriptions-by-country.

[Intel2023] Intel. *Thermal Design Power (TDP) in Intel® Processors*, 2023.

[ITU2014] ITU. *ITU L1410: Methodology for environmental life cycle assessments of information and communication technology goods, networks and services*, December 2014.

[Jay2023] Mathilde Jay, Vladimir Ostapenco, Laurent Lefevre, Denis Trystram, Anne-Cécile Orgerie et Benjamin Fichel. *An experimental comparison of software-based power meters: focus on CPU and GPU*. In 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pages 106–118, Bangalore, India, May 2023. IEEE.

[Khan2018] Kashif Khan, Mikael Hirki, Tapio Niemi, Jukka Nurminen et Zhonghong Ou. *RAPL in Action: Experiences in Using RAPL for Power Measurements*. ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), vol. 3, January 2018.

[Kingma2014] Diederik P. Kingma et Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. Publisher: [object Object] Version Number: 9.

[Krizhevsky2012] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. In Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012.

[Lacoste2019] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt et Thomas Dandres. *Quantifying the Carbon Emissions of Machine Learning*. Rapport technique arXiv:1910.09700, arXiv, November 2019. arXiv:1910.09700 [cs] type: article.

[Lannelongue2021] Loïc Lannelongue, Jason Grealey et Michael Inouye. *Green Algorithms: Quantifying the Carbon Footprint of Computation*. Advanced Science, vol. 8, no. 12, page 2100707, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/advs.202100707.

[Laros2013] James H. Laros, Phil Pokorny et David DeBonis. *PowerInsight - A commodity power measurement capability*. In 2013 International Green Computing Conference Proceedings, pages 1–6, June 2013.

[Lecun1998] Y. Lecun, L. Bottou, Y. Bengio et P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, vol. 86, no. 11, pages 2278–2324, November 1998.

[Leopold2019] George Leopold. *AWS to Offer NVIDIA's T4 GPUs for AI Inferencing*, 2019.

[Li2023] Pengfei Li, Jianyi Yang, Mohammad A. Islam et Shaolei Ren. *Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models*, October 2023. arXiv:2304.03271 [cs].

[Libertson2021] Frans Libertson, Julia Velkova et Jenny Palm. *Data-center infrastructure and energy gentrification: perspectives from Sweden*. Sustainability: Science, Practice and Policy, vol. 17, no. 1, pages 152–161, January 2021.

[Ligozat2022] Anne-Laure Ligozat, Julien Lefevre, Aurélie Bugeau et Jacques Combaz. *Unraveling the Hidden Environmental Impacts of AI Solutions for Environment Life Cycle Assessment of AI Solutions*. Sustainability, vol. 14, no. 9, page 5172, April 2022.

[Liu2014] Jingping Liu, Cheng Yang, Haoyi Wu, Ziyin Lin, Zhexu Zhang, Ronghe Wang, Baohua Li, Feiyu Kang, Lei Shi et Ching Ping Wong. *Future paper based printed circuit boards for green electronics: fabrication and life cycle assessment*. Energy Environ. Sci., vol. 7, no. 11, pages 3674–3682, 2014.

[Luccioni2023a] Alexandra Sasha Luccioni, Yacine Jernite et Emma Strubell. *Power Hungry Processing: Watts Driving the Cost of AI Deployment?*, November 2023. arXiv:2311.16863 [cs].

[Luccioni2023b] Alexandra Sasha Luccioni, Sylvain Viguier et Anne-Laure Ligozat. *Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model*. Journal of Machine Learning Research, vol. 24, no. 253, pages 1–15, 2023.

[Lugosch2020] Loren Lugosch. *Sequence-to-sequence learning with Transducers*, 2020. https://lorenlugosch.github.io/posts/2020/11/transducer/.

[Madon2022] Maël Madon, Georges Da Costa et Jean-Marc Pierson. *Characterization of Different User Behaviors for Demand Response in Data Centers*. In José Cano et Phil Trinder, editeurs, Euro-Par 2022: Parallel Processing, volume 13440, pages 53–68. Springer International Publishing, Cham, 2022. Series Title: Lecture Notes in Computer Science.

[McMahan2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson et Blaise Aguera y Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, pages 1273–1282. PMLR, April 2017. ISSN: 2640-3498.

[Nassajfar2021] Mohammad Naji Nassajfar, Ivan Deviatkin, Ville Leminen et Mika Horttanainen. *Alternative Materials for Printed Circuit Board Production: An Environmental Perspective*. Sustainability, vol. 13, no. 21, page 12126, November 2021.

[Naumov2019] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong et Misha Smelyanskiy. *Deep Learning Recommendation Model for Personalization and Recommendation Systems*, May 2019. arXiv:1906.00091 [cs].

[OmegaWatt2018] OmegaWatt. *OmegaWatt*, 2018. https://mv.omegawatt.fr/.

[Ozkan2018] Elif Ozkan, Nilay Elginoz et Fatos Germirli Babuna. *Life cycle assessment of a printed circuit board manufacturing plant in Turkey*. Environmental Science and Pollution Research, vol. 25, no. 27, pages 26801–26808, September 2018.

[Patterson2021] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier et Jeff Dean. *Carbon Emissions and Large Neural Network Training*, 2021.

[Patterson2022] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier et Jeff Dean. *The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink*. Rapport technique arXiv:2204.05149, arXiv, April 2022. arXiv:2204.05149 [cs] type: article.

[Patterson2024] David Patterson, Jeffrey M. Gilbert, Marco Gruteser, Efren Robles, Krishna Sekar, Yong Wei et Tenghui Zhu. *Energy and Emissions of Machine Learning on Smartphones vs. the Cloud*. Communications of the ACM, vol. 67, no. 2, pages 86–97, February 2024.

[Pedregosa2011] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos et David Cournapeau. *Scikit-learn: Machine Learning in Python*. MACHINE LEARNING IN PYTHON, page 6, October 2011.

[Petit2020] Benoit Petit. *Scaphandre*, 2020. https://github.com/hubblo-org/scaphandre.

[Pirson2021] Thibault Pirson et David Bol. *Assessing the embodied carbon footprint of IoT edge devices with a bottom-up life-cycle approach*. Journal of Cleaner Production, vol. 322, page 128966, November 2021. arXiv:2105.02082 [cs].

[Prakash2023] Shvetank Prakash, Matthew Stewart, Colby Banbury, Mark Mazumder, Pete Warden, Brian Plancher et Vijay Janapa Reddi. *Is TinyML Sustainable?* Communications of the ACM, vol. 66, no. 11, pages 68–77, November 2023.

[Qiu2021] Xinchi Qiu, Titouan Parcollet, Daniel J. Beutel, Taner Topal, Akhil Mathur et Nicholas D. Lane. *Can Federated Learning Save The Planet?* arXiv, April 2021. arXiv:2010.06537 [cs].

[Qiu2024] Xinchi Qiu, Titouan Parcollet, Javier Fernandez-Marques, Pedro P. B. Gusmao, Yan Gao, Daniel J. Beutel, Taner Topal, Akhil Mathur et Nicholas D. Lane. *A first look into the carbon footprint of federated learning*. The Journal of Machine Learning Research, vol. 24, no. 1, pages 129:5899–129:5921, March 2024.

[Radford2018] Alec Radford, Karthik Narasimhan, Tim Salimans et Ilya Sutskever. *Improving Language Understanding by Generative Pre-Training*, 2018.

[Raffin2023] Guillaume Raffin et Mathilde Jay. *NVML sensor (CPU+GPU)*, 2023. https://github.com/TheElectronWill/nvml-sensor.

[Rasoldier2022] Aina Rasoldier, Jacques Combaz, Alain Girault, Kevin Marquet et Sophie Quinton. *How realistic are claims about the benefits of using digital technologies for GHG emissions mitigation?* June 2022.

[Reback2022] Jeff Reback, Jbrockmendel, Wes McKinney, Joris Van Den Bossche, Matthew Roeschke, Tom Augspurger, Simon Hawkins, Phillip Cloud, Gfyoung, Sinhrks, Patrick Hoefler, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Shahar Naveh, JHM Darbyshire, Richard Shadrach, Marc Garcia, Jeremy Schendel, Andy Hayden, Daniel Saxton, Marco Edward Gorelli, Fangchen Li, Torsten Wörtwein, Matthew Zeitlin, Vytautas Jancauskas, Ali McMaster et Thomas Li. *pandas-dev/pandas: Pandas 1.4.3*, June 2022.

[Richardson2023] Katherine Richardson, Will Steffen, Wolfgang Lucht, Jørgen Bendtsen, Sarah E. Cornell, Jonathan F. Donges, Markus Drüke, Ingo Fetzer, Govindasamy Bala, Werner Von Bloh, Georg Feulner, Stephanie Fiedler, Dieter Gerten, Tom Gleeson, Matthias Hofmann, Willem Huiskamp, Matti Kummu, Chinchu Mohan, David Nogués-Bravo, Stefan Petri, Miina Porkka, Stefan Rahmstorf, Sibyll Schaphoff, Kirsten Thonicke, Arne Tobian, Vili Virkki, Lan Wang-Erlandsson, Lisa Weber et Johan Rockström. *Earth beyond six of nine planetary boundaries*. Science Advances, vol. 9, no. 37, page eadh2458, September 2023.

[Ritchie2022] Hannah Ritchie. *Primary, secondary, final, and useful energy: Why are there different ways of measuring energy?* Our World in Data, 2022. https://ourworldindata.org/energy-definitions.

[Rockström2009] Johan Rockström, Will Steffen, Kevin Noone, Åsa Persson, F. Stuart Iii Chapin, Eric Lambin, Timothy M. Lenton, Marten Scheffer, Carl Folke, Hans Joachim Schellnhuber, Björn Nykvist, Cynthia A. De Wit, Terry Hughes, Sander Van Der Leeuw, Henning Rodhe, Sverker Sörlin, Peter K. Snyder, Robert Costanza, Uno Svedin, Malin Falkenmark, Louise Karlberg, Robert W. Corell, Victoria J. Fabry, James Hansen, Brian Walker, Diana Liverman, Katherine Richardson, Paul Crutzen et Jonathan Foley. *Planetary Boundaries: Exploring the Safe Operating Space for Humanity*. Ecology and Society, vol. 14, no. 2, page art32, 2009.

[Rombach2022] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser et Bjorn Ommer. *High-Resolution Image Synthesis with Latent Diffusion*

Models. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10674–10685, New Orleans, LA, USA, June 2022. IEEE.

[Ronneberger2015] Olaf Ronneberger, Philipp Fischer et Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In Nassir Navab, Joachim Hornegger, William M. Wells et Alejandro F. Frangi, editeurs, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, volume 9351, pages 234–241. Springer International Publishing, Cham, 2015. Series Title: Lecture Notes in Computer Science.

[Rotem2012] Effi Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann et Doron Rajwan. *Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge*. IEEE Micro - MICRO, vol. 32, pages 20–27, March 2012.

[Russakovsky2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg et Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. arXiv:1409.0575 [cs], January 2015. arXiv: 1409.0575.

[Savazzi2021] Stefano Savazzi, Sanaz Kianoush, Vittorio Rampa et Mehdi Bennis. *A Framework for Energy and Carbon Footprint Analysis of Distributed and Federated Edge Learning*. In 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pages 1564–1569, September 2021. arXiv:2103.10346 [cs].

[Savazzi2022a] Stefano Savazzi, Vittorio Rampa, Sanaz Kianoush et Mehdi Bennis. *An Energy and Carbon Footprint Analysis of Distributed and Federated Learning*. IEEE Transactions on Green Communications and Networking, pages 1–1, 2022. Conference Name: IEEE Transactions on Green Communications and Networking.

[Savazzi_2022b] Stefano Savazzi_, Vittorio Rampa, Sanaz Kianoush et Mehdi Bennis. *On the Energy and Communication Efficiency Tradeoffs in Federated and Multi-Task Learning*. In 2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pages 1431–1437, September 2022. ISSN: 2166-9589.

[Schmidt2021] Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler et Sasha Luccioni. *CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing*. Zenodo, 2021. https://github.com/mlco2/codecarbon.

[Schwartz2019] Roy Schwartz, Jesse Dodge, Noah A. Smith et Oren Etzioni. *Green AI*. arXiv:1907.10597 [cs, stat], August 2019. arXiv: 1907.10597.

[Schöne2019] Robert Schöne, Thomas Ilsche, Mario Bielert, Andreas Gocht et Daniel Hackenberg. *Energy Efficiency Features of the Intel Skylake-SP Processor and Their Impact on Performance*. 2019 International Conference on High Performance Computing & Simulation (HPCS), pages 399–406, July 2019. arXiv: 1905.12468.

[Schöne2021] Robert Schöne, Thomas Ilsche, Mario Bielert, Markus Velten, Markus Schmidl et Daniel Hackenberg. *Energy Efficiency Aspects of the AMD Zen 2 Architecture*. 2021 IEEE International Conference on Cluster Computing (CLUSTER), pages 562–571, September 2021. arXiv: 2108.00808.

[Sen2018] Satyabrata Sen, Neena Imam et Chung-Hsing Hsu. *Quality Assessment of GPU Power Profiling Mechanisms*. In 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pages 702–711, Vancouver, BC, May 2018. IEEE.

[Sevilla2022] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn et Pablo Villalobos. *Compute Trends Across Three Eras of Machine Learning*. In 2022 International Joint Conference on Neural Networks (IJCNN), pages 1–8, Padua, Italy, July 2022. IEEE.

[Simon2024] Thibault Simon, David Ekchajzer, Adrien Berthelot, Eric Fourboul, Samuel Rince et Romain Rouvoy. *BoaviztAPI: a bottom-up model to assess the environmental impacts of cloud services*. In HotCarbon'24 - 3rd Workshop on Sustainable Computer Systems, Santa Cruz, United States, July 2024.

[S.K2022] Prashanthi S.K, Sai Anuroop Kesanapalli et Yogesh Simmhan. *Characterizing the Performance of Accelerated Jetson Edge Devices for Training Deep Learning Models*. Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 6, no. 3, pages 1–26, December 2022.

[Spirals2019] Inria Spirals. *PyJoules*, 2019. https://pyjoules.readthedocs.io/en/latest/index.html.

[Spirals2020] Inria Spirals. *PowerAPI*, 2020. http://powerapi.org/.

[Steffen2015] Will Steffen, Katherine Richardson, Johan Rockström, Sarah E. Cornell, Ingo Fetzer, Elena M. Bennett, Reinette Biggs, Stephen R. Carpenter, Wim De Vries, Cynthia A. De Wit, Carl Folke, Dieter Gerten, Jens Heinke, Georgina M. Mace, Linn M. Persson, Veerabhadran Ramanathan, Belinda Reyers et Sverker Sörlin. *Planetary boundaries: Guiding human development on a changing planet*. Science, vol. 347, no. 6223, page 1259855, February 2015.

[Strubell2019] Emma Strubell, Ananya Ganesh et Andrew McCallum. *Energy and Policy Considerations for Deep Learning in NLP*. Florence, Italy, June 2019. arXiv: 1906.02243.

[Thompson2020] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee et Gabriel F. Manso. *The Computational Limits of Deep Learning*. arXiv:2007.05558 [cs, stat], July 2020. arXiv: 2007.05558.

[UN2015] UN. *Paris Agreement to the United Nations Framework Convention on Climate Change*, December 2015.

[Vasconcelos2023] Miguel Vasconcelos, Daniel Cordeiro, Georges Da Costa, Fanny Dufossé, Jean-Marc Nicod et Veronika Rehn-Sonigo. *Optimal sizing of a globally distributed low carbon cloud federation*. In 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pages 203–215, Bangalore, India, May 2023. IEEE.

[Vaswani2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser et Illia Polosukhin. *Attention is All you Need*. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[Victor Schmidt] Victor Schmidt, Alexandra (Sasha) Luccioni, Alexandre Lacoste et Thomas Dandres. *ML CO2 IMPACT*.

[Weaver2012] Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra et Shirley Moore. *Measuring Energy and Power with PAPI*. In 2012 41st International Conference on Parallel Processing Workshops, pages 262–268, September 2012. ISSN: 2332-5690.

[Wu2022] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin S Lee, Bugra Akyildiz, Max Balandat, Joe Spisak, Ravi Jain, Mike Rabbat et Kim Hazelwood. *Sustainable AI: Environmental Implications, Challenges and Opportunities*. In Proceedings of the 5th MLSys Conference, Santa Clara, CA, USA, 2022.

A

Peer-reviewed contributions

A.1 Conference and workshops articles

- Mathilde Jay, Vladimir Ostapenco, Laurent Lefèvre, Denis Trystram, Anne-Cécile Orgerie, Benjamin Fichel. An experimental comparison of software-based power meters: focus on CPU and GPU. *CCGrid 2023 - 23rd IEEE/ACM international symposium on cluster, cloud and internet computing*, May 2023, Bangalore, India. pp.1-13.
- Christophe Cérin, Mathilde Jay, Laurent Lefèvre, Denis Trystram. A Methodology and a Toolbox to Explore Dataset related to the Environmental Impact of HTTP Requests. *2023 IEEE International Conference on Big Data (BigData) - 3rd International Workshop on Big Data Analytics for Sustainability*, Dec 2023, Sorrento (Naples), Italy. pp.1-10.
- Adrien Berthelot, Eddy Caron, Mathilde Jay, Laurent Lefèvre. Estimating the environmental impact of Generative-AI services using an LCA-based methodology. *CIRP LCE 2024 - 31st Conference on Life Cycle Engineering*, Jun 2024, Turin, Italy. pp.1-10.

A.2 Journals

An extended version of a conference article was accepted and is under the publication process:

- Adrien Berthelot, Eddy Caron, Mathilde Jay et Laurent Lefevre. The environmental impact of generative AI: measuring and understanding. *2024 Special Issue on Sustainability and Computing of Communications of the ACM (CACM)*, 2024.

A.3 Scientific posters

- Mathilde Jay, Laurent Lefèvre, Denis Trystram. Comparaison expérimentale de logiciels de mesure d'énergie : côté GPU. *Compas 2022*, Jul 2022, Amiens, France.

- Adrien Berthelot, Eddy Caron, Mathilde Jay, Laurent Lefèvre. Towards a multi-criteria evaluation of the environmental footprint of generative ai services. *ICT4S 2024 - International Conference on Information and Communications Technology for Sustainability*, Jun 2024, Stockholm, Sweden. pp.1-1, 2024.

A.4 Communications

- Mathilde Jay. Méthodologies pour la résilience de l'intelligence artificielle. *Résilience et IA, Plate-Forme Intelligence Artificielle (PFIA)*. 27 June - 1er July 2022, Saint-Étienne, France. https://ci.mines-stetienne.fr/pfia2022/

- Ma thèse en 3 minutes. *MIAI days*. 9 December 2022, Grenoble, France.

- Mathilde Jay and Vladimir Ostapenco. An experimental comparison of software-based power meters: focus on CPU and GPU. *Journées non thématiques GDR RSD*. 27 January 2023, Lyon, France. https://gdr-rsd.fr/journees2023/

- Mathilde Jay. Évolution de la méthodologie d'évaluation de la consommation énergétique de l'apprentissage et pistes d'amélioration. *GreenDays*. 23 March 2023, Lyon, 2023. https://perso.ens-lyon.fr/laurent.lefevre/greendayslyon2023/programme_greendays2023.html

- Mathilde Jay. Apprentissages frugaux dans des centres de calculs proches de l'utilisateur. *Neuvième journée PERSPECTIVES ET DEFIS DE l'IA sur le thème de « IA et écologie » organisée par L'Association Française pour l'Intelligence Artificielle (AFIA)*. 4 April 2023, Paris, France. https://afia.asso.fr/les-journees/pdia-2023/

- Mathilde Jay. Comparaison de wattmètre logiciels. *EcoCode Challenge*. 5 April 2023, Paris, France. https://www.davidson.fr/blog/davidson-accueille-la-deuxieme-edition-du-challenge-ecocode-dans-ses-locaux

- Danilo Carastan Dos Santos and Mathilde Jay. Impact environnemental de l'IA: La partie immergée de l'IA-iceberg. *Pint of Science*. 22 May 2023, Grenoble, France. https://pintofscience.fr/event/impact-environnemental-de-lia

- Panel discussion at the GreenTech Forum. *La recherche en numérique : écoresponsabilité, défis scientifiques, environnementaux et sociétaux*. 21 Novembre 2023, Paris, France. https://www.greentech-forum.com/2023/programme-evenements

- Mathilde Jay. More than electricity, more than carbon: Assessing the environmental cost of the Stable Diffusion service. *Journées de Recherche en Apprentissage Frugal (JRAF)*. 13-14 December 2023, Grenoble, France. https://jraf-2023.sciencesconf.org/

- Mathilde Jay and Guillaume Raffin. Prendre la mesure de la measure. *GreenDays*. 28 March 2024. https://perso.ens-lyon.fr/laurent.lefevre/greendaystoulouse2024/programme_greendays2024.html

A.5 Program committee and workshop chair

Program committee:

- ICML 2023 Workshop FL
- NeurIPS 2023 Workshop WANT
- NeurIPS 2023 Workshop Diffusion
- ICML 2024 Workshop WANT

Session chair:

- "What computer science research in the Anthropocene?" Workshop. *ICT For Sustainability (ICT4S)*. June 2024, Stockholm, Sweden. https://conf.researchr.org/track/ict4s-2024/ict4s-2024-workshops

A.6 Impact

The methodology we presented in [Berthelot2024] was taken as an example by AFNOR, a leading French standards organization, to develop their general reference framework for Frugal AI [1]. This framework, aiming to measure and reduce the environmental impact of AI, was written by French experts from both the academic and industrial worlds.

The published articles got significant attention, as evidenced by their number of citations on Google Scholar. [Jay2023] was cited 40 times within 16 months, while [Berthelot2024] received 17 citations in the first nine months.

[1] https://www.boutique.afnor.org/fr-fr/norme/afnor-spec-2314/referentiel-general-pour-lia-frugale-mesurer-et-reduire-limpact-environneme/fa208976/421140

B

Mathematical Foundations of Machine Learning and Neural Networks

This Chapter introduces key concepts of ML.

B.1 Mathematical foundations

Artificial Intelligence (AI) is intelligence delivered by a digital system. It might also be defined as a computer system displaying human-like intelligence. AI is a field of research in computer science but can be seen as a field of Information and Communication Technologies as it is integrated into digital services, and relies on the same equipment: user interfaces, networks, and data center components. Machine Learning (ML) is a sub-field of AI in which the system automatically learns from data. ML performs well in many tasks, from detecting objects in images to translating languages.

The concepts behind machine learning were inspired by the human brain. When children learn to name animals, they are shown images and told which animal they correspond to. The process is done repeatedly until they can make out the features differencing the animals and name all the animals in the pool of images. This particular example is a classification task whose goal is to match image inputs to categories, but there exist many other tasks. A regression is similar but outputs are numerical values, for example, predicting the age of a child based on her height. Both tasks are **supervised** tasks, implying that the output is known during learning. On the other hand, **unsupervised** learning does not require truth labels and corresponds to tasks such as creating clusters with similarities among the inputs.

The goal of ML is to model the relationship between inputs $x \in X$ and outputs $y \in Y$. This is done by (1) finding a **model** and (2) providing it with a **set of data examples** $\{x_i, y_i\}_{i=1}^{d} \subset X \times Y$ until the model is able to map the input to the closest output. The first phase is done by searching for the best parametric function $f_\theta : X \mapsto Y$ among a family of functions F_θ to approximate the relationship. The second step consists of adjusting the parameters $\theta \in \Theta$, such that the output of our model $f_\theta(x)$ closely matches the desired output y.

For that purpose, one needs to define a **loss function** L that can be used to minimize the error between $f_\theta(x)$ and y. The choice of loss function needs careful consideration since the shape of

the input, output, and model directly influence its effectiveness. It also needs to be convex to use optimization theory. The **quality** refers to a metric that can be used to evaluate the progress of learning while not being used in the optimization. Indeed, some metrics might be better designed for the task while not having the previously listed requirements. The choice of the targeted quality is also important since reaching the next quality point requires more computations and energy than the previous, as was stated in the introduction.

The objective is to find $\hat{\theta}$ such as Equation B.1 is verified.

$$\hat{\theta} = arg\,min_\theta(L(f_\theta(X), Y)) \tag{B.1}$$

Equation B.1 is a non-convex optimization problem that is not solvable with optimization theory. Thus, $\hat{\theta}$ is approximated using **Gradient Descent**. This optimization technique finds a local minimum by taking repetitive steps down the function's slope. The direction is determined by the gradient ∇f_θ. The next step is computed from Equation GD. That is one of the most significant characteristics of ML since those steps require the same amount of computations. This can simplify the evaluation of the training energy consumption.

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta L(f_\theta, X, Y) \tag{GD}$$

η is a hyperparameter called the **learning rate** that can be used to control the descent and is usually dynamically adjusted during training.

Equation GD is often too expensive to compute on the whole dataset. Stochastic Gradient Descent (SGD) consists of computing the gradient on each sample of the dataset, and the parameters are updated systematically as in Equation SGD. This technique has significant drawbacks since samples can have highly different gradients thus leading to inefficient learning. To avoid this, the dataset is divided into **batches** $X_b \times Y_b$ of several samples of data, and the batch loss gradients are averaged as in Equation MBGD. This algorithm is called mini-batch gradient descent. Applying equation MBGD on one batch is referred to as a **step**. Going through all batches corresponds to one **epoch**. The **batch size** is an important hyperparameter since it is a balance between computation capabilities and learning speed-up.

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta L(f_\theta, x, y) \quad \text{with} \quad x \in X, y \in Y \tag{SGD}$$

$$\theta_{n+1} = \theta_n - \eta \sum_{\{x,y\} \in X_b \times Y_b} \frac{1}{|X_b|} \nabla_\theta L(f_\theta, x, y) \quad \text{with} \quad X_b \subset X, Y_b \subset Y \tag{MBGD}$$

Such a process is referred to as **Centralized Learning**. One drawback is that applying the algorithm on one batch needs to be done on one machine. This can be a bottleneck when the dataset is too large. To solve this issue, learning can be distributed on two levels: model-parallelism and data-parallelism. On the first level, different model parts are trained on different machines on the same data. For simplicity reasons, the second level is preferred: each machine is tasked with one batch (Equation MBGD), and the parameters trained in each machine are aggregated on a central machine when each working machine has finished training.

$$\theta_{n+1} = \frac{1}{m} \sum_{i \in \{1,m\}} \theta_i \tag{B.2}$$

Gradient accumulation consists of accumulating the gradients computed on several batches to simulate a bigger batch size. The gradients are not used to update the parameters after each batch forward and backpropagation passes but they are stored and summed up. After a given number of batches, the sum is used to update the parameters which are sent to the central machine. Gradient accumulation enables to use of bigger batch sizes and reduces the time lost in sharing the data and aggregating.

Many evolutions of stochastic gradient descent have been developed to automatically adapt the learning rate to the learning state and stabilize the batch gradient while staying in a reasonable computation space. Adam [Kingma2014] is the most effective and common of them. The method computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients, with little memory requirements. Such methods are referred to as **optimizers**.

The number of steps or epochs needed to reach the optimal is usually impossible to predict. Because of this, the loss needs to be monitored regularly and the learning is stopped when the loss stops decreasing. This is called **early stopping**.

Over-fitting is a phenomenon that happens systematically in Machine Learning. It means the model has become too specialized or too biased on the dataset and is not able to generalize to new data. To avoid this situation, the dataset is divided into train, test, and validation datasets. Early stopping is based on the test dataset. A trade-off needs to be found between under-fitting and over-fitting. The validation dataset is used when the training is done as a final loss or quality and enables model comparison. There exist many **regularization** techniques to reduce over-fitting. Examples include reducing the batch size and transforming the samples (crop, resize, rotate).

B.2 Model architectures: Neural networks

Many model architectures exist. In this thesis, the focus is put on Neural Network (NN) as they have had a lot of interest from research and academia in the past decades.

Rosenblatt's 1957 tech report introduced the perceptron as "a perceiving and recognizing automaton". Designed to mimic a human neuron, it is also called an artificial neuron. In the human brain, a neuron is activated if the input signals are charged enough. To model this, the inputs $x \in X$ are aggregated in a weighted sum. To represent the threshold, a bias b is added to the sum and the result is passed to an activation function $\sigma : \mathbb{R} \mapsto \{0,1\}$, as in Equation B.3.

$$y = \sigma(\boldsymbol{w}^T \boldsymbol{x} + b) \tag{B.3}$$

A single perceptron enables a linear separation of the space. To model more complex relationships, layers of several neurons can be added to form a Multi-Layer Perceptron (MLP). Each layer neuron output is an input to the next layer, resulting in a densely connected network also called a **Neural Network (NN)**. The weights w and biases b correspond to the parameters $\theta \in \Theta$ defined in Section B.1. A neural network can be represented by the parametric function f_θ.

Evolutions of the artificial neuron have been proposed to model even more complex relationships. A Convolutional Neural Network (CNN) is able to detect features in images, a Recurrent Neural Network (RNN) can process stream data, and a Residual Neural Network (ResNet) can skip neurons to avoid vanishing gradients, thus, enabling learning on much bigger networks.

As explained in the previous section, the parameters are trained using gradient descent. The process used to compute the gradients of a NN is called **backpropagation**. Computing the successive gradients of each of the layers can be computationally expensive, and this is evaluated through the number of Floating Point Operation (FLOP) required to train a model.

Finding the best NN architecture for a given task can be tedious. It became a research problem in itself: Neural Architecture Search. The hyperparameter search is a laborious task too, and various strategies exist.

As more and more tasks are explored with deep learning models, the tendency has become to start from so-called **foundation models**, already trained and effective on one task, and use them as backbones for another task by combining them, fine-tuning them or adding more layers.

Large Language Model (LLM) are NN models with an exceptionally high number of parameters that can comprehend and generate human language text. BERT [Devlin2019] and GPT [Radford2018] are two popular LLMs.

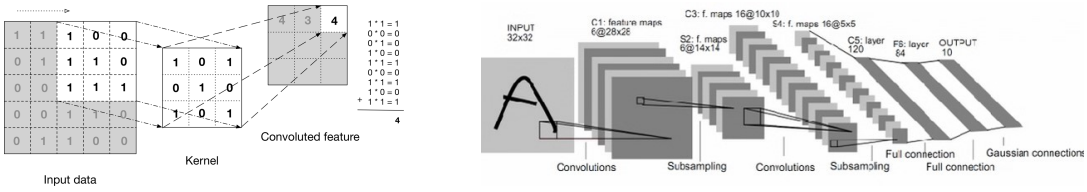
Generative-AI refers to models that can generate text or images. They are known to require more parameters and computations than previously proposed models. The most famous models are ChatGPT, Dall-e, and MidJourney.

B.3 A focus on Convolutional Neural Networks

Convolutional Neural Networks are now the reference in image processing and are used for both image classification and semantic segmentation.

A Convolutional Neural Network (CNN) is based on a mathematical operation called convolution, which expresses how the shapes of two input functions influence each other. It is defined by $f * g(x) = \int f(y)g(x-y)dy$, f and g being the input signals. From this formula, it can be stated that the result function is maximum where the inputs have a similar shape. It becomes the correlation formula in case of identical input. In signal processing, this operation can be used to detect a certain pulse in a signal. By extension in image processing, convolution can be used to detect a certain feature such as a line or a curve in the image.

Typically, the first input of the convolution is the original image and the second is a two-dimension grid called a kernel or a filter. In figure B.1a, the kernel is moved from left to right and from the top to the bottom. For each position, the output of the convolution is the sum of the element-wise product of each value of the kernel with the original image. The output is, therefore, the highest where the image is similar to the kernel. Here, the kernel can be used to detect crosses in the original image.



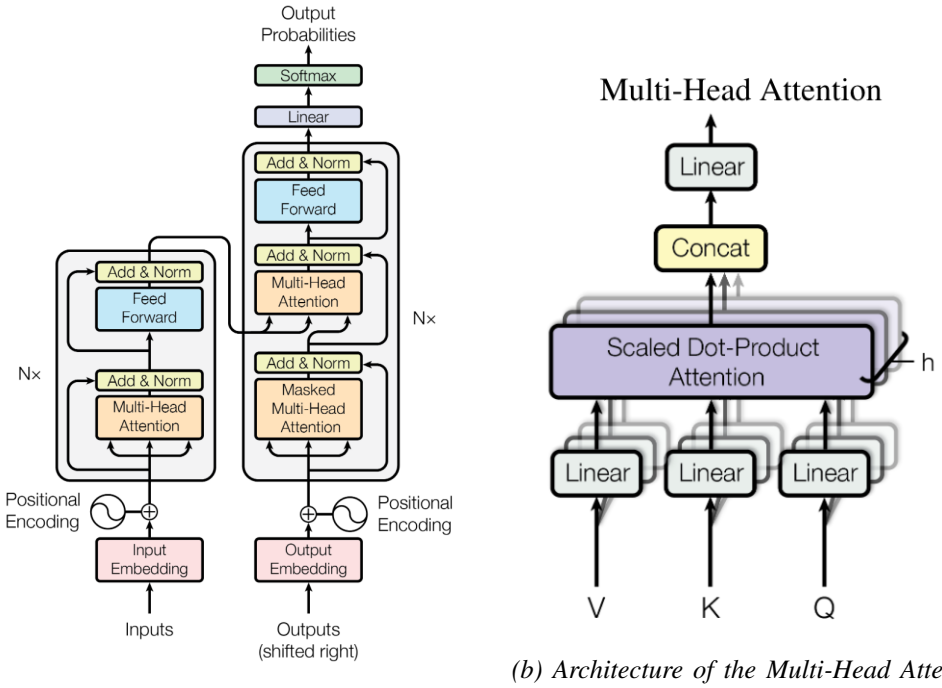
(a) Example of a convolution kernel *(b) LeNet CNN for Handwritten digit recognition [Lecun1998].*

Figure B.1: Details of CNN

A CNN is a succession of convolution layers, as can be seen in figure B.1b. In each layer, several kernels are used to produce different feature maps. The number and the size of kernels can be used to adapt the granularity of details to the problem. To have several levels of details, cascaded layers can be used. A CNN can, therefore, learn a hierarchical representation of the data. Figure B.1b shows one of the first CNNs called LeNet [Lecun1998]. Developed for handwritten digit recognition, the model has two convolution layers. The first one extracts six features from the original image. and the second one, sixteen. Subsampling layers allow for reducing the dimension of the feature maps. Three fully connected layers provide the outputs, here ten, as the number of digits. Classification tasks require linear output in the form of a probability distribution over the number of classes.

B.4 A focus on Transformers

Transformers is a model architecture that revolutionized NLP by enabling efficient and scalable processing of sequential data. It was published in 2017 by Google Brain. Unlike traditional recurrent neural networks (RNNs), Long Short-Term Memory (LSTM) and Gated Recurrent Neural Networks, Transformers utilize self-attention mechanisms to capture dependencies across entire sequences simultaneously, allowing for faster training and better handling of long-range dependencies. This architecture led to the development of advanced models like BERT, GPT, and their successors. Transformer has an encoder-decoder architecture as can be seen in Figure B.2a. Both the encoder and the decoder rely on multi-head attention layers as can be seen in Figure B.2b. Quite similar to kernels in convolution layers, Multi-Head attention allows the model to attend to different aspects of the input simultaneously. Instead of finding patterns and features in images, one head can attend to grammar and another to meaning. An attention layer is more parallelizable than an RNN. The positional encoding (Fig. B.2a) allows the model to keep the sequential order that would be lost in the attention layer.



(a) Architecture of a Transformer [Vaswani2017] *(b) Architecture of the Multi-Head Attention layer [Vaswani2017]*

Figure B.2: Transformer model architectures

C Node specifications

C.1 Champollion

C.1.1 Front view of the cluster

The cluster is placed in racks and cabinets as can be seen in Figure C.1.

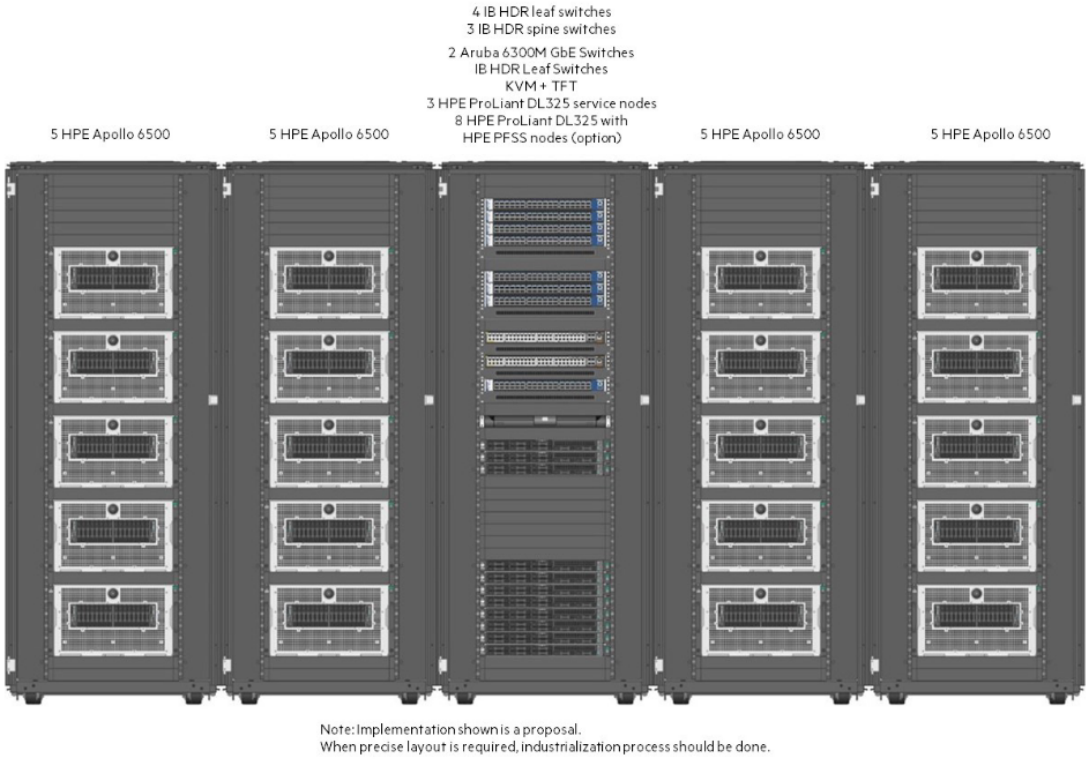


Figure C.1: Front view of Champollion.

C.1.2 Inter-node communications

The inter-node communications are handled by 8 Infiniband Mellanoc switches organized as in Figure C.2.

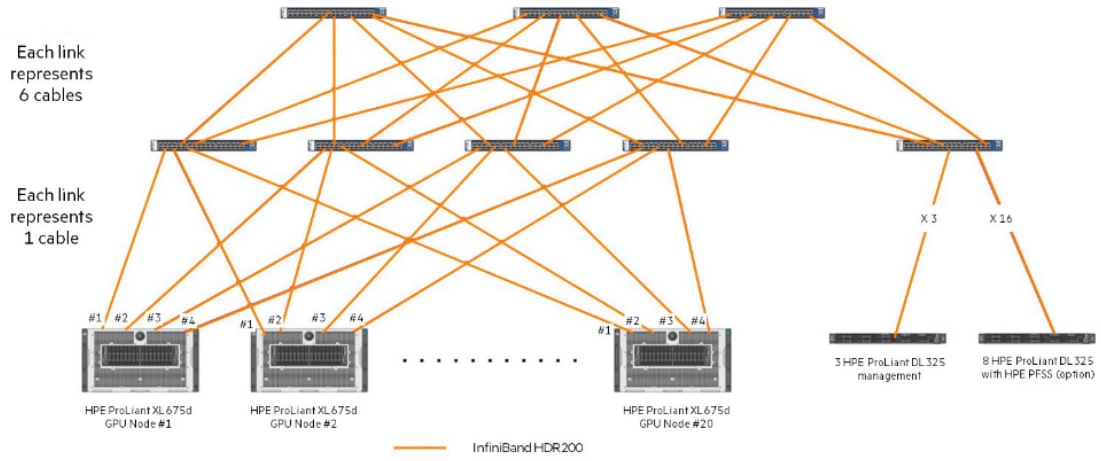


Figure C.2: Organisation of switches in the Champollion cluster.

C.1.3 Inter-GPU communications

The inter-GPU communications rely on the NVLink technology, as can be seen in Figure C.3.

C.2 Jetson

The NVIDIA Jetson AGX Xavier 32 GB development kit (Figure C.4) has a width and a depth of 10 cm. It is 8.7 cm high.

C.3 Sirius

Table C.1 presents the specifications of the Sirius cluster of Grid'5000, a large-scale test bed for experimental research [Balouek2013].

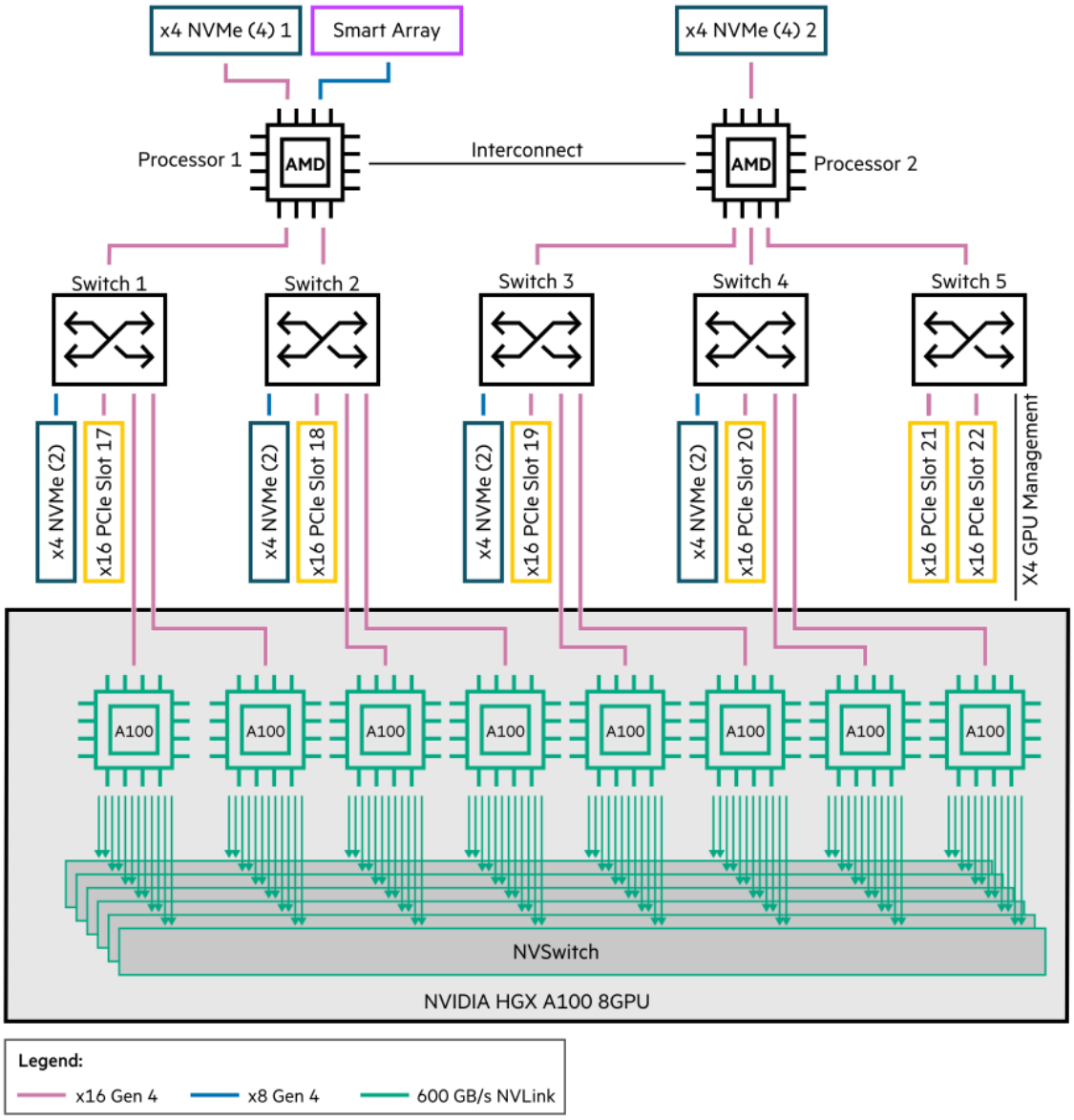


Figure C.3: Schema of inter-GPU communications in Apollo nodes.

Cluster	Sirius
System	Nvidia DGX A100
CPU model	AMD EPYC 7742 (Zen 2, 64 cores/CPU)
# CPUs	2
GPU model	Nvidia A100-SXM4-40GB
# GPUs	8
Memory	1 TB

Table C.1: Experimental setup

Figure C.4: NVIDIA Jetson AGX Xavier 32 GB development kit

List of Figures

List of Tables