Master of Science in Informatics at Grenoble
Master Informatique
Specialization Data science and artificial intelligence

Estimating the environmental impact of AI inference deployment : Large scale VS Edge devices

Djoser SIMEU

Defense Date, 2025

Research project performed at LIG

Under the supervision of:
Danilo CARASTAN DOS SANTOS, Laurent LEFEVRE, Denis TRYSTRAM

Defended to the jury composed of:
President :
Silviu MANIU
Expert :
Thomas BRILLAND

June 2025

This work is partially supported by the French National Research Agency in the framework of the "France 2030" program (ANR-11-LABX-0025-01) for the LabEx PERSYVAL.

Abstract

Nowadays, AI powered technologies are present everywhere, this omnipresence is increasing every year. The computations required by AI models consume a lot of natural resources and have a significant impact on our environment. The purpose of this study is to evaluate whether AI inference can be performed effectively on Edge devices, with a lower environmental impact when compared to deploying on large-scale devices. To answer this question, we developed a new methodology that compares different hardware products used for the deployment of large language model (LLM) inference in terms of associated environmental impacts.
We considered a wide range of AI inference deployments, consisting of numerous devices from large-scale to desktop and Edge devices, released between 2018 and 2023, and also several types of AI inference flows.We developed an environmental impact quantification method that provides us a global warming potential (GWP) estimation for each deployment setting based on a life cycle assessment approach that combines the manufacturing impacts and the use phase impacts. We considered a wide range of computing devices to perform our analysis.
The results of the comparative analysis observed show significant differences in terms of GWP between edge and large-scale devices. From the estimations and the experimental settings, we show that the LLM inference deployed on a large-scale H100 GPU emits around 10 times more CO2 than a deployment on a Jetson AGX Orin edge GPU. We also show that Edge devices can be significantly slower than the other devices considered in this study. We show that it is possible to perform inference on edge devices, and we have characterized the trade-off between performances and environmental impact which occur when using edge devices for AI inference. This study could be useful to (1) shed light on the environmental impacts associated to AI inference by quantifying its impacts, and (2) to help decision-making for people wishing to design AI inference with the lowest environmental impact.

Acknowledgement

I would like to express my sincere gratitude to my supervisors Danilo CARASTAN DOS SANTOS, Laurent LEFÈVRE and Denis TRYSTRAM who drive me in my work to make me learn how to construct a scientific methodology thanks to our frequent meetings. I also want to express my gratitude to Pierre NEYRON, who is a research engineer in the DATAMOVE team. He helped me a lot on all the aspects of my work, specifically on how to deploy various computing systems.

Résumé

De nos jours, les technologies basées sur l'IA sont présentes partout, et cette omniprésence s'accroît d'année en année. Les calculs requis par les modèles d'IA consomment beaucoup de ressources naturelles et ont un impact significatif sur notre environnement. L'objectif de cette étude est d'évaluer si l'inférence de l'IA peut être réalisée efficacement sur des appareils Edge, avec un impact environnemental moindre par rapport à un déploiement sur des appareils à grande échelle.

Pour répondre à cette question, nous avons développé une nouvelle méthodologie qui compare différents produits matériels utilisés pour le déploiement de l'inférence de grands modèles de langage (LLM) en termes d'impacts environnementaux associés. Nous avons pris en compte un large éventail de déploiements d'inférence d'IA, comprenant de nombreux appareils allant des appareils à grande échelle aux appareils de bureau et Edge, sortis entre 2018 et 2023, ainsi que plusieurs types de flux d'inférence d'IA. Nous avons développé une méthode de quantification de l'impact environnemental qui nous fournit une estimation du potentiel de réchauffement global (PRG) pour chaque cadre de déploiement basé sur une approche d'évaluation du cycle de vie qui combine les impacts de la fabrication et les impacts de la phase d'utilisation. Nous avons pris en compte une large gamme d'appareils informatiques pour effectuer notre analyse. Les résultats de l'analyse comparative observés montrent des différences significatives en termes de PRP entre les appareils périphériques et les appareils à grande échelle. À partir des estimations et des paramètres expérimentaux, nous montrons que l'inférence LLM déployée sur un GPU H100 à grande échelle émet environ 10 fois plus de CO2 qu'un déploiement sur un Jetson AGX Orin edge. Nous montrons également que les appareils Edge peuvent être nettement plus lents que les autres appareils pris en compte dans cette étude. Nous montrons qu'il est possible d'effectuer de l'inférence sur des appareils périphériques et nous caractérisons le compromis entre les performances et l'impact sur l'environnement qui se produit lors de l'utilisation d'appareils périphériques pour l'inférence de l'IA. Cette étude pourrait être utile pour (1) mettre en lumière les impacts environnementaux associés à l'inférence de l'IA en quantifiant ses impacts, et (2) aider à la prise de décision pour les personnes souhaitant concevoir l'inférence de l'IA avec le plus faible impact environnemental.

Contents

Abstract i

Acknowledgement i

Résumé i

1 Introduction 1

2 Preliminaries 3
2.1 Preliminary concepts of AI systems inference . . . 3
2.1.1 Neural Network : Multi-Layer perceptron . . . 3
2.1.2 Transformer Architecture . . . 3
Encoder . . . 4
Decoder . . . 4
Tokenization . . . 5
Variations of the transformer architecture . . . 7
2.2 Large scale VS Edge devices . . . 8
2.2.1 Grid'5000 . . . 8
2.2.2 Nvidia Jetson devices . . . 9
2.3 AI Inference benchmark : MLPerf Inference . . . 10
2.3.1 General purpose . . . 10
Queries & samples . . . 10
Scenarios . . . 10
Load Generator . . . 11
2.3.2 BERT model . . . 12
Architecture . . . 12
SQUAD dataset . . . 13
TensorRT parallelization . . . 13

3 Related works 15
3.1 Environmental impacts quantification . . . 15
3.1.1 AI impacts trend . . . 15
3.1.2 LCA methodology and MLCA approach . . . 16
3.2 Previous works limitations . . . 17

3.3 MLPerf inference benchmark limitations . . . 17

4 Methods **19**
4.1 Large-scale to edge devices continuum . . . 19
4.2 Estimating the environmental impact of hardware manufacturing . . . 19
4.3 Performance estimation of Large scale and Edge devices : StarCoder inference . 21
4.3.1 StarCoder Architecture & Complexity . . . 22
4.3.2 StarCoder environmental impact estimation method . . . 23
4.4 Experimental methodology . . . 23
4.4.1 Installation methods . . . 24
Grid'5000 nodes . . . 24
Jetson AGX Xavier . . . 24
Jetson AGX Orin . . . 24
4.4.2 Energy measurements . . . 24
AGX Xavier . . . 25
AGX Orin . . . 25
Grid'5000 GPUs . . . 25
4.4.3 MLPerf inference Compatibility . . . 25
4.4.4 Experimental settings . . . 26

5 Results **29**
5.1 Grid'5000 hardware manufacturing impact . . . 29
5.2 StarCoder case study results . . . 31
5.2.1 Roofline model . . . 31
Edge computing deployment: . . . 31
Large-scale computing deployment: . . . 31
5.2.2 StarCoder environmental impact estimation . . . 32
5.3 Experimental results . . . 33
5.3.1 Power consumption profiles and PUE . . . 33
5.3.2 Inference time analysis . . . 36
5.3.3 Environmental impacts analysis . . . 37
5.3.4 Results analysis . . . 38

6 Conclusion **41**

A Appendix **43**

Bibliography **49**

— 1 —
Introduction

According to the Intergovernmental Panel on Climate Change (IPCC) report published in 2023 [2], the observed global warming is mainly due to human activities, particularly Greenhouse Gas (GHG) emissions. Global warming has a high probability of overtaking 1.5°C in the short term. Each degree of warming will intensify climate risks [2]. Information and Telecommunications Technologies (ICT) contribute to the global warming in the order of 4% of the global GHG emissions [5]. This contribution is quite the same as for the aerial transport industry [6]. Nowadays, Artificial Intelligence (AI) has been largely infused in the ICT industry and fuels its growth. For instance, the huge improvements of AI technology applied to natural language processing (NLP) allowed to deploy AI systems at a large scale (e.g., ChatGPT, Gemini, DeepSeek).

These AI systems comes with a significant environmental cost. Berthelot *et al.* [1] estimated the environmental impacts of the generative AI service "StableDiffusion". They showed that AI systems have non-negligible environmental impacts. For instance, they estimated that in one year of use, the system emits around the same quantity of CO2 as more than five hundred round trips from Paris to New York for one passenger[1]. They showed that the inference phase of the life cycle of the generative AI service emits between 2 and 4 times more CO2 and requires between 4 and 5 times more primary energy than the training phase. This observation, coupled with the growth of deployment of such systems, is alarming in our context of environmental challenge. In addition, Morand *et al.* [10] showed that the environmental impact of the hardware used for the training and the deployment of large AI systems has continuously increased from 2013 to 2023.

It is actually difficult to find data about the effective GHG emissions related to AI systems. This problem is due to the fact that an important part of the AI applications are deployed in large-scale data centers of private companies (e.g., Microsoft Azure, Google Cloud Platform, AWS). Such private companies are not fully transparent in terms of quantification of the GHG emissions associated with their AI systems.

There exists an alternative to this large-scale inference deployment, which consists of using Edge devices to handle the computation required by the inference process. This type of hardware is less energy-consuming, cheaper, it requires less network communication and can be placed closer to the user. However, very little is known about the environmental impacts associated to AI inference deployed on edge devices, when compared to a standard large-scale deployment.

[1] Considering that a round trip Paris-New York emits around 684 KgCO2eq (source: Google Flights).

The purpose of this study is to know whether Edge devices can be more eco-efficient for AI inference compared to large-scale deployments. For this task, we present the following contributions:

- We propose an experimental method that estimates and compares the environmental impacts, in terms of CO_2 equivalent emissions, associated to AI inference under different hardware deployments (including Edge devices) and inference scenarios. We emphasize on the inference due to its significant emissions when compared to training as aforementioned. The method is based on a life cycle assessment method, which allows us to estimate the environmental impacts associated to the manufacturing and use phases of an AI inference system.
- We use the proposed method to perform an experimental campaign with a performance evaluation benchmark built for AI inference systems, and different types of computing accelerator devices, ranging from large-scale to desktop and Edge devices, to evaluate their environmental impacts under numerous inference situations. The results highlight that the edge devices used for AI inference could emit around 10 times less CO_2 equivalent than large scale devices.

We hope that our contributions can help the research community to build environmentally efficient AI inference systems, by choosing the deployment settings and the device type (e.g., Edge, desktop, large-scale) that is most appropriate for their specific needs, and with the lowest level of environmental impacts.

This report starts by introducing the key notions required to understand AI systems' inference process. After that, we present the differences between edge devices and large-scale devices and the complexity related to their differentiation. We present the methodology that we used to quantify the environmental footprint of inference for this study, this methodology is based on life cycle assessment based methodologies from the state of the art. Then we present the set of hardware that we consider in this study based on the set of computing components available on the Grid'5000 platform, we applied the methodology to them to observe some trend in the environmental impact of hardware manufacturing regarding different scales. We also describe the different approach that we used for defining an experimental methodology to measure the execution time and the energy consumption of the considered hardware.

— 2 —
Preliminaries

2.1 Preliminary concepts of AI systems inference

The goal of this section is to recall the mains concepts to understand the operations performed during the inference process. We introduce the preliminary concepts required to understand how the inference is performed at the model level.

2.1.1 Neural Network : Multi-Layer perceptron

The first concept that we want to introduce in this section is the forward propagation of a neural network, and more precisely of a multi-layer perceptron (MLP) because it is a central part of the inference process. MLP are defined by an acyclic neural network organized as a succession of fully connected layers of neurons, from the input layer to the output layer. The inference process corresponds to the forward propagation of the input data through the network. The forward propagation between two consecutive layers can be described in a matrix representation as :

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$
$$\mathbf{a}^{(l)} = \sigma\left(\mathbf{z}^{(l)}\right)$$

where:
- $\mathbf{a}^{(l)}$ is a vector representing the outputs of all neurons in layer l.
- σ is the activation function, a non-linear function applied element-wise.
- $\mathbf{W}^{(l)}$ is the weight matrix for layer l, where each element $w_{ij}^{(l)}$ represents the weight from neuron i in layer $l-1$ to neuron j in layer l.
- $\mathbf{a}^{(l-1)}$ is a vector representing the activations (outputs) of all neurons in the previous layer $l-1$.
- $\mathbf{b}^{(l)}$ is the bias vector for layer l, where each element $b_i^{(l)}$ is the bias term for neuron i.

Then, the output of the model is the output of the forward propagation of the last layer. Figure 2.1 illustrate the forward propagation process of an MLP.

2.1.2 Transformer Architecture

The AI systems considered in this study correspond to the models able to solve natural language processing tasks (NLP) by constructing an abstract representation of the human language

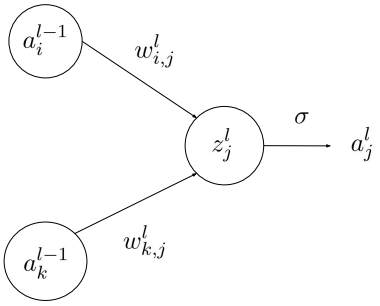

Figure 2.1: Graphical representation of the forward propagation of an MLP network.

through a deep neural architecture. This kind of model can, after training, be deployed for question answering, sentence analysis, or other types of NLP tasks (such as live chat interaction as GPT [1] models). These kinds of models are called large language models (LLM). In 2017, the paper “Attention is all you need" written by Vaswani et al. [14] introduced the Transformer architecture, which is a deep neural architecture based on the attention mechanism initially developed for translation models. This new architecture provides strong parallelization possibilities and achieves new state-of-the-art performances on a huge set of NLP tasks. Nowadays, this architecture is the basis architecture of the majority of the LLM deployed at a large scale (GPT models, Llama models [2], Mistral models [3], etc.). In this section, we present the most important notions behind the inference process of the transformer architecture. Figure 2.2 shows a graphical representation of such architecture; we describe in the next section how each block of the architecture works. As illustrated in Figure 2.2, the transformer architecture is composed of two stacks of blocks : the encoder stack at the left and the decoder stack at the right. Let's take an English-to-French translation context to explain the role of these stacks.

Encoder The encoder takes as input the English sentence that we want to translate, for example, "Hello, how are you ?”. The role of the encoder is to construct an abstract representation of the input sequence in a latent vector space.

Decoder In the other hand, the decoder takes as input the French sentence under construction and gives as output the next word of the sentence. The generation of the translated sequence is an iterative process where we start with an empty sequence and we finish with the full translated sequence. For example, the first forward pass output would be “Bonjour” and the second would be “comment” to construct the translated sentence “Bonjour comment vas tu ?". This process is called an autoregressive generation. The Figure 2.3 illustrates the autoregressive generation principle.

[1] Generative pretrained transformer : `https://openai.com/index/introducing-gpt-4-5/`

[2] `https://www.llama.com/`

[3] `https://mistral.ai/fr/models`

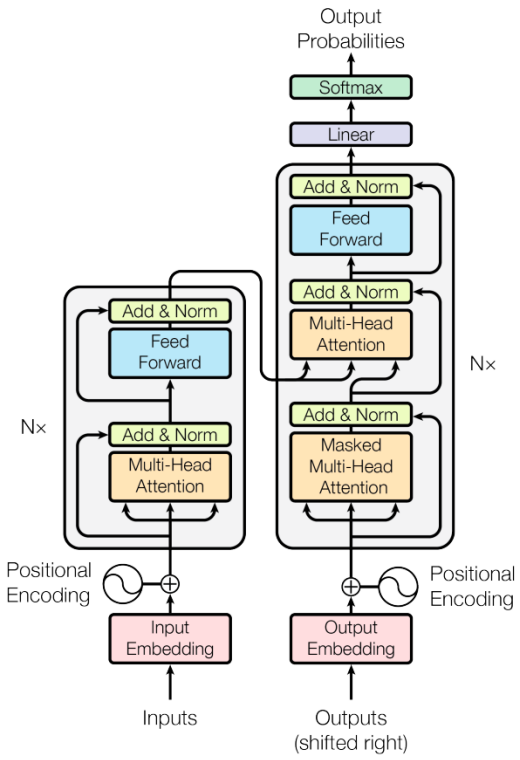

Figure 2.2: Graphical representation of a classical transformer architecture from "Attention is all you need" [14].

The As illustrated in Figure 2.2, both stacks are composed of a succession of N blocks.

Tokenization

The input and output word sequences must be preprocessed before the inference. This is due to the fact that the parameters of the LLM architecture are floating-point values, so it's not possible to perform the forward pass of the model with non-numeric values. The tokenization process consists of splitting the text sequence at the word level and associating each word with a numerical discrete value (a token). This association is made by using a finite set of couples of word-token called vocabulary.

Query, Key, and Value Principle

To understand the computation performed in the inference process of a transformer architecture, we must introduce the queries, keys and values notions on which the transformer architecture is based. The names query (Q), key (K) and value (V) come from the information retrieval (IR) domains, where the user sends a query to the system, the IR system defines the relevancy score of documents by matching the query with their keys (embedded representation of documents), and finally the IR system returns the values of the most relevant documents to the user. The mechanism applied in transformer architecture is quite similar; we will see that in the next

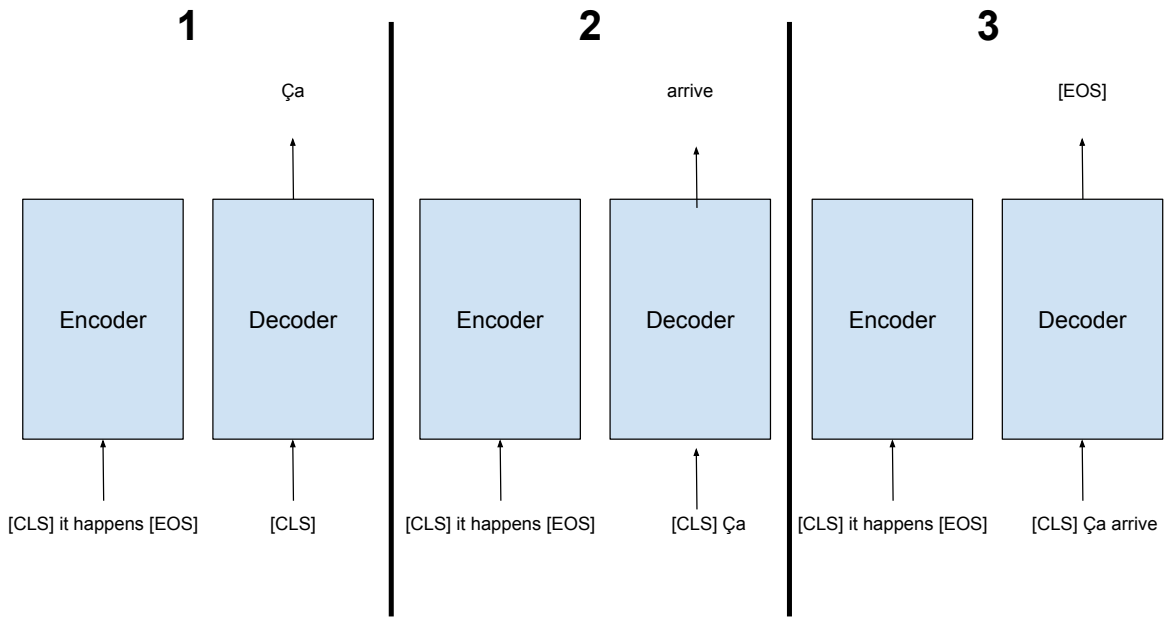

Figure 2.3: Graphical representation of the autoregressive generation process of the transformer architecture with the English sentence "it happens". The [CLS] and [EOS] tokens specify the beginning and the end of the token sequences.

section. The QKV principle is represented in the Figure 2.2 as the three arrow before each attention block of the architecture.

Multi-Head attention block

The multi-head attention block is one of the most important improvements provided by the transformer architectures. Firstly, to obtain the vectors Q, K and V from the embedded representation of the input sequence E[4], we apply learnable projection matrices W_q, W_k and W_v to obtain:

$$Q = E.W_q$$

$$K = E.W_k$$

$$V = E.W_v$$

where :

$$W_q, W_k, W_v \in R^{d_{emb} \times d}$$

[4]The embedded representation of the input sequence is obtained by using a tokenization process plus positional embeddings.

where d_{emb} represents the dimension of the embedding representation space of the sequence and d represents the hidden size of the model (the dimension of the Q, K and V vector space). To facilitate the interpretation, we consider that $d_{emb} = d$. The attention mechanism corresponds to matching the Q and K vectors by using a scaled dot product, using a softmax function on the previous results to have probability distributions for each query, and applying a dot product between the obtained matrix and the V vector of the sequence. A matrix representation of this process is:

$$Attention = softmax(\frac{Q.K^T}{\sqrt{d}}).V$$

This process allows us to capture the internal relation between tokens of the input sequences. Figure **??** shows a graphical representation of this process. The principle of multi-head attention comes from the definition of multiple matrices W_q, W_k and W_v:

$$W_q^i, W_k^i, W_v^i \; where \; i \in \{1, ..., h\}$$

where h represents the number of heads. With this approach, we have an attention process for each head, which allows us to capture multiple levels of relation between the tokens of the sequence; each head is so-called an attention head.

Feed forward block

The second important block of the transformer-decoder architecture is the feedforward block, which corresponds to 1 hidden layer MLP of d input neurons, f hidden neurons, and d output neurons, with an Rectified linear unit ($ReLU$) activation for the hidden layer. It can be expressed by this formula:

$$FFN(x) = max(0, x.W_1 + b_1).W_2 + b_2$$

where W_1 and W_2 are the connection weights between the input layer and the hidden layer and the hidden layer and the output layer, respectively, and b_1 and b_2 are the associated biases.

Variations of the transformer architecture

Lot of models derived the transformer architecture to fit to different NLP use case with less computation or for better performances. There exists a huge variety of variations of transformer architecture. In this study, we will focus our work on two specific types of derived transformer architecture:

- Encoder-only architecture: These architectures are well-fitted for NLP tasks that do not require the generation of tokens. The goal is to map through the encoder stack the input sequence into a latent space where we can easily capture specific characteristics of the sequence. This type of model can be used for sentiment analysis in sentences or for information retrieval systems.

- Decoder-only architecture: These architectures are well-fitted for NLP tasks where the model must generate an output sequence by using the next-token prediction autoregressive process of the decoder stack. This type of architecture can be used for live chat interaction models or transcription models.

2.2 Large scale VS Edge devices

AI systems' inference involves computing resources able to efficiently perform compute-intensive tasks, such as the forward pass of a neural network, as described in Section 2.1. The recent AI models' architecture are built in a way to allow parallel computation to speed up the inference process (such as multi-head parallelization for transformer architecture). The graphical processing units (GPUs), compute-intensive hardware built for efficient parallel computation, are well-fitted to this type of task. Their architecture, defined with the Single Instruction Multiple Thread (SIMT), allows a group of threads to execute the same instruction at the same time on different data streams. GPUs allow programmers to construct computationally efficient algorithms based on parallel patterns. There exists a huge variety of GPUs designed for different use cases. For instance:

- Video Game: GPUs are specialized for Computer Graphics.
- Super-Computer: which requires a high peak performance in terms of Floating point operations per seconds (FLOPS)
- Edge computing: GPUs with a low energy consumption, low price and strong portability.

2.2.1 Grid'5000

In order to perform a comparative analysis, we used the GPUs available on the Grid'5000 platform. Grid'5000 is a large-scale testbed for experiment-driven research. This platform provides access to a large amount of computing resources through the allocation of computing nodes located in the different data centers associated with Grid'5000. Among the resources available on the platform, there is a huge variety of GPUs. We used in this study a sample of GPUs available through Grid'5000 to perform the environmental impact analysis. The selection of this sample was based on the availability of the resources, and we selected only Nvidia GPUs compatible with the experimental methodology. Table 2.1 presents the sample of GPUs selected from the Grid'5000 platform.

Name	Architecture	Memory (GB)	Foundry	TDP (W)	Date
NVIDIA-A100	Ampere	40	TSMC	400	2020
NVIDIA-A40	Ampere	48	Samsung	300	2020
NVIDIA-A5000	Ampere	24	Samsung	230	2021
NVIDIA-AGX-Xavier	Volta	32	TSMC	30	2018
NVIDIA-GTX-1080Ti	Pascal	11	TSMC	250	2017
NVIDIA-L40	Ada Lovelace	48	TSMC	300	2022
NVIDIA-Quadro-P6000	Pascal	24	TSMC	250	2016
NVIDIA-Quadro-RTX-6000	Turing	24	TSMC	260	2018
NVIDIA-Quadro-RTX-8000	Turing	48	TSMC	260	2018
NVIDIA-RTX-2080Ti	Turing	11	TSMC	250	2018
NVIDIA-Tesla-H100	Hopper	80	TSMC	700	2023
NVIDIA-L40S	Ada Lovelace	48	TSMC	300	2022
NVIDIA-Tesla-M2075	Fermi	6	TSMC	225	2011
NVIDIA-Tesla-P100	Pascal	16	TSMC	250	2016
NVIDIA-Tesla-T4	Turing	16	TSMC	70	2018
NVIDIA-Tesla-V100	Volta	32	TSMC	250	2018

Table 2.1: Table of Grid'5000 NVIDIA GPU Characteristics

2.2.2 Nvidia Jetson devices

The Nvidia Jetson hardware are computing accelerator devices built for robotics and embedded computing. In recent years, the emergence of the Internet of Things (IoT) has appeared as the increasing number of connected devices in our daily lives. These connected devices can be seen as data sources; we can take the example of smart buildings with the set of interconnected sensors of the building. With a huge amount of data sources, the global system of the smart building must perform computation from the data collected by the different sensors of the building to make decisions such as turning off the light of the building. Regarding the way how the system can perform the computation, there is an important question to answer, what is better:

- Move the data collected by the sensors to a centralized datacenter to handle computation.
- Move the computing devices closer to the data sources and perform decentralized computations.

With the emergence of AI systems in recent years, the number of users of AI powered applications is nonnegligible, so the question still the same: Is it better to move the prompt of the user to a centralized datacenter or, is it better to move the computing devices closer to the users?
Edge computing has as its goal to develop performant hardware that can be placed closer to the data sources, such as in vehicles, robots, smart buildings, etc. Jetson hardware are types of edge computing devices provided by Nvidia with a significantly lower power consumption compared to classical computing accelerators. Initially, the Jetson devices were designed for robotics perspectives, but with the "AI growth" Nvidia shifted the focus of Jetson devices to be AI inference-specialized edge devices. The Jetson devices considered in this study are presented as developer kits defined by some main components:

- 1 CPU
- 1 GPU
- Flash Disks
- 1 Dynamic Random Access memory (DRAM) shared between the CPU and the GPU
- Connection ports (HDMI, Cameras, etc.)
- a fan system

We used the operating system "Linux 4 Tegra" (L4T) which is a Linux version designed for Nvidia edge computing devices. We also used the Jetpack software development kit, which contains a set of Nvidia software required for AI inference and GPU programming, such as CUDA or TensorRT. The Grid'5000 platform contains 12 Jetson AGX Xavier 32 GB, but the version of L4T currently installed on them is not compatible with the experimental methodology. So, we used for this study a Jetson AGX Xavier 16GB and a Jetson AGX Orin 64GB developer kit available at LIG laboratory.

2.3 AI Inference benchmark : MLPerf Inference

To allow us to monitor different energy consumption profiles corresponding to different deployment scenarios (edge inference, single user, inference server, etc.), we must define a parameterized environment that allows us to execute a set of LLM inferences. To do that, we used the MLPerf inference benchmark [13] which is a benchmarking method for evaluating performances of machine learning (ML) inference systems. In this section, we present the different functionalities of the benchmark that we used.

2.3.1 General purpose

The MLPerf inference benchmark allows us to deploy a test inference platform, this environment has as goal to evaluate the inference performance of an system under test (SUT). This SUT corresponds to a set of computing devices compatible with the benchmark. MLPerf inference is composed of a well-defined architecture of tools that we will define in this section.

Queries & samples

In the MLPerf inference benchmark, the SUT has to process a set of inferences; the inference results are saved and used to compute inference-level metrics such as query latency. The inferences are manipulated by using a two-level representation hierarchy. A sample represents a single inference as a user prompt on the ChatGPT platform.

Scenarios

MLPerf inference's allow the benchmark users to define use case scenarios corresponding to their simulation context. Four simulation scenarios are predefined in the benchmark : offline, single-stream, multi-stream, and server. These scenarios differ by their characteristics: number of queries, number of samples by query, amount of time between two queries, number of inference streams, etc. The Figure 2.4 show the differences between these four scenarios in terms of queries scheduling.

- Offline : This scenario corresponds to a batch-processing inference, where all the samples are sent in a single query. This setting corresponds to an inference task such as people identification in a photo album.

- Single-stream: In this scenario, each query is composed of a single sample; the time between two queries corresponds to the time to process the first one. This setting corresponds to an inference task such as an edge offline voice transcription (Alexa, Google Home, etc.).

- Multi-stream: This scenario corresponds to a series of batch processing inferences. This scenario is similar to the single-stream scenario; there are only two differences:each query contains a batch of samples, and the time between two queries is a fixed parameter. This setting corresponds to an inference task such as computer vision applied to autonomous vehicles, where, at each timestamp, the system must perform inferences on different data coming from the different cameras of the vehicle.

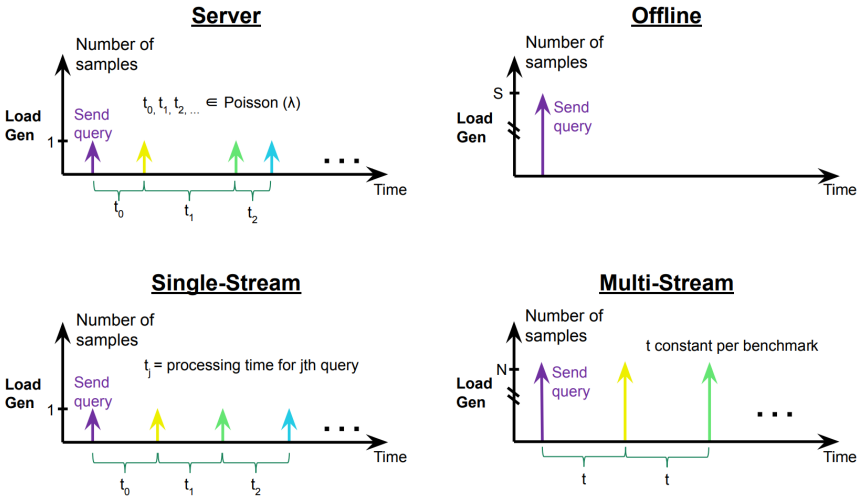

Figure 2.4: Representations of the differences in terms of query scheduling for the different scenarios from the MLPerf inference benchmark paper [13].

- Server: This scenario corresponds to an inference server platform, where each query is composed of a single sample, the queries come from different inference streams, and the time between queries is defined by using the Poisson distribution to simulate the behavior of the users of the platform. This setting correspond to an inference server such as the ones used to host the computations required by ChatGPT inference, where multiple users can try to access concurrently to the same computing resources.

Load Generator

LoadGen works as a traffic generator during the execution of the benchmark. At the end of the execution, LoadGen is used to report performance metrics such as average query latency. The

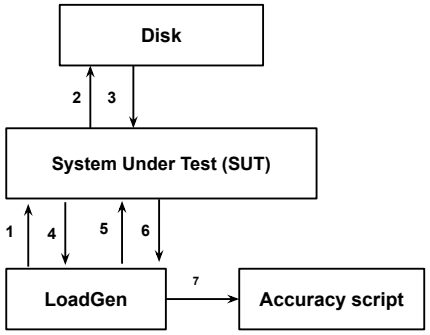

Figure 2.5: Graphical representation of the interaction of the LoadGen module during the execution of the benchmark, taken from [13].

Figure 2.5 show how the LoadGen module interact with the SUT during the execution of the benchmark:

- 1: The LoadGen module sends a request to the SUT to communicate to it the parameters of the benchmark such as the simulation scenarion in use.
- 2-3: The SUT loads the samples and the parameters of the ML model associated to the current benchmark from the disk to the device memory.
- 4: The SUT sends a request to the LoadGen module to start the inference process.
- 5: The LoadGen schedules the query to send to the SUT for the inference process.
- 6: The SUT sends back the results to the LoadGen to save benchmark related data.
- 7: The LoadGen module uses the Accuracy script to verify the validity of the execution and to compute perfomances related metrics.

2.3.2 BERT model

In the MLPerf Benchmark, there is a huge variety of ML models that we can use corresponding to a huge variety of tasks, for instance :

- Image classification with ResNet50 model
- Text to image with Stable Diffusion
- Medical image segmentation with 3D-unet
- Question Answering with BERT
- Text summarization with GPT-J

In this study, we want to compare different deployment approaches for LLM inference, so, we decided to focus this analysis on the LLM provided by MLPerf and, more precisely, on the BERT model. In this section, we present the architecture of the BERT model, we also present precisely on which tasks we analyze the environmental impacts of the deployment, and finally, we present how does work the GPU parallel algorithm provided by the TensorRT library which optimize the inference process.

Architecture

The Bidirectional Encoder Representation Transformer (BERT) [4] is an LLM architecture that was the new state-of-the-art model on eleven NLP tasks at its release time. BERT's novelty comes from the pretraining method, where classical LLM architectures learn left-to-right token relations by using a next-token prediction pretraining approach; the pretraining of the BERT model consists of two tasks, masked language model training (MLM) and next sentence prediction (NSP):

- MLM: the MLM training consists of allowing the model to learn how token sequences are constructed by randomly masking tokens of the input sequence and training the model to find the masked token.

- NSP: the NSP pre-training consists of allowing the model to learn the relationship between continuous token sequences by giving as input to the model pairs of sequences and training the model to identify if the two sequences are consecutive or not.

This new training method trains the model to capture bidirectional token relations. The architecture of the BERT model correspond to a classical transformer encoder stack, the only difference is the use of a Gaussian Error Linear Unit (GeLU) hidden activation function applied on the hidden layer of the feed forward network of the encoder blocks. The Table 2.2 shows the main parameters of the BERT models that we use in this study.

Parameter	**Value**
hidden_act	gelu
hidden_size	1024
intermediate_size	4096
max_position_embeddings	512
num_attention_heads	16
num_hidden_layers	24
vocab_size	30522

Table 2.2: BERT Model Architecture Parameters

SQUAD dataset

The task on which we test the deployment settings corresponds to the Stanford Question Answering Dataset (SQUADv1.1). This task consists of giving as input to the model a paragraph and a question, and the objective of the BERT model is to find the segment of the paragraph that answers the associated question. The Figure 2.6 shows an example of a paragraph and associated questions. We used a BERT architecture finetuned on the SQUAD task to obtain a testing accuracy of 99%.

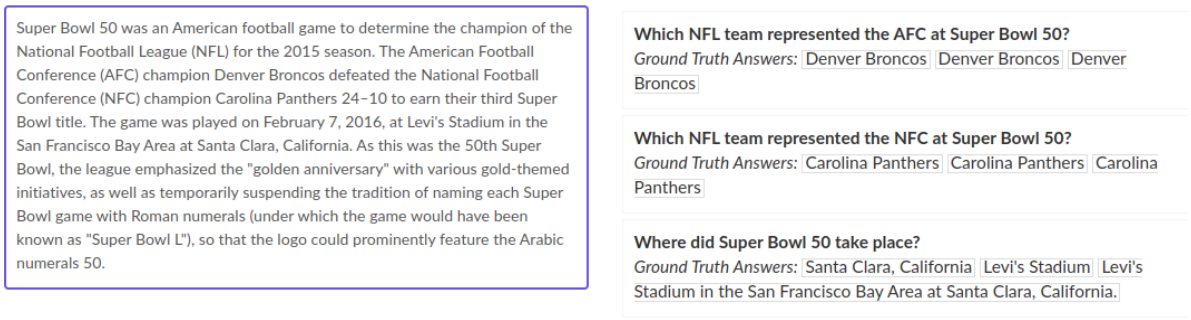

Figure 2.6: Example of the SQUADv1.1 dataset on the subject "Superbowl".

TensorRT parallelization

As we said while describing the differences between large-scale and edge devices, nowadays, GPUs are intensively used for AI training and inference. The common use of these type of devices for these tasks come from the large amount of computing units in these piece of hardware

and their organizations, these characteristics make the GPUs quiet efficient for linear algebra related tasks such as neural network inference (see Section 2.1). The GPU developer are able to program NVIDIA GPU parallel programs by using the CUDA Nvidia framework, which allow them to write CUDA kernels which correspond to a piece of code which will be executed on the NVIDIA GPU. These kernels describe the organization of the computing units required in the parallel program to solve the target tasks. TensorRT is another NVIDIA framework built to optimize neural network inference, this framework provided optimized implementation of CUDA kernels corresponding to commons AI inference function such as Multihead attention layer. The main mains optimization provided by the TensorRT library on the BERT inference is the fusion of the Add and Norm kernel in the layer norm layer of the transformer block, and the optimization of the GeLU hidden activation function of the FFN block [5]. The TensorRT-optimized kernels are used in the MLPerf inference benchmarks used in this study.

[5]For more detail about the optimization of the TensorRT library on BERT see : TensorRT with BERT

— 3 —
Related works

Environmental impact estimations applied to ICT require well-defined methods to provide relevant and interpretable analysis; the life-cycle assessment domain (LCA) provides us a set of tools to quantify the environmental impact associated with each part of the life cycle of an ICT resource. We know how to apply LCA methods for specific computing devices impact analysis from manufacturing impact databases (ADEME), but it is more complicated while considering the life cycle of digital services. More specifically, in this study, we aim to estimate the environmental impact associated with generative AI services. For instance, the data from ChatGPT shows that more than one hundred million of users use the platform every day. These problematics raise multiple important questions: "such as how to consider the hardware product manufacturing impact required by the AI service? Or how can we associate an environmental impact with the use phase of the AI service ?" There are only few studies on the inference part of this question, in the following sections we present the most relevant ones.

3.1 Environmental impacts quantification

We based the environmental impact quantification method on different studies from the state-of-the-art. In this section, we present different macroscopic observations that allow us to capture the global AI environmental impacts trend. We also present the life cycle assessment methodologies on which we based our approach to quantify the environmental impact associated with AI systems inference.

3.1.1 AI impacts trend

It is actually quite hard to find data about the effective GHG emissions related to AI systems. This problem is due to the fact that an important part of the AI applications deployed at a large scale use data centers of private companies to perform their computations (Microsoft Azure, Google Cloud Platform, AWS, etc.). These private companies are not fully transparent in terms of quantification of the GHG emissions associated with AI systems. The works of Berthelot et al. [1] which estimates the environmental impact of the generative AI service "Stable Diffusion", show that, for one year of use, the service produces on the order of 360 tons of carbon equivalent emissions. These observations show that AI systems have a non-negligible environmental impact. In addition, by using macroscopic observations, we can capture the trend of AI

application deployments. By looking at the annual report of Nvidia[1], which is a major provider of accelerators hardware products, we can see that the company drives their hardware to be specialized for AI training and inference. We can also see that the demand for data centers of Nvidia accelerators specialized for AI inference/training significantly increases over the years. We can observe that by looking at the revenues of the company in the data center market in 2024, which have increased up to 217% from 2023. This observation indicates the increase in the number of AI systems deployed at a large scale. The predictions expect that this trend will continue to increase in the coming years.

3.1.2 LCA methodology and MLCA approach

We proposed a quantification methodology to measure the environmental footprint of AI systems inference based on a life cycle assessment (LCA) approach. The LCA approach used to measure the environmental impact of informatics systems provides a quantification of the footprint of every part of the lifecycle of the studied system. We are interested in the inference part of the life cycle of AI systems. The study of Berthelot et al. [1] which estimate the environmental impact of the generative AI system StableDiffusion by using an LCA approach, provide us with a relevant formula to quantify the environmental impacts of the inference process of the AI system for an inference i processed by using the hardware GPU for an execution time t:

$$I_{inference} = C_{i,GPU} \times EGM_g \times PUE + a_{GPU}(t) \times F_{GPU}$$

- $C_{i,GPU}$ represent the electricity consumption of performing the inference i on the hardware e in kwH.
- EGM_g represent the electricity grid mix impact in geographic area g.
- PUE represent the power usage effectiveness of the hardware used.
- $a_{GPU}(t)$ represent the time-based allocation of the hardware e during t time units.
- F_{GPU} represent the manufacture footprint of the hardware GPU in $kg\ CO2\ eq$.

There is a huge variety of environmental impact factors that we could capture through the variables F_{GPU} and EGM_g (water consumption, abiotic depletion potential, primary energy, etc.). We decided to focus this study on the global warming potential factor, which estimates the contribution of the system to the global climate change in kgCO2. We made this choice because of the strong representativity of this metric and because it is the one with the most important amount of data available in the literature. Regarding the manufacturing impact F_{GPU}, we used an approach inspired by the study of Morand et al. [11] about a machine learning life cycle assessment method that considers that the manufacturing footprint is the sum of the footprint of the GPU processing unit and the GPU memory unit. So, the manufacturing impact of the hardware GPU can be defined as follows:

$$F_{GPU} = F_{GPU_{proc}} + F_{GPU_{mem}}$$

[1] https://s201.q4cdn.com/141608511/files/doc_financials/2024/ar/NVIDIA-2024-Annual-Report.pdf

3.2 Previous works limitations

The PhD study made by Jay [7] provides a comparative analysis of the associated environmental impact of AI training between edge devices (Jetson AGX Xavier) and large-scale supercomputers (Apollo). The conclusion was that the edge devices could not be considered as a relevant alternative for AI training due to poor computing performance. We aim to go further by considering the inference process instead of the training.

A relevant study by Luccioni et al. [9], which used an LCA approach to estimate the environmental impact associated with the BLOOM LLM, provides us relevant methods to estimate the associated environmental impact of the inference process. The limitation of this study is that they considered an inference server composed of 16 NVIDIA A100 GPUs, which corresponds to a large-scale deployment. In this study, we aim to apply a similar environmental impact quantification method (for inference) in order to provide a comparative analysis between large-scale and edge devices.

3.3 MLPerf inference benchmark limitations

MLPerf inference benchmark provides a submission method to share performance results of specific inference deployment configurations from edge to large-scale deployment[2]. A lot of companies use this submission module to share their results, such as Nvidia, Cisco, AMD, or Dell. But, as mentioned, the submissions consider only performance-related metrics without including the associated environmental impact of the inference process. This study aims to fill this important missing point of view to allow people who want to design inference servers to take into account these impacts.

[2] https://mlcommons.org/benchmarks/inference-edge/

— 4 —
Methods

4.1 Large-scale to edge devices continuum

We want to differentiate the edge devices and the large-scale devices to compare their environmental footprint regarding the life cycle of an AI system. The first approach was to define a continuum from edge to large-scale for ranking the GPUs of this study by defining that large-scale devices consume more energy and have a better performance than edge devices. But in reality the picture is more complex than that; the scale of a device (edge or large) depends on multiple factors that are not always provided by hardware specifications : portability, security, effective energy consumption, network requirements, etc. In the environmental impact analysis, instead of considering only edge devices and large-scale devices, we considered the different use cases for which the different GPUs are produced; this information is provided by the constructors.

- Large scale use case related to datacenter infrastructure where a high peak performance is required.
- Desktop use case of professional tasks which can run on personal servers/computers.
- Gaming use case where the hardware is specialized for computer graphics.
- Edge device use case where the hardware must be close to the user.

The Figure 4.1 show the differences between the categories of GPUs that we defined.

4.2 Estimating the environmental impact of hardware manufacturing

To apply the LCA approach to estimate the environmental impact associated with LLM inference, we must estimate the environmental impact related to hardware manufacturing. The pieces of hardware considered in this study are the GPUs presented earlier. Knowing that, we can use the method defined in the study of Morand et al. [11] about a machine learning life

Figure 4.1: Graphical representation of the different types of GPUs considered in this study.

cycle assessment method. As such, we can define the manufacturing environmental impact of GPUs as:

$$F_{GPU} = F_{GPU_{proc}} + F_{GPU_{mem}}$$

Where $F_{GPU_{proc}}$ represents the manufacturing environmental impact of GPUs processing chips and $F_{GPU_{mem}}$ the same for the memory chip. In this section, we present the method that we chose to estimate the environmental impact associated to GPU manufacturing, and we explain how do we compute $F_{GPU_{proc}}$ and $F_{GPU_{mem}}$.

GPU processing unit impact

To compute the environmental footprint of the processing chip of a GPU, we based our approach on the study of Pirson et al. [12] which provides us with an important dataset of measured environmental impact of integrated circuits (ICs), which are the key components of processing chip manufacturing, in function of multiple criteria (technological node size, foundry, date, ...). The value associated with the environmental footprint is given in $kgCo2\ eq/cm^2$ of wafer for the GWP metrics and in L/cm^2 for water consumption. With this dataset, for both metrics, the environmental impact of processing chip manufacturing can be defined as follows :

$$F_{GPU_{proc}} = GPU_{die\ area} \times I_{proc\ data}(GPU)$$

Where $I_{proc\ data}(GPU)$ represents the impact factor data retrieved in the dataset from the information of the GPU to evaluate (technological node, release date, etc.). And $GPU_{die\ area}$ represents the total size of the processing chip in cm^2. The dataset from [12] contains different data that come from different sources (literature, foundry, etc.). This diversity in terms of sources comes with the problem that for a given GPU, we may have multiple values coming from different sources. To handle this problem, we decided to consider confidence intervals instead of single values for the environmental impact of GPUs' processing units with multiple associated values from the dataset.

GPU Memory

To compute the environmental impact of the memory chip of GPUs, the first approach was to use a similar method as the one used for the processing chip. We start by computing the die area

Memory Type	Bit Density (Gb/mm^2)	Chip Name	GWP (gCO2/GB)
HBM2e	0.274	DDR4	280
HBM2	0.274	DDR4	280
GDDR5	0.137	DDR3	360
GDDR5X	0.137	DDR3	360
GDDR6	0.274	DDR4	360
LPDDR4X	0.152	LPDDR4	290
LPDDR5	0.152	LPDDR5	290

Table 4.1: Memory types characteristics

of the memory chip as the amount of memory of the GPU in Gb, $GPU_{mem\ size}$ divided by the memory density of the memory architecture in Gb/cm^2, $GPU_{mem\ density}$. After that, we multiply the die area obtained by the same impact factor $I_{proc\ data}(GPU)$ used for the processing chip. So, the environmental footprint of the manufacturing of the GPU memory was modeled as follows:

$$F_{GPU_{mem}} = \frac{GPU_{mem\ size}}{GPU_{mem\ density}} \times I_{proc\ data}(GPU)$$

Considering the impact factor of the manufacturing of the memory chip as the same as the one of the processing chip is a strong assumption that we think is not true in reality due to significant differences in the manufacturing process between the two types of ICs. The memory used for GPU is a Dynamic Random Access Memory (DRAM). The study by TechInsights [8] which had as its goal to model carbon emissions associated with wafer fabs, provides data for computing the GWP associated with the DRAM-based memory manufacturing process in $kgCO2/GB$ for different types of DRAM memory chips. Based on these data, the second approach is to define the environmental impact of GPU memory unit manufacturing as follows:

$$F_{GPU_{mem}} = GPU_{mem\ size} \times I_{mem\ data}(GPU)$$

where $I_{mem\ data}(GPU)$ represents the data provided by the TechInsights study [8]. The Table 4.1 shows the data used to compute the environmental impact of the memory depending on the memory type.

4.3 Performance estimation of Large scale and Edge devices : StarCoder inference

The work done by Coignion et al [3] study the electricity consumption and the CO2 emission associated with the deployment of inference servers computing the inferences of coding assistant architectures StarCoder (LLM). To better understand the differences in terms of performance and energy consumption between large-scale and edge devices for inference and to apply the environmental impact quantification methodology, we define a method to highlight such differences by estimating the runtime, the energy consumption, and the environmental impact of edge and large-scale inference with the StarCoder model.

4.3.1 StarCoder Architecture & Complexity

The StarCoder model is based on a transformer decoder-only (see Section 2.1.2) architecture, which consists of a stack of N transformer decoder blocks. The inference process transformer decoder block is based on 4 main computing blocks : 1 multi-head attention block, 1 feed-forward network block, and 2 Add & Norm blocks. Our goal here is to define an accurate lower bound on the number of floating-point operations ($FLOPs$) performed by the model for 1 inference process. To do that, we will consider only the computations done by the multi-head attention and the feed-forward blocks, which are the most expensive ones. In our approach to computing a lower bound of the number of $FLOPs$ for 1 inference process, we assume that the computations of each attention head are made in parallel, meaning that we estimate the amount of $FLOPs$ required for the multi-head attention process as the number of $FLOPs$ required for the computation of a single attention head. By looking at the matrix formula of the attention process, we can see that the complexity of the attention is of order $O(l^2 \times d)$, where l is the length of the input sequence in tokens and d is the hidden size of the model (size of Q, K, V vectors). We will use this complexity approximation as the lower bound for the number of $FLOPs$ for the attention process. Classically, the attention process by transformer decoder architecture is a masked multi-head attention where we apply a mask on the scaled dot product to ensure that while computing the relation between tokens, the model does not take into account the token after the current word. This masking process is not considered in the estimation due to its low computational cost.
The second important block of the transformer-decoder architecture is the feedforward block, which corresponds to 1 hidden layer neural network of d input neurons, f hidden neurons, and d output neurons, with an $ReLU$ activation for the hidden layer. The forward pass of the FFN is done independently for each token of the sequence. In the estimation, we consider that the computation of the FFN is parallelized at the token level. So we can see that the complexity of the FFN can be approximated by $O(2 \times f \times d)$, so we consider it as a lower bound of the FFN this complexity. We can see on Table 4.2 the architecture of the StarCoder model where

Table 4.2: StarCoder Hyperparameters

Hyperparameter	**StarCoder**
Hidden size	6144
Intermediate size	24576
Max. position embeddings	8192
Num. of attention heads	48
Num. of hidden layers	40
Attention	Multi-query
Num. of parameters	≈15.5B

we can retrieve the different variables that we defined:

- Hidden size = d
- Intermediate size = f
- Max position embeddings = l

- Num of hidden layer = N

The notion of multi-query is similar to the notion of multi-head attention; the difference is that instead of having multiple projection matrices for Q, K, and V for each head, we have different Q vectors for each head, but we use the same K, V vectors for all heads. Because we previously considered that the attention computation of each head is parallelized, this difference doesn't change the lower bound. So we can now compute the lower bound of $FLOPs$ of the architecture :

$$FLOPs_{attention} = l^2 \times d \approx 4.1232 \times 10^{11}$$

$$FLOPs_{FFN} = 2 \times f \times d \approx 3.0199 \times 10^{8}$$

$$FLOPs_{Total} = N \times (FLOPs_{attention} + FLOPs_{FFN}) \approx 1.6505 \times 10^{13}$$

The final results of the StarCoder analysis are shown in the Results section

4.3.2 StarCoder environmental impact estimation method

In order to apply the LCA approach to estimate the associated environmental impact of the inference process of StarCoder model and to implement the comparative analysis, we will consider the following assumptions :

- Both settings are powered by the electricity production of the same geographical region $g =$"France metropolitaine", where $EGM_g = 79.1\ gCO2\ eq/kwH$ from Base empreinte[1]
- Both settings have a classical data center $PUE = 1.5$.
- $a_{GPU}(t)$ can be considered as $t/lifetime_{GPU}$, where $lifetime_{GPU}$ corresponds to the lifetime of the hardware under study (t represents the runtime duration in hours). This information is really hard to estimate because of the lack of information reported regarding the end-of-life part of the life cycle of GPUs. For this study we will consider $lifetime_{GPU} = 7\ years$ for all the GPUs of the study; this assumption would be relaxed in further works. So :
- To compute $C_{i,GPU}$, we will consider, for this first analysis, that during the inference, the instantaneous power provided to the GPUs corresponds to their TDP. So we can compute $C_{i,GPU}$ as :

$$C_{i,GPU} = t \times TDP_{GPU}$$

The results of this case study are presented in Section 5.2.

4.4 Experimental methodology

A strong assumption that we want to relax in this study is that the inference power consumption is fixed during all the runtime and is equal to the TDP of the GPU under test. This assumption hides a huge part of the study because it prevents us from considering the different power consumption profiles among the sample of GPUs of the comparative analysis. These different

[1] `https://base-empreinte.ademe.fr/donnees/jeu-donnees/c8d09c07-100d-321d-b603-76aa866c004a/0/false/null`

profiles depend on GPU drivers and GPU architectures, and we think that the consideration of these profiles can significantly affect the environmental impact observations. To relax this assumption, we decided to build an experimental methodology based on the MLPerf benchmark implementation.

4.4.1 Installation methods

To allow us to perform an experimental analysis over the set of hardware considered in this study, we had to install the required modules for deploying the MLPerf inference environment, from the operating system booted on the device to the set of Python packages required by the benchmark. We defined three different installation methods corresponding to three deployment contexts.

Grid'5000 nodes For the Grid'5000 (G5k) nodes, which correspond to non-edge GPUs, we used the G5k tool kadeploy to deploy a `debian-11-big` as an operating system on the node. After that, for both versions of MLPerf inference, the environment required to run the benchmark is built through an nvidia-docker container, so we installed nvidia-docker by using the G5k script `g5k-setup-nvidia-docker`.

Jetson AGX Xavier For the AGX Xavier device, compatible with MLPerf inference v1.1, we had to flash the device with the compatible version of L4T (r32.6.1). To do that, we used the SDK Manager Nvidia tool, which allows us to flash a specific version of L4T on Jetson devices. The MLPerf inference benchmark v1.1 on Jetson devices doesn't use a Docker image. For deploying the environment, the environment runs locally, so we had to define an installation script that installs on the device all the required modules, such as TensorRT, ONNX, or PyTorch.

Jetson AGX Orin For the AGX Orin device compatible with MLPerf inference v4.1, we also had to flash the device with the right version of L4T (r36.3.0). This time, to facilitate the flashing process and to improve reproducibility, we use a tool from Grenoble named Kameleon. This tool allows us to totally deploy an informatics system from a recipe, which takes the form of a YAML file that describes the different installation steps. In our context, we used it to build a Docker image containing all the dependencies required to flash a given version of L4T; this image is then used to flash the device. The MLPerf inference benchmark v4.1 environment runs through an NVIDIA Docker container. After flashing the device, we install NVIDIA Docker and can build the benchmark image.

When the environment is set, the model and data of the benchmark are downloaded through web requests. The models use an ONNX format, which is a format optimized for graph storage. After building the binary files through the Makefile of the benchmark, we are now able to run the experiments on the different devices.

4.4.2 Energy measurements

We aim to build an experimental methodology to perform a comparative analysis regarding the GWP associated with LLM inference, we have to define an efficient method to measure

precisely the amount of energy consumed by the GPU during the inference process. We based the energy measurements method on software-level tools, which allows us to monitor the instantaneous power provided to the GPUs. For the Jetson devices, we used *tegrastats*, a module provided with L4T that allows us to track the power consumption of every component of the devkits. For the other GPUs of the study, we used the Nvidia Management Library (NVML) through the usage of $nvidia - smi$, which allows us to monitor the power consumption of Nvidia GPUs. We defined three implementations of the measurement tool corresponding to 3 different deployment contexts.

AGX Xavier With the AGX Xavier, the MLPerf environment runs locally, so the implementation of the energy measure was defined as a bash script which starts the *tegrastats* module with an update frequency of $50ms$ and which formats the outputs into a CSV file. At the end of the execution, a bash script is called to stop the measure. The *tegrastats* software use power monitoring chips to track the power consumption of the different modules of the board, the Figure 4.2 show one of the monitoring chips used by *tegrastats*[2].

AGX Orin With the AGX Orin device, we used the same scripts as for the AGX Xavier. The difference is that we didn't succeed in connecting the nvidia-docker container of the MLPerf inference environment to the hardware bus that contains the hardware component required for *tegrastats*, so we can't call *tegrastats* in the container. So we defined an SSH access from the nvidia-docker container to the local environment of the device to call the start and stop bash scripts.

Grid'5000 GPUs For the GPUs available through the Grid'5000 platform (non-edge), the NVML library is directly accessible in the nvidia-docker container. So we defined the start and stop scripts by using $nvidia - smi$ with an update frequency of $10ms$.

To transform the instantaneous power consumption as a function of time curve into effective energy consumption, we used the trapezoidal rule to compute the area under the curve to obtain the energy consumed in *joule* that we can easily convert into kwH. To improve the confidence of the analysis, we performed each power measurement ten times to analyze the variability of the observations.

4.4.3 MLPerf inference Compatibility

There is a huge variety of versions of the MLPerf inference benchmark, and each version has its own TensorRT and L4T compatible version. With the sample of GPUs (see Table 2.1) considered in this study, the different devices come from different generations. These differences introduce compatibility issues with the different versions of MLPerf inference. For example, MLPerf inference version 1.1 uses TensorRT version 8.0.1.6 which is not compatible with the most recent GPU micro-architecture, such as Hooper (H100) or Ada Lovelace (L40)[3]. Another example is that the MLPerf inference benchmark version 4.1 on Jetson devices requires L4T

[2] `https://www.onsemi.com/products/power-management/dc-dc-power-conversion/controllers/ncp81239`

[3] Github of TensorRT v 8.0:`https://github.com/NVIDIA/TensorRT/blob/release/8.0/plugin/bertQKVToContextPlugin/qkvToContextInt8InterleavedPlugin.cpp` (line 65)

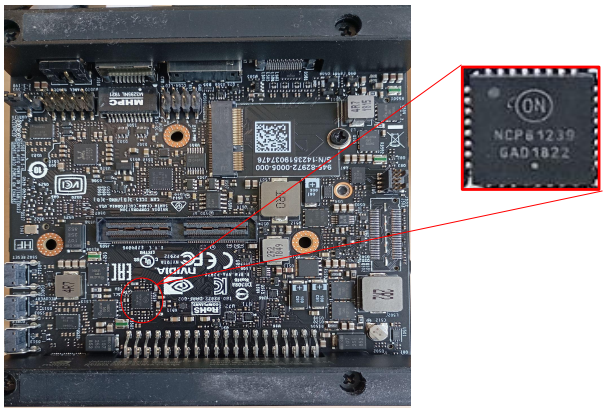

Figure 4.2: Snapshot of the Jetson AGX Xavier board (from under the board). The red circle shows the NCP81239 chip (ON semiconductor) used by *tegrastats* to track the power consumption of the board.

release 36.3.0, which is supported on AGX Orin devices but not on AGX Xavier devkits. Due to these compatibility issues, we decided to use two different versions of the benchmark to perform the comparative analysis :

- MLPerf inference v1.1 requires L4T release 32.6.1 with Jetpack 4.6 and TensorRT 8.0.1.6. We used it to perform the experimental methodology on AGX Xavier, A100, A40, T4, 2080Ti, Quadro RTX 6000 and Quadro RTX 8000 devices.
- MLPerf inference v4.1 requires L4T release 36.3.0 with Jetpack 6.0 and TensorRT 8.6.2. We used it to perform the experimental methodology on AGX Orin, A100, H100 and L40S devices.

There are important differences between the versions of the modules required by the benchmarks; these differences can introduce variances in terms of energy consumption profiles between the two versions. For this reason, we tested the A100 GPU on both benchmarks to see if the energy consumption profile of the GPU is different in function of the benchmark and if a cross-version comparison is possible.

4.4.4 Experimental settings

As we mentioned in the presentation of the MLPerf inference benchmark, there is a huge variety of parameters that we can tune to simulate a specific use case of ML models, from the scenario in use (single-stream, multi-stream, offline, or server) to the number of queries and samples to process during the test. With the set of GPUs considered in this study, we are not

able to simulate data center inference servers classically used to handle the computations required by large-scale LLMs, which involve several large-scale GPUs. In this study we decided to consider, for each experiment, a single GPU to perform the inferences required by the test. This choice allows us to highlight the different power consumption profiles of each GPU, further works would add the dimension of the number of device in use. In that context, we decided to focus the experiments on the single-stream scenario, which represents an edge usage of an ML model. We made this choice because we think that, with the server scenario, it's the most related to LLM utilization. We didn't consider the server scenario in this study because of the hardware constraint that we mentioned before, but further work would compare the single-stream and the server scenario in terms of environmental impacts to improve the consistency of the analysis.
Regarding the number of queries to process during the experiments, we want to simulate a single user's edge utilization of an LLM, so we defined 5 user profiles depending on the utilization rate of the model. We model the user profiles by using five different values to define the number of queries to process during the test: 16, 32, 64, 128, and 256. With the single-stream scenario, the queries are sent consecutively to the SUT to reduce the idle time (see Figure 2.4). We know that this is not realistic because the user of an LLM does not send their query consecutively; further work would modify this scenario to define a time interval between queries sent to simulate a realistic edge usage of an LLM based on human-computer interaction studies. For these experiments, we used the BERT model that we described previously because it is the only LLM available on both considered versions of the benchmark. As we mentioned before, the most popular generative LLM models nowadays use an autoregressive inference process for inference, while the BERT model performs the inference through a single forward pass. This corresponds to a strong limitation of our work; further work would perform this analysis with autoregressive generation models. The data collected during each experiment relates to power consumption metrics with the measurement method (see Section 4.4.2) and to performance-based metrics with the maximum time required by the SUT to process a query during the experiment[4].

[4]In the future, we denote this time as "maximum query latency."

5
Results

5.1 Grid'5000 hardware manufacturing impact

To observe the impact of the GPU's production use case on the manufacturing impact, we decided to represent the Grid'5000 GPUs by their manufacturing impact in function of their use case by using the formulas defined in the previous section.
The Figure 5.1 percentage of contribution to the global manufacturing footprint of the processing chip and the memory chip for each GPU for both memory footprint quantification methods.

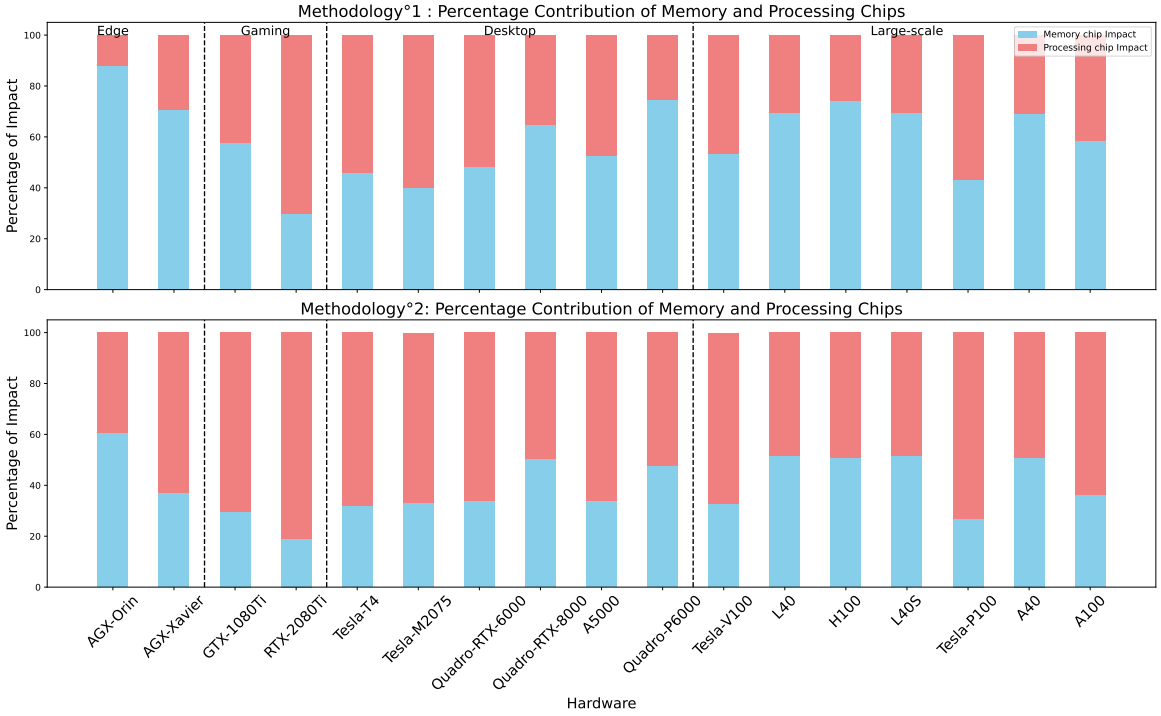

Figure 5.1: Bar plot representing the average percentage of contribution to the global manufacturing footprint of the processing and the memory chip for both memory impact quantification method.

We can see that there is a huge difference between the methodology that uses the memory density to compute the area of the memory chip and the methodology that uses the type of memory chip in use to compute the environmental impact of the memory chips. The first methodology provides us with a high percentage of contribution of the memory, on average around 50%. While the second approach provides a less important contribution to the global manufacturing footprint, on average around 20%. We can see that there is GPU variability while looking at the difference of results obtained with both methodologies. The GPU where the difference of results is the most impressive is the only Edge device of the analysis where we have a difference around 40% between both methodologies. These results can be explained by the memory model used by the Nvidia Jetson AGX Xavier GPU, which is *LPDDR4X* which uses the *LPDDR4* memory chips, which have a weak memory density that can explain the high results obtained with the first methodology; but the *LPDDR4* memory chips are the chips that have one of the lowest $kgCO2/GB$ of the DRAM-based memory manufacturing GWP dataset. The memory model characteristics of the considered GPUs are shown in Table 4.1. These observations can explain the difference in results that we observed for the edge device. The Figure 5.2 show the estimations of all the GPUs of the dataset.

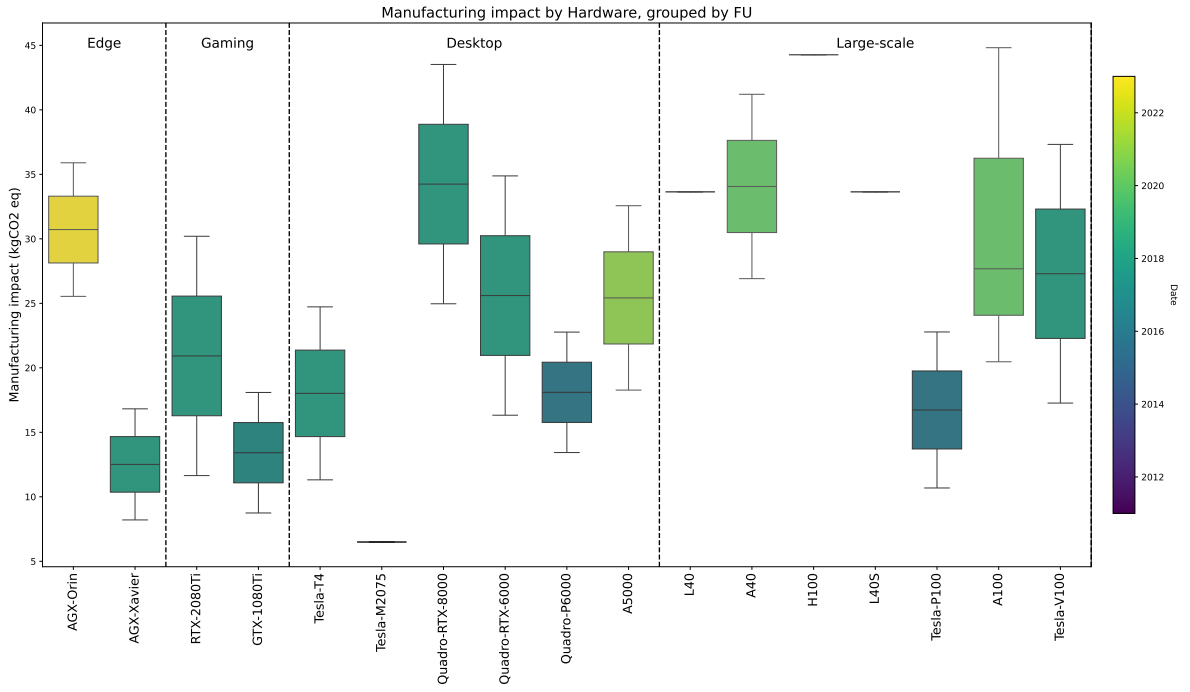

Figure 5.2: Box plot comparing the environmental impact of Grid'5000 GPUs manufacturing, illustrated by the Global Warming Potential (GWP) factor in kg CO2 equivalent as a function of their production use cases (Edge, Gaming, Desktop and Large scale).

We can observe that there is a significant impact of the use case of the GPUs on their manufacturing footprint, by comparing the GWP of the Edge device (Nvidia Jetson AGX Xavier) with the GWP of the large-scale devices which is in average 2 times higher than the one of the

Jetson device. These results have been obtained by using the second approach for computing the environmental impact of the GPU memory chips defined previously.

5.2 StarCoder case study results

In this section we present the results of the comparative analysis between large scale and edge devices for the environmental impact associated to the StarCoder model inference.

5.2.1 Roofline model

In order to compare the deployment models that we want to study (edge and large scale) from a computational point of view and to estimate lower bounds on the execution time required for 1 inference in both deployment contexts, we apply the roofline model. This model upper bounds the amount of FLOPS of a parallel program depending on its operational intensity, where :

$$\text{Operational Intensity} = \frac{\text{Total FLOPs}}{\text{Total Bytes Accessed}}$$

where:
- Total FLOPs is the total number of floating-point operations performed by the algorithm.
- Total Bytes Accessed is the total number of bytes read from or written to memory.

The bounds of the Roofline model are defined by the peak memory bandwidth of the hardware used, which represents the maximum number of bytes that can be transferred between the processing units and the global memory in 1 second, and by the peak performance of the hardware in FLOPS.
Regarding the total number of bytes read from or written to memory, we assume that the model is stored in global memory with a precision of $FP32$, so we consider the number of bytes transferred as : $15.5 \times 10^9 \times 4 \approx 57.74GB$.
Regarding the hardware specification[1] required for applying the roofline, we consider the edge device and large-scale settings as follows :

Edge computing deployment: We consider that the inference process is performed by using 1 NVIDIA Jetson AGX Xavier 32GB with a memory bandwidth capacity of 136.5 GB/s, a peak computing performance of 1410 $GFLOPS$ in $FP32$ precision, and a Thermal Design Power (TDP) of 30 W. By applying the hardware manufacturing environmental impact quantification, we can estimate F_{Xavier} to be around 17 $kgCO2\ eq$.

Large-scale computing deployment: We consider that the inference process is performed by using 1 NVIDIA A100 40G with a memory bandwidth capacity of 1.56 TB/s, a peak computing performance of 19.49 $TFLOPS$ in $FP32$ precision, and a TDP of 400 W. By applying the hardware manufacturing environmental impact quantification, we can estimate F_{A100} to be around 31 $kgCO2\ eq$.
By using the lower bound of $FLOPs$ previously computed, we can compute a lower bound on the operational intensity of the inference process as 285.764 $FLOPs/Byte$. We can see in Fig-

[1]Hardware specifications come from https://www.techpowerup.com/

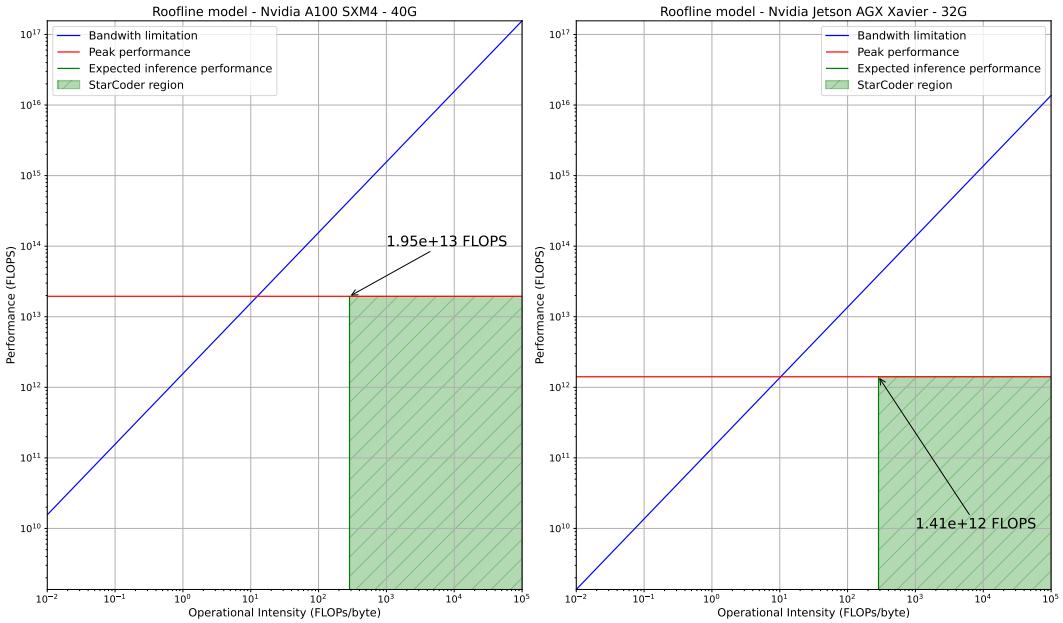

Figure 5.3: Graphs of the roofline model for large-scale deployment (left) and edge device deployment (right).

ure 5.3 that, by using the lower bound of operational intensity previously defined, the amount of $FLOPS$ of the StarCoder inference process is bound by the peak performance of the hardware used in both settings. In addition, we can compute lower bounds of the execution time of the inference in both settings by considering that the inference process completes 100% of the peak performance of the hardware in both settings (reaches the red line in Figure 5.3):

$$Execution\ time_{Edge} = \frac{FLOPs}{FLOPS_{Edge}} \approx 12\ s$$

$$Execution\ time_{Large\ scale} = \frac{FLOPs}{FLOPS_{Large\ scale}} \approx 1\ s$$

5.2.2 StarCoder environmental impact estimation

With the previous approach, we defined a lower bound of the execution time of the inference process; we based the approximation of the environmental impacts on these estimations combined with the hardware specifications of both deployment contexts. By applying the assumptions defined in Section 4.3.2, we can easily compute the environmental impacts of one inference for both settings as :

- Allocation variables:

$$a_{Xavier}(t) \approx 5.4 \times 10^{-8}$$

$$a_{A100}(t) \approx 5 \times 10^{-9}$$

- Energy consumption:

$$C_{i,Xavier} = \frac{12}{3600} \times 30 \approx 0.1\ WH$$

$$C_{i,A100} = \frac{1}{3600} \times 400 \approx 0.1\ WH$$

- Final environmental impacts estimations :

$$I_{Xavier} = C_{i,Xavier} \times EGM_g \times PUE + a_{Xavier} \times F_{Xavier} \approx 1.3 \times 10^{-2}\ gCO2\ eq$$

$$I_{A100} = C_{i,A100} \times EGM_g \times PUE + a_{A100} \times F_{A100} \approx 1.2 \times 10^{-2}\ gCO2\ eq$$

This simple application of the environmental impact methodology highlights two trade-off that we must consider in this study :

- Runtime duration - Runtime power This trade-off relates to the computation of the $C_{i,GPU}$ variable in kilowatt-hours, which is computed as the product between the runtime duration and runtime power consumption assumed to be fixed and equal to the TDP of the device for this analysis. We can observe this trade-off by looking at the results observed. For both GPUs, the energy consumption of the inference is estimated at around $0.1\ WH$, while the two inferences have different characteristics. This observation is due to the fact that the AGX Xavier counterbalances its low instantaneous power consumption with a high runtime duration with, while, the A100 counterbalances it's low runtime duration with a high power consumption during inference.

- Allocation time - Manufacturing impact We have a situation here where the right part of the final formula (after the $+$) is different for the two settings of the evaluation (1.55×10^{-4} for the A100 and 9.18×10^{-4} for the AGX Xavier). This difference is due to the fact that the low manufacturing impact of the AGX Xavier is counterbalanced by its high allocation time, while it's the total opposite for the A100, but the difference between the two GPUs in terms of manufacturing impact is less important than the one in terms of allocation time ($\frac{F_{A100}}{F_{Xavier}} < \frac{a_{Xavier}}{a_{A100}}$).

5.3 Experimental results

In this section, we present the results observed by applying the experimental methodology. We used the trapezoidal rule to transform the power consumption profiles into energy consumption measures, and we finally applied the environmental impact quantification method for inference to compute GWP values from the observations. We also include in the analysis performance-based indicators such as the maximum latency observed in each experiment.

5.3.1 Power consumption profiles and PUE

In this section, we present the results of the power measurement method applied to the different GPUs of the analysis for the different configurations that we defined in the previous sections. We executed ten times the power measure for each setting. As you can see in Figure 5.4 for the A100 Nvidia GPU, we observed a strong stability in the results of the ten measures for all the

experimental conditions. For the cross-version comparison, as you can see on the same figure where we executed on the A100 GPU the same configurations for each considered version of the benchmark, we observed non-negligible differences between the measurement results of the two version of the benchmark in terms of runtime duration and an of power consumption. The execution of the benchmark with the version *v*4.1 is significantly faster than with *v*1.1, with a difference of two seconds with the setup with 256 queries. In terms of power consumption, the maximum instantaneous power observed is around 200 W for the *v*4.1 and around 175 for the *v*1.1. These differences are due to software improvements applied to the *v*4.1 from *v*1.1, such as TensorRT kernel optimizations between its versions 8.0.1.6. and 8.6.2., which can improve the GPU utilization rate. From these observations, we decided to not apply a cross-version analysis to avoid introducing biases in the conclusions of this study.

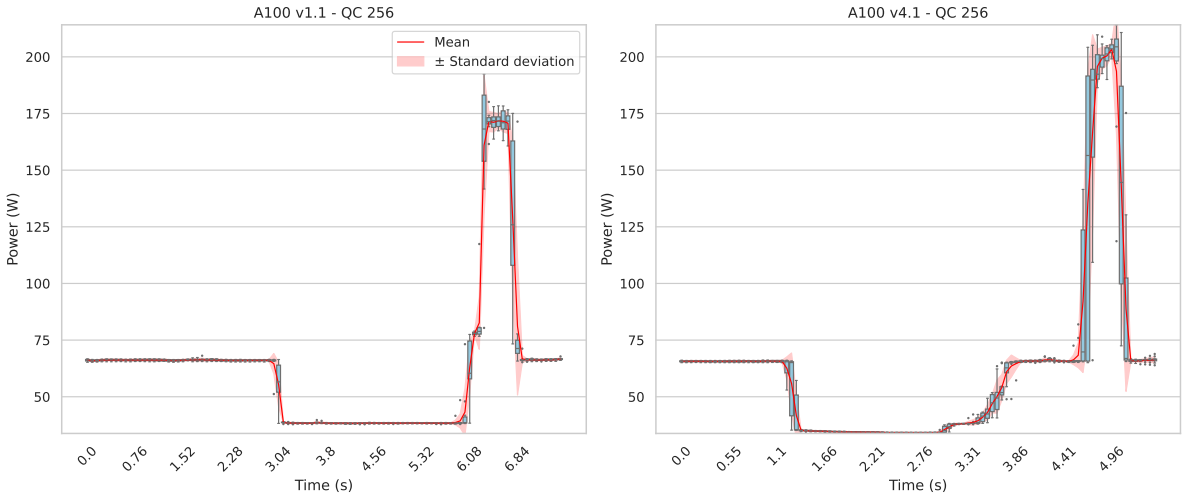

Figure 5.4: The power consumption profile of the A100 GPU obtained by applying an interpolation of 100 time points over 10 executions of the benchmark configuration. The left graph represents the v1.1 and the right graph represents the v4.1. The configuration of both versions is 256 queries to process for the SUT. The box represents the results observed of the interpolation, the red line represents the mean, and the red region represents the standard deviation.

The Figure 5.5 shows the different power consumption profiles of the GPUs of the analysis for the different experimental conditions and for both versions of the benchmark. As we expected, for both versions, for a query count from 16 to 128, by increasing the number of queries to process, we increase the peak instantaneous power measured. This is due to the fact that by increasing the number of queries, we increase the workload of the GPUs, so it requires more power to perform computations. For 256 queries, we can observe that the two version of the benchmark don't have the same behavior, in the *v*4.1 the power continue to increase but for the *v*1.1, the peak power stays the same, but it lasts longer as we can see for the Quadro RTX GPUs. These differences can be due to the fact that the GPUs used for the v1.1 are used with a maximum utilization rate of 128 queries, while the GPUs of the version *v*4.1 are underused due to their more recent release date and their better performances. We can also observe really different power consumption profiles among the GPUs of the analysis. The Jetson devices consume significantly less power compared to the other GPUs, but we can see that this low power consumption is counterbalanced with a longer runtime for the AGX Xavier with 256 queries.

Another thing that we can observe on the Figures 5.4 and 5.5 is that we can easily observe the two main phases of the execution of the benchmark : the copy of the BERT engine from the disk to the GPU DRAM memory (the part of the power graphs with low variations, from 0 to 6 seconds for the A100 with v1.1 for 256 queries) and the compute-intensive phase (the peak power region). We can see that the loading phase duration is not the same for all the SUTs of the methodology; these differences are mainly due to two factors: the I/O performance of the memory of the GPU and the I/O performance of the disk storage. We decided to conserve these differences while constructing the analysis to include the loading of the model in the environmental impacts quantification methodology. Further work would find a way to normalize the impact of external devices on the model loading time. You can find in the Appendix chapter all power consumption profiles measured in this study.

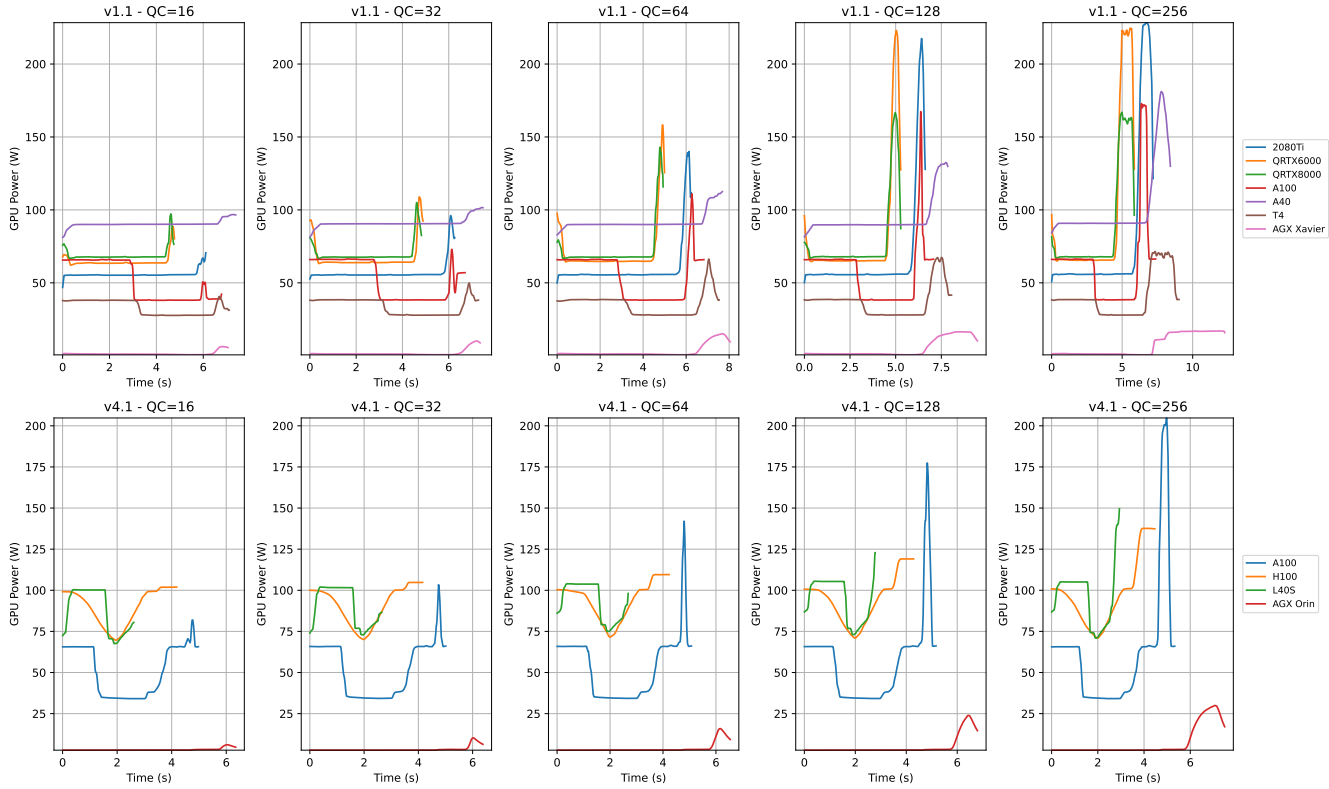

Figure 5.5: Power consumption profile of the GPUs of this study in the different configurations of this study. The two rows represent the versions of the benchmark, while the columns represent the number of queries from 16 to 256.

Now, we have the power curve, but before applying the environmental impact quantification methodology, we must define the Power Usage Effectiveness (PUE) of the GPUs of the analysis. The PUE is a high-performance computing-related metric that can be computed as follows :

$$\text{PUE} = \frac{\text{Total Data Center Energy}}{\text{IT Equipment Energy}}$$

Where IT equipment energy corresponds to the energy provided to the computing device required to feed cores and memory, while the total data center energy corresponds to the total

amount of energy required to feed the infrastructures required to maintain the safety of the computing devices, such as external cooling systems or light for people coming to maintain computing resources. For the large-scale devices, we decided to use the average PUE of the Grid'5000 platform in 2009: 1.5[2]. We made this choice because the use case that we want to model through this study is an edge utilization of an LLM. The large-scale GPUs, by definition, can't be placed close to the user because these devices require a strongly controlled environment such as data centers. On the other hand, the edge, gaming, and desktop GPUs of this study can be used as desktop devices that do not require external infrastructure to maintain the device. For this reason, we decided to consider a PUE of 1 for such devices.

5.3.2 Inference time analysis

One of our goals in this study is to observe if edge devices can be used for LLM inferences realistically. A relevant way to achieve that goal is to analyze the time required for the inference process, which is one of the most important characteristics of the deployment from the user's point of view. The MLPerf inference benchmark, in the LoadGen module, implements a method to measure the maximum latency observed in an experiment. The Figure 5.6 shows the maximum latency results observed for all the GPUs in this study for all the experiments performed on them. We can easily see that the edge devices are significantly slower than the

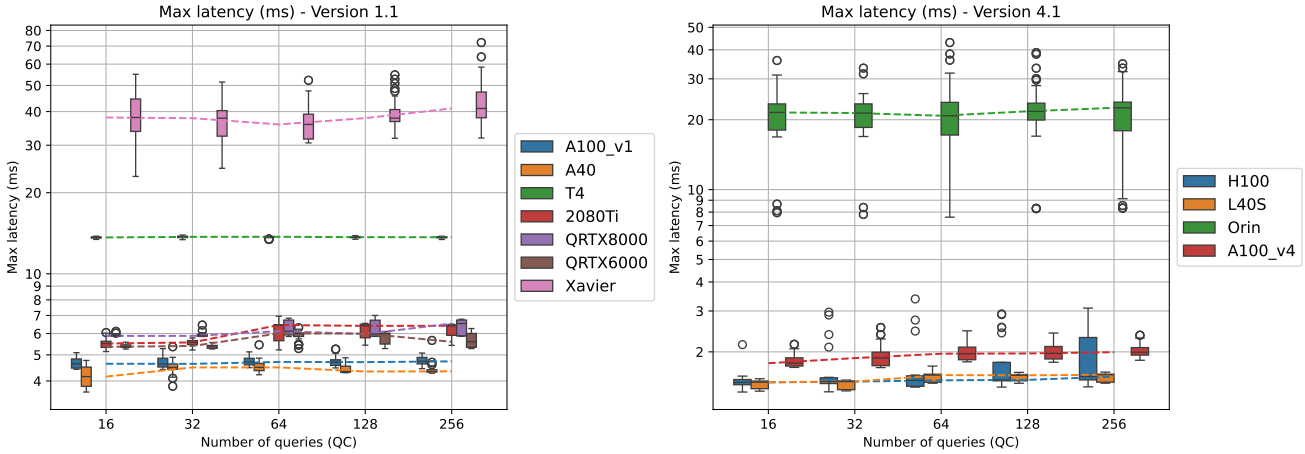

Figure 5.6: Boxplots, which represent the maximum query latency in milliseconds observed in every experiment in function of the number of queries to process during the test. The experiment was made on the sample of GPUs considered in this study for both versions of the MLPerf inference benchmark considered in this study (v1.1 on the left and v4.1 on the right). The y-axes are represented with a logarithmic scale.

other devices, around 40 ms for the AGX Xavier against 5 ms for the other GPUs of the version *v*1.1. For the version *v*4.1, the AGX Orin is around 20 ms against 1.8 ms for the other GPUs. We can see that the differences are on the order of 30-20 ms, which is quasi-imperceptible from a human point of view. But it's important to recall that the model considered in this study is BERT, which is not an autoregressive generative model, so the final output is generated in one

[2]https://intranet.grid5000.fr/stats/indicators

forward pass. If we consider that one inference of the BERT model requires the same amount of computation as an autoregressive generative model, and we consider that we want to generate a token sequence of 128 tokens, we could expect a maximum latency in the order of 2.5 seconds for the AGX Orin versus 230.4 ms for the other GPUs of the *v*4.1. The differences are nonnegligible and strongly perceptible by humans. Further work would include an experimental analysis of autoregressive generative models such as GPT models to provide real latency data instead of estimations and would include a human-computer interaction study to evaluate which inference latency time values are still acceptable from the user's point of view.

5.3.3 Environmental impacts analysis

By applying the same environmental impact quantification methodology as the one we applied to the StarCoder model, we are able to estimate the GWP associated with the inference processes of the experimental conditions. The only difference with the previous analysis is that, instead of considering a fixed power consumption during the inference process, we use the power consumption profiles collected during the experiments. We transform these power curves in watts into energy in kilowatt-hours by using the trapezoidal rule, which consists of computing the area under the curve by summing the area associated with all the consecutive points of the temporal power series. For all the experimental conditions of this study, we used the average of the ten repetitions of the measure to define the considered power profiles. We can

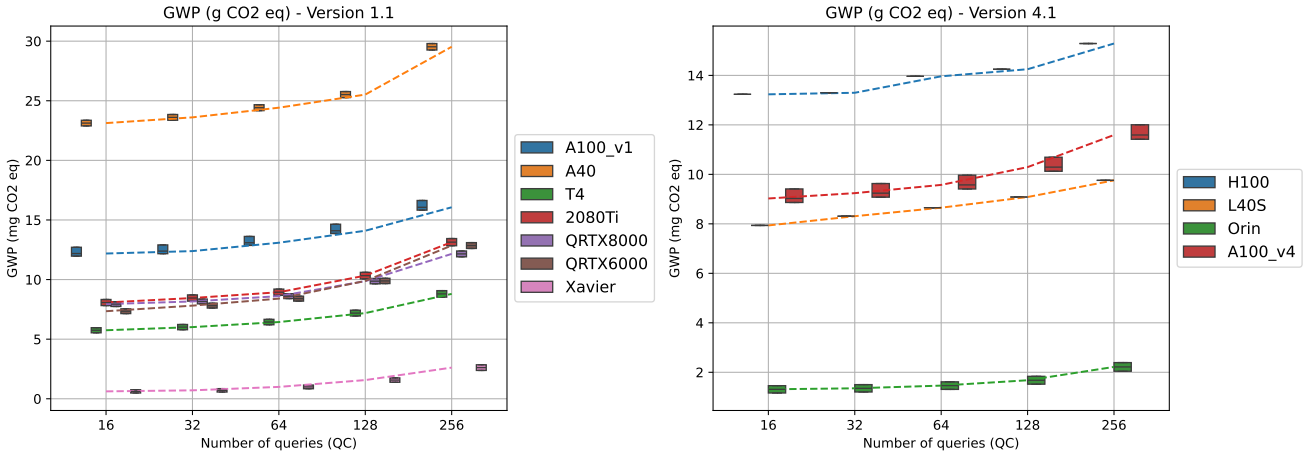

Figure 5.7: Boxplots, which represent the estimation of GWP in grams of CO2 emission equivalent observed in every experiment as a function of the number of queries to process during the test. The experiment was made on the sample of GPUs considered in this study for both versions of the MLPerf inference benchmark considered in this study (v1.1 on the left and v4.1 on the right).

see in Figure 5.7 the results of the environmental impact quantification methodology applied to the experimental conditions. The box plots represent the confidence interval of the GPUs' manufacturing impact that we presented in the previous sections. We can see that the boxes observed are tight; this is due to the allocation time measure over a 7-year lifetime, which allows us to reduce the uncertainty intervals. As we expected, for all the GPUs of the analysis, increasing the number of queries to process during the test also increases the GWP estimated. This

observation is due to the observation made in the power profiles section about the increase in the peak power consumption. We can also observe that the edge devices produce significantly less CO2 than the other GPUs in the study. The Table 5.1 shows the detail of the results of the methodology averaged over all the observations. We can see that the edge devices emit at least $6\times$ less CO2 than the other GPUs of the analysis from the estimation methodology in all the experimental conditions. To illustrate this differences in a realistic application, let's take a large scale LLM platform such as ChatGPT which receive, each day, queries from 122.5 million of unique users [3]. For the analysis, we will consider that each user sends 16 queries, the model in use is BERT, and each user uses an independent computing device. With this setting definition and with the estimation results, we can easily compare the GWP associated with a deployment on H100 GPUs and a deployment on AGX Orin GPUs :

$$GWP_{Orin} = 122.5M \times 1.31\ mg \approx 160\ kg\ CO2eq$$

$$GWP_{H100} = 122.5M \times 13.24\ mg \approx 1.6\ T\ CO2eq$$

We can observe a difference of factor 10 between the two deployment configurations. These results highlight the potential of edge devices to be used to handle inference of LLM. This observation is due to the low power consumption of edge devices during the inference process, as illustrated in Figure 5.5.

5.3.4 Results analysis

The observations of the previous sections shows that the Edge devices have a lower environmental impact associated with a slower inference process compared to large-scale GPUs. We observe in Figure 5.7 that the slope of the GWP curve of the Edge GPUs is significantly less important compared to the other devices evaluated. This observation means that the amount of CO2 emitted for the inference process increases while increasing the number of inferences to perform, but this increase is slower for the Edge GPUs. From this point, we can expect that by increasing the query counts considered, the differences in terms of environmental impact between edge and large-scale devices would also increase.
This conclusion encourages us to consider experimental settings with more inference to process to model an edge utilization of LLM by a group of users, such as a working group in a company or a student class with parallel accesses to the computing resources. But these parallel accesses to the GPUs would introduce important differences in terms of maximum query latency compared to the current sequential inference process. An interesting perspective would be to include multiple edge inference deployments considering multiple numbers of users using the system to analyze the impact of this dependent variable (number of users) on the associated environmental impact. These experimental settings can be modeled by building a new scenario in the benchmark between single-stream and server.

[3] ChatGPT statisitics

QC	Version	GPU	GWP (mg CO2 eq)	QC	Version	GPU	GWP (mg CO2 eq)
16	**v1.1**	2080Ti	8.07	32	**v1.1**	2080Ti	8.45
		QRTX6000	7.35			QRTX6000	7.81
		QRTX8000	7.94			QRTX8000	8.16
		T4	5.75			T4	6.01
		A40	23.13			A40	23.61
		AGX Xavier	**0.62**			**AGX Xavier**	**0.77**
		A100	12.29			A100	12.49
	v4.1	A100	9.10		**v4.1**	A100	9.32
		H100	13.24			H100	13.30
		L40S	7.94			L40S	8.31
		AGX Orin	**1.31**			**AGX Orin**	**1.35**
64	**v1.1**	2080Ti	8.94	128	**v1.1**	2080Ti	10.33
		QRTX6000	8.40			QRTX6000	9.89
		QRTX8000	8.61			QRTX8000	9.87
		T4	6.44			T4	7.19
		A40	24.42			A40	25.52
		AGX Xavier	**1.0**			**AGX Xavier**	**1.57**
		A100	13.19			A100	14.20
	v4.1	A100	9.65		**v4.1**	A100	10.37
		H100	13.96			H100	14.25
		L40S	8.65			L40S	9.08
		AGX Orin	**1.46**			**AGX Orin**	**1.68**
256	**v1.1**	2080Ti	13.14				
		QRTX6000	12.85				
		QRTX8000	12.15				
		T4	8.80				
		A40	29.53				
		AGX Xavier	**2.61**				
		A100	16.17				
	v4.1	A100	11.67				
		H100	15.29				
		L40S	9.76				
		AGX Orin	**2.22**				

Table 5.1: Global warming potential (mg CO_2 eq) for different QC values by GPU and version, with the GPU having the minimal impact per condition and version highlighted in **bold**.

— 6 —
Conclusion

This study had the objective to compare the environmental impacts associated to the energy hungry inference process of LLM by using different devices to handle the deployment of the model. We focused on the comparison between edge and large scale devices available through both Grid5000 platform and desktop edge devices. We used existing studies to construct a global warming potential dataset (GWP), which models the manufacturing impact of the GPUs considered in this study. We proposed an experimental methodology to measure the effective energy consumption associated to the inference process of the LLM for the different GPUs. We used the common MLPerf inference benchmark to simulate a single user edge utilization of the BERT LLM for question answering tasks. We transformed the effective energy consumption data in kilowatt-hour, collected by using this methodology, into GWP measure in $CO2$ equivalent emissions by using the french electricity grid mix. Our main result was to be able to provide a good estimate of the environmental impact of the different deployment settings by summing the inference impact and the manufacturing impact pondered by an allocation factor. The results show significative differences in terms of GWP between edge devices and the other GPUs considered in this study. We showed that the inference process on H100 large-scale GPU emit around 10 times more CO2 than using Jetson AGX Orin edge GPU. More precisely, from an experimental point of view, the deployment settings designed in this study corresponds to an edge utilization of an LLM, in other word, the single-stream scenario (see Section 2.3.1). But our experimental methodology is also able to evaluate other types of inference deployments, such as inference server, where multiple users can access concurrently a single computing device. It would be interesting to produce hybrid scenarios by combining different multiple scenarios in a single deployment (e.g., mix between single stream and server).

We also analyzed the query latency of the deployments. This analysis shows that the deployment on edge devices is significantly slower than a large scale deployment, with around 20 *ms* for the AGX Orin against around 1.8 *ms* for the H100. Potentially, such differences are not perceptible by most users. The latency perspective must be investigated further to know which latency is required to satisfy chatbot interactions by, for instance, looking at human computer interaction literature.

This study opens the relevant question of the Jevons paradox, it considers that improving the efficiency of a technology can increase its global use. It usually counterbalances the energy benefits. Most LLM users minimize the number of requests. Even those who voluntarily optimize their utilization of LLM. For instance, users who don't use LLM for cost reasons could start to use it if the associated impact decreased.

An important latency source in LLM powered application is the networks' communications that we did not consider in this study, we can expect that data exchanges with edge devices would be less impactful than with large scale devices, further work would include this network's communication in the environmental impact quantification and in the computation of the latency. We could also improve the power measurement method to perform a physical measures directly on devices instead of using software-level measures.

— A —
Appendix

In this section, we present the additional figures that present the distribution of the power consumption profiles for each GPU of this experimental study. We perform each experiment ten times to analyze the distribution of the power measured for each condition. This analysis shows that the power measure tracks the power consumption of the GPU with low variability. The boxes represent the distribution of the power measured for each repetition of the experimental conditions.

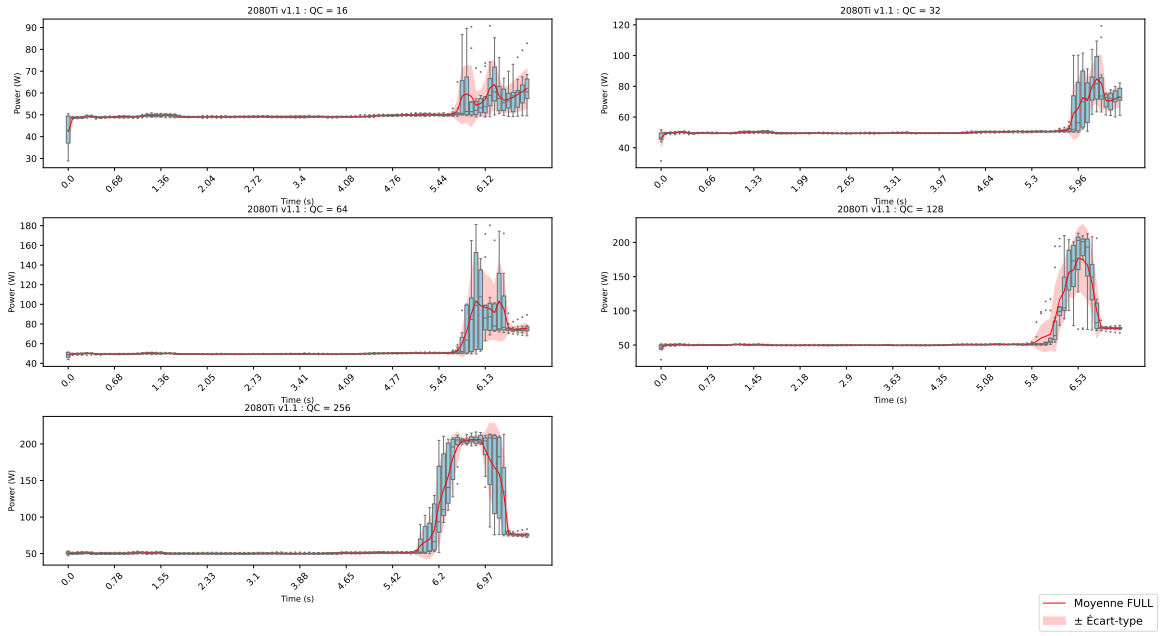

Figure A.1: Power profiles of the 2080Ti GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v1.1.

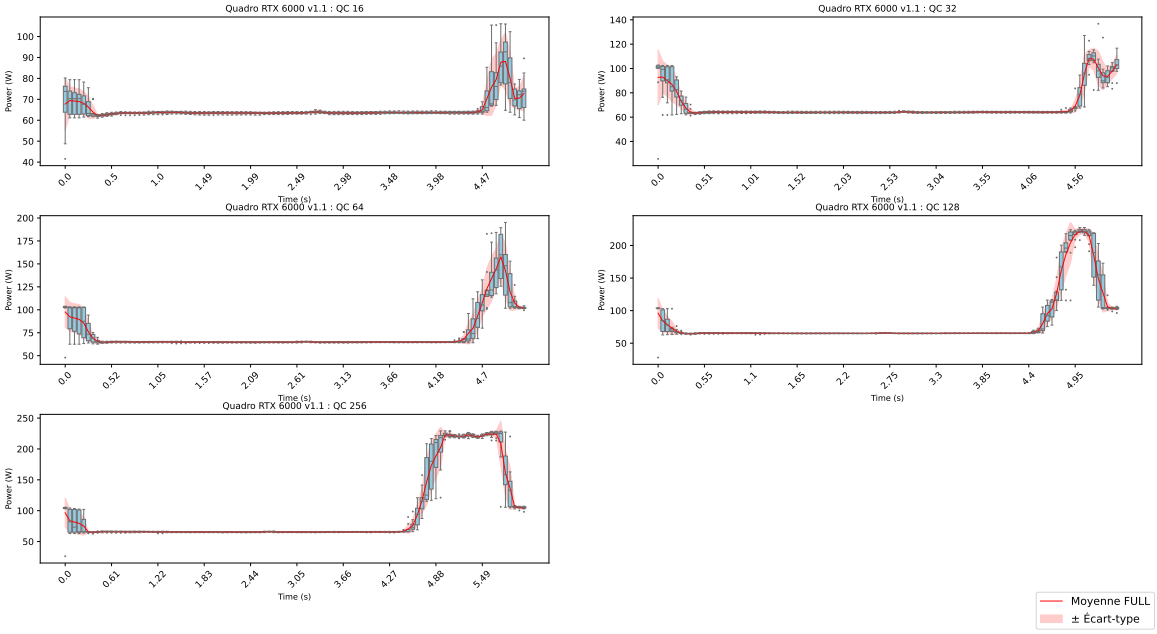

Figure A.2: Power profiles of the Quadro RTX 6000 GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v1.1.

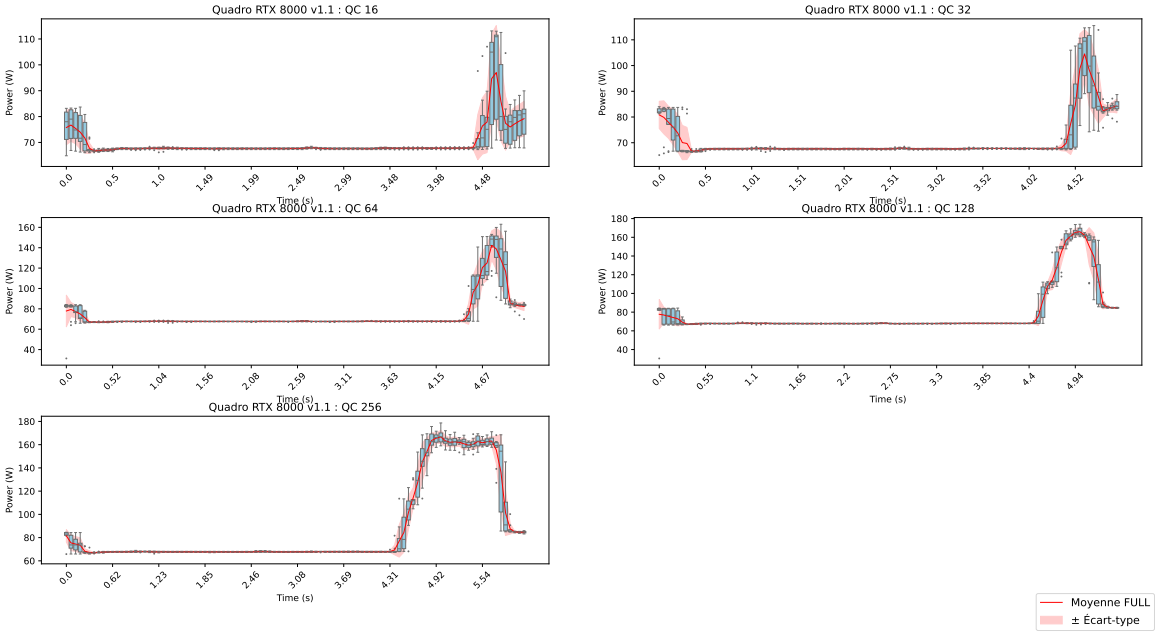

Figure A.3: Power profiles of the Quadro RTX 8000 GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v1.1.

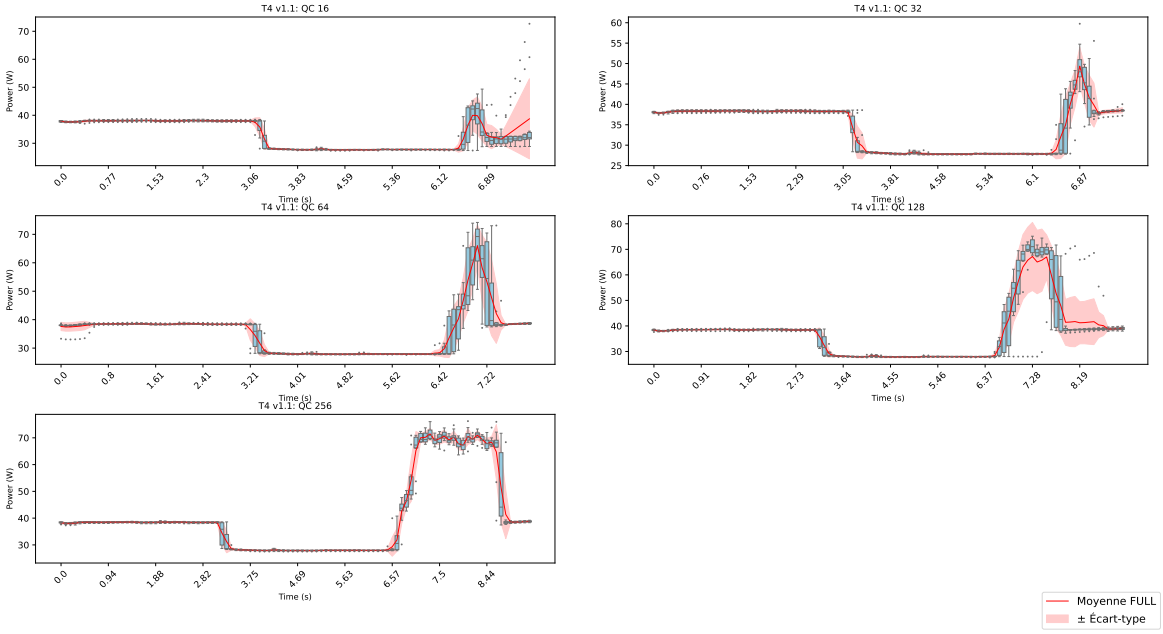

Figure A.4: Power profiles of the T4 GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v1.1.

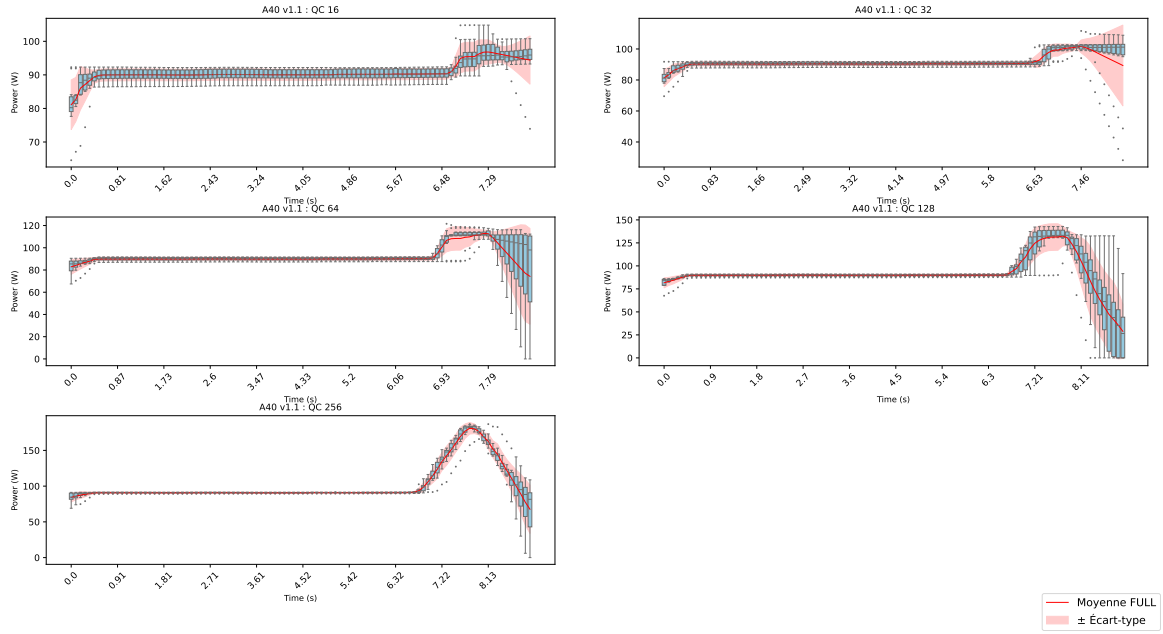

Figure A.5: Power profiles of the A40 GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v1.1.

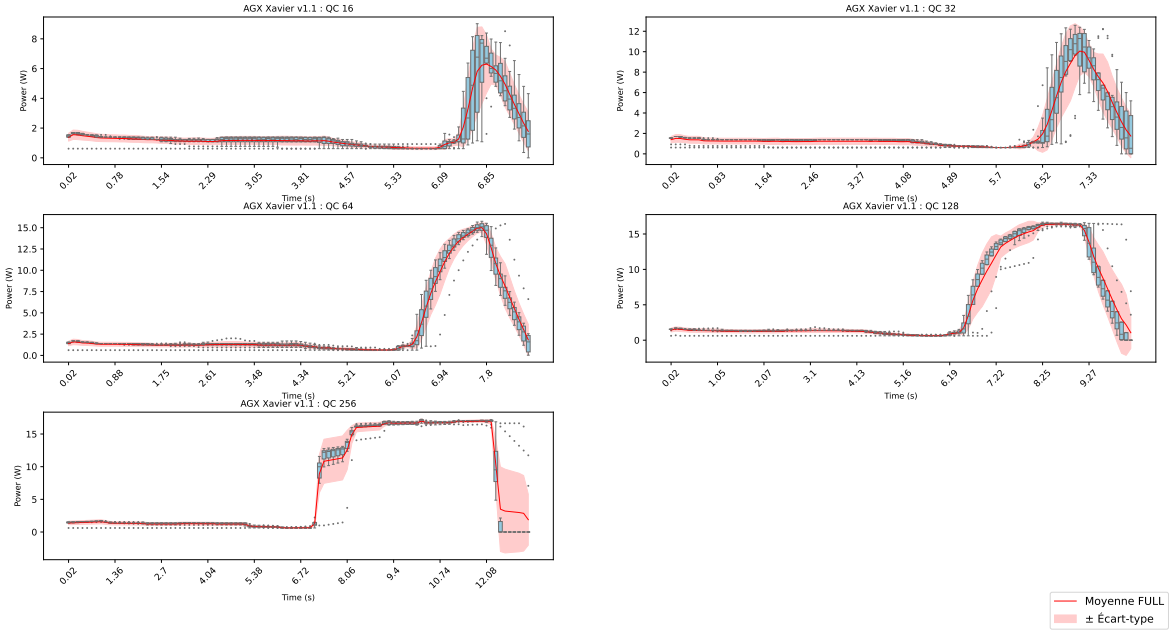

Figure A.6: Power profiles of the AGX Xavier GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v1.1.

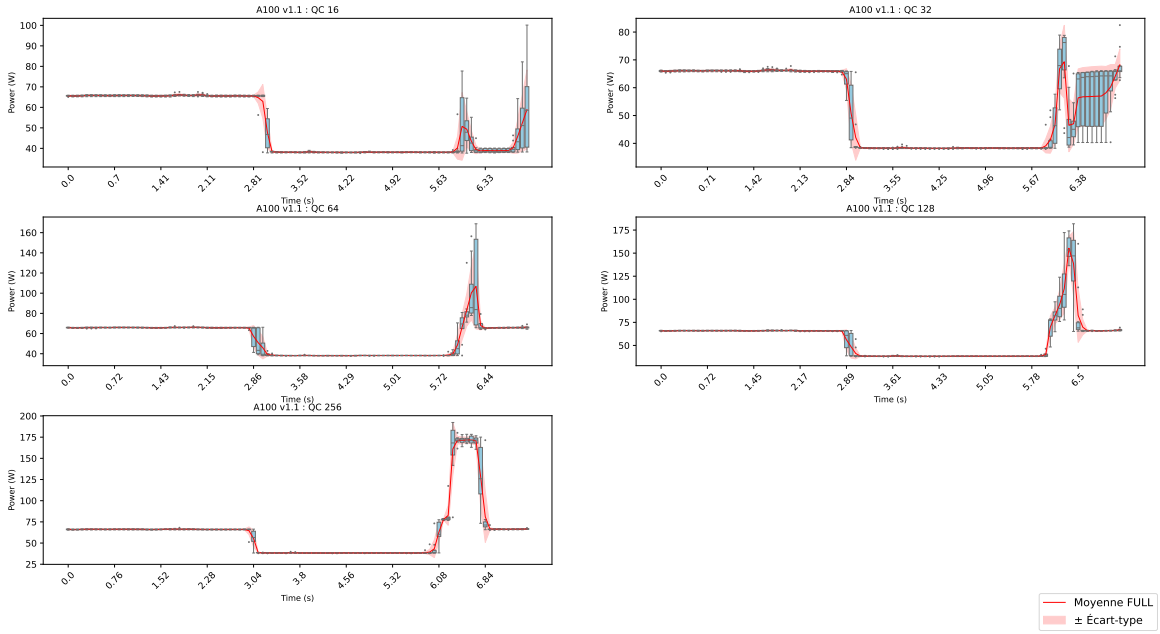

Figure A.7: Power profiles of the A100 GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v1.1.

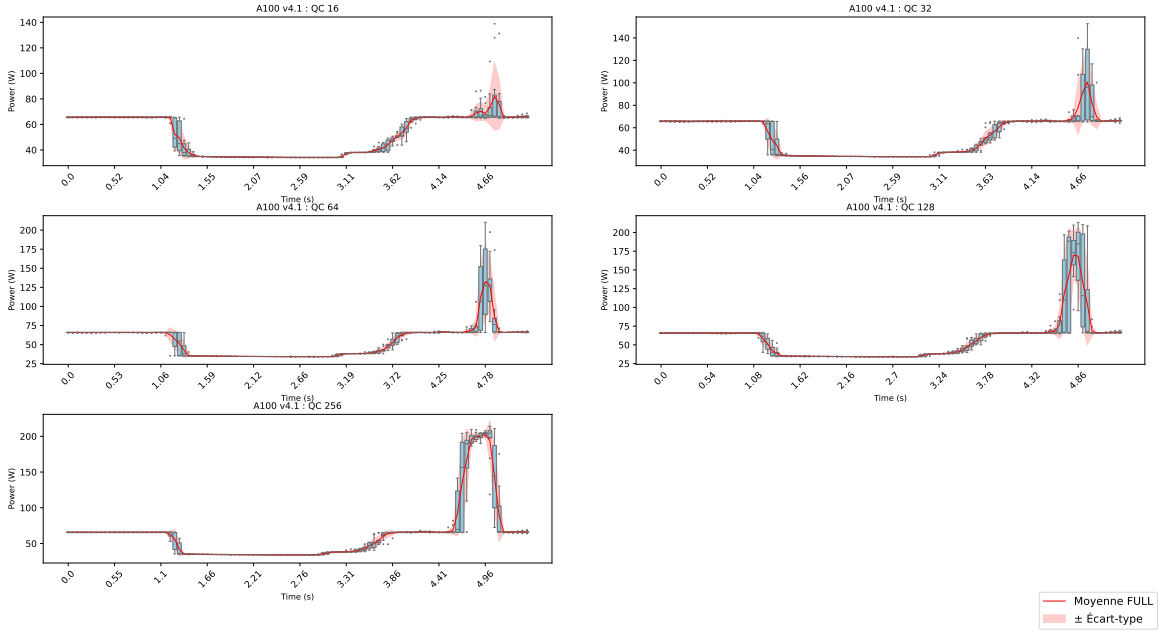

Figure A.8: Power profiles of the A100 GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v4.1.

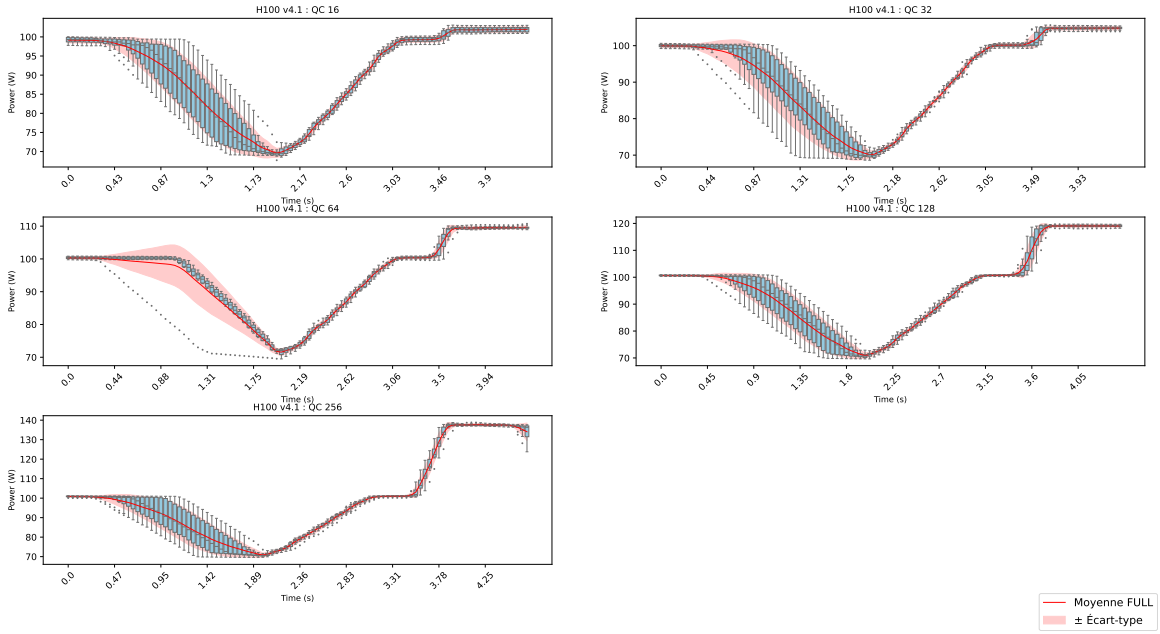

Figure A.9: Power profiles of the H100 GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v4.1.

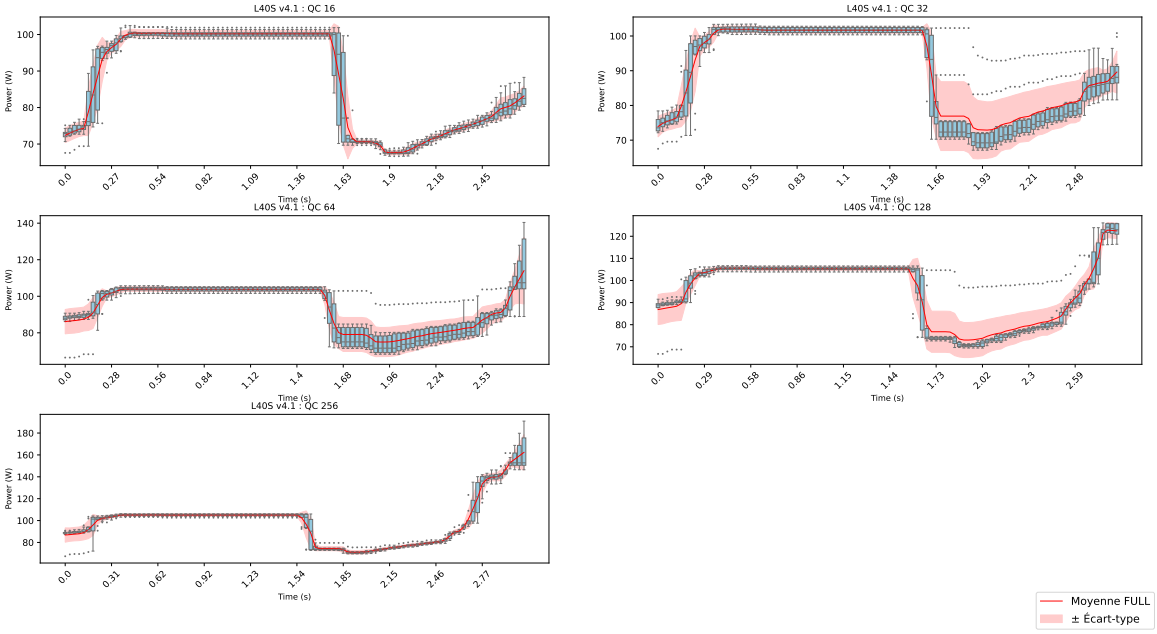

Figure A.10: Power profiles of the L40S GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v4.1.

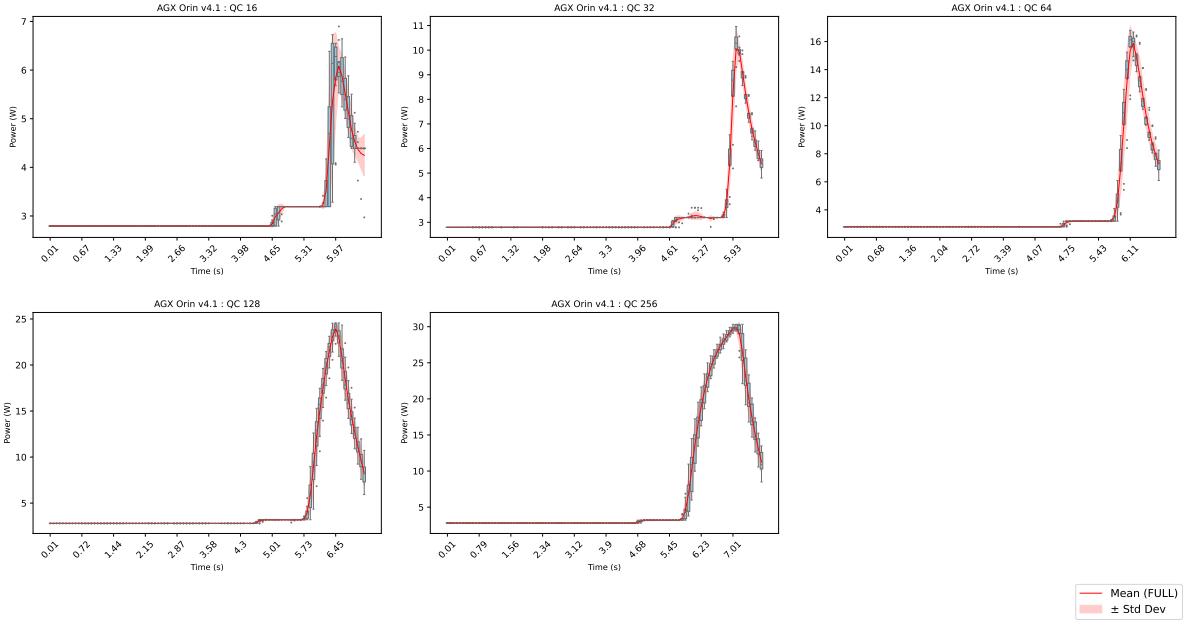

Figure A.11: Power profiles of the AGX Orin GPU observed over the 10 experiment performed for each experimental conditions (QC) with MLPerf inference v4.1.

Bibliography

[1] Adrien Berthelot, Eddy Caron, Mathilde Jay, and Laurent Lefèvre. Estimating the environmental impact of generative-ai services using an lca-based methodology. *Procedia CIRP*, 122:707–712, 2024.

[2] Katherine Calvin, Dipak Dasgupta, Gerhard Krinner, Aditi Mukherji, Peter W Thorne, Christopher Trisos, José Romero, Paulina Aldunce, Ko Barret, Gabriel Blanco, et al. Ipcc, 2023: Climate change 2023: Synthesis report, summary for policymakers. 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. *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.*, pages 1–34, 2023.

[3] Tristan Coignion, Clément Quinton, and Romain Rouvoy. Green my llm: Studying the key factors affecting the energy consumption of code assistants. *arXiv preprint arXiv:2411.11892*, 2024.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[5] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S Blair, and Adrian Friday. The real climate and transformative impact of ict: A critique of estimates, trends, and regulations. *Patterns*, 2(9):100340, 2021.

[6] Our World in Data. Climate change and flying: what share of global co2 emissions come from aviation?, 2020.

[7] Mathilde Jay. *A Versatile Methodology for Assessing the Electricity Consumption and Environmental Footprint of Machine Learning Training: from Supercomputers to Edge Devices*. PhD thesis, Université Grenoble Alpes, 2024.

[8] Scotten W Jones. Modeling 300mm wafer fab carbon emissions. In *2023 International Electron Devices Meeting (IEDM)*, pages 1–4. IEEE, 2023.

[9] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253):1–15, 2023.

[10] Clément Morand, Anne-Laure Ligozat, and Aurélie Névéol. How green can ai be? a study of trends in machine learning environmental impacts. *arXiv preprint arXiv:2412.17376*, 2024.

[11] Clément Morand, Anne-Laure Ligozat, and Aurélie Névéol. Mlca: a tool for machine learning life cycle assessment. In *2024 10th International Conference on ICT for Sustainability (ICT4S)*, pages 227–238. IEEE, 2024.

[12] Thibault Pirson, Thibault P Delhaye, Alex G Pip, Grégoire Le Brun, Jean-Pierre Raskin, and David Bol. The environmental footprint of ic production: Review, analysis, and lessons from historical trends. *IEEE Transactions on Semiconductor Manufacturing*, 36(1):56–67, 2022.

[13] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459. IEEE, 2020.

[14] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.