Estimating the environmental impact of inference deployment[*]

Djoser SIMEU

October 2024

Introduction

With the huge improvements of AI systems applied on Natural Language processing tasks (NLP) that we observed in the 3 last years, we observed also that this new AI model which learn how to understand Natural human language begin to be deployed at a large scale (ChatGPT, Gemini, DeepSeek ...). These deployments don't came without a significant environmental cost, the works of Berthelot.A et al. [BCJL24] which estimate the environmental impact of the generative AI service "StableDiffusion", shown that, for one year of use, the service produce 360 tons of carbon equivalent emission and have an impact on metal scarcity equivalent to the production of 5659 smartphones. This observation coupled with the growth of deployment of such system is alarming in our context of environmental challenge. In addition, a study of Morand.C et al [MLN24a] shown that the environmental impact of the hardware used for the training and the deployments of large AI systems have continuously increased from 2013 to 2023. In that wondering context, the goal of this study is to evaluate the difference in terms of environmental impact between : large-scale deployments and edge devices deployments. This study is focused on the inference deployment of AI models, due to their huge importance in the global environmental impact of AI systems during their complete life cycles, the importance of the inference deployments is also well presented in the "StableDiffusion" study [BCJL24]. This paper start by presenting the differences between edge devices and large scale deployments model for inference. After that, we defined the method used to combine Life-Cycle Assessment (LCA) based environmental impact with performance metrics by following a Quality of Experience (QoE) approach. Our analysis end by a case study of an open-source coding assistant LLM "StarCoder" which allow us to test our comparison methodology and to draw a conclusion on the observed results.

Large scale VS Edge device's inference deployments

AI systems inference deployment involves high-performance hardware such as Graphical processing units (GPU) or Tensor Processing Units (TPU) which provide a strong parallelization of suitable

[*]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.

tasks such as matrix/tensor operations. But how we presented in the previous section, the environmental impact of such pieces of hardware grows linearly with their performances over years. This grows is due to the increase of the die area[1] or to the miniaturization of the Integrated Circuits (IC) or to increase of the amount of energy required by the hardware. These high-performance hardware products with a huge environmental impact are denoted as ”large-scale devices” in our study. In contrast to that, ”edge devices” are energy-efficient computing resources with a low power consumption, such as NVIDIA Jetson hardware products. You may notice that there is no strong demarcation between edge and large-scale devices.

Quality of Experience comparison method

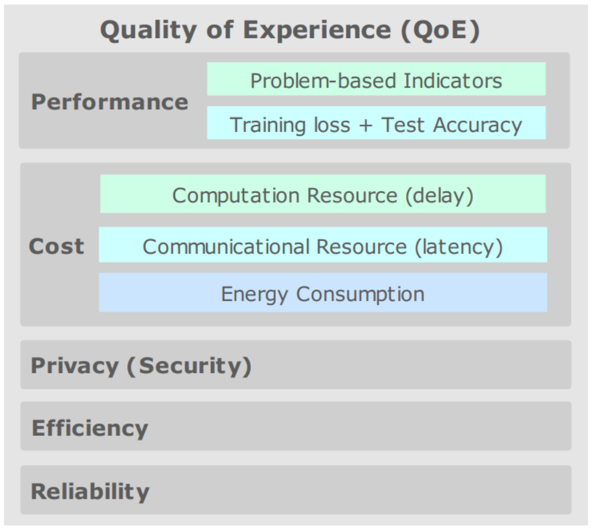
Quality of Experience (QoE)
Performance
Problem-based Indicators
Training loss + Test Accuracy
Cost
Computation Resource (delay)
Communicational Resource (latency)
Energy Consumption
Privacy (Security)
Efficiency
Reliability

Figure 1: Presentation of the Quality of Experience (QoE) approach, where the evaluation of an AI application is defined by a combination of metrics [DZF+20].

In Figure 1 we show the evaluation method presented in the survey [DZF+20] on the edge intelligence. This approach allow the evaluation methodology to consider multi-criteria: Performance, Cost, Privacy (Security), Efficiency and Reliability. Applying to our study, this approach can be combined to the LCA methodology to consider at the same time the environmental impact and the performance of the application. The environmental impacts can be measure by using the same criteria used in [BCJL24] which provide an accurate and a relevant way to quantify the environmental impact of an informatics system:

- Abiotic Depletion Potential (ADP): which estimate the amount of mineral resources extracted to allow the system to run (kgSb = metal scarcity).
- Global Warming Potential (GWP): estimate the contribution of the system to the global climate change (kgCO2).
- Primary Energy (PE): estimate the global energy footprint of the system (MJ).

Regarding the performance of the model, we can evaluate it by using two principal metrics :

[1]The die area correspond to the surface of the processing chip

- Latency: the time requires performing an inference process of the model following the deployment context used (sec).
- Problem based indicators : which measure the quality of the answer of the model regarding a specific task. In our context, the tasks considered in our study are related to NLP's domain (WER).

The study of Berthelot.A et al. [BCJL24] provide to us a relevant formula to quantify the environmental impacts of the inference process of AI system, for an inference i processed by using the hardware e for an execution time t:

$$I_{inference} = Ci, e \times EGM_g \times PUE + a_e(t) \times F_e$$

where $EGM_g, F_e \in \{ADP, GWP, PE\}$ and :

- Ci, e represent the electricity consumption of performing the inference i on the hardware e.
- EGM_g represent the electricity grid mix impact in geographic area g.
- PUE represent the power usage effectiveness of the hardware used.
- $a_e(t)$ represent the time-based allocation of the hardware e during t time units.
- F_e represent the global footprint of the hardware e based on its life cycle. The study of Morand.C et al [MLN24b] about a machine learning life cycle assessment method provide to us a relevant way to estimate the impact of GPU based on the die area of their processors and the memory size:

  $$F_e = e_{die\ area} \times I_{die\ area} + e_{memory\ size} \times I_{memory} + I_\epsilon$$

  Where $I \in \{ADP, GWP, PE\}$

With this evaluation methodology, we can define the research question associated to this study as : In which context, edge deployment for inference have a better QoE measure compare to a large scale computing deployment?

Case study : StarCoder inference deployment

The work done by Coignion.T et al [CQR24] study the electricity consumption and the CO^2 emission associated to the deployment of inference servers computing the inferences of coding assistants architectures (LLM). This study provide to us a strongly reproducible methodology, the server used described in the paper are composed of : 4 NVIDIA A100-SXM4-40GB (GPU) and 1 AMD EPYC 7513 (CPU). This setting relate to a large scale computing deployment. The LLMs used in this work are open source model based on the StarCoder architecture [LAZ+23]. This paper allows us to reuse its methodology to perform a comparative analysis between edge device and large scale inference deployment based on the StarCoder coding assistant model.

Estimating the number of floating point operations performed during the inference process

The StarCoder model is based on a transformer decoder-only architecture, which consist of a stack of N transformer decoder block [Vas17].The Figure 2 show a graphical representation of such architectures. As we can see on the graphical representation, the inference process transformer decoder block is based on 4 main computing blocks : 1 Multi-head attention block, 1 Feed Forward

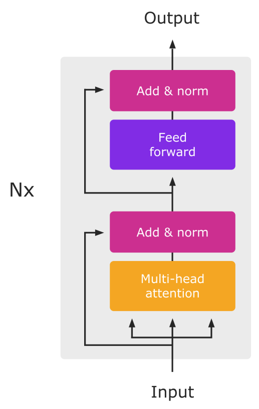

Figure 2: Simple representation which present the architecture of a transformer decoder only model.

network block, and 2 Add & Norm blocks. Our goal here is to define an accurate lower bound on the number of floating point operations ($FLOPs$) perform 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.

Query, Key and Value principle

Before estimating the amount of computation done by a transformer decoder block, 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 return 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 section.

Multi-Head attention block

The Multi-head attention blocks is one of the most important improvement provide by the transformer architectures. Firstly, to obtain the vector Q, K and V from the embedded representation of the input sequence E[2], 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 :

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

where d_{emb} represent the dimension of the embedding representation space of the sequence and d represent hidden size of the model (the dimension of Q, K and V vectors space). To facilitate our interpretation, we consider that $d_{emb} = d$. In that context, The attention mechanism correspond to matching the Q and K vector by using a scaled dot product, using a softmax function on the

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

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 internal relation between tokens of the input sequences. Figure 3 show a graphical representation of this process. The principle of Multi-head attention come from the definition of multiples matrices W_q, W_k and W_v:

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

where h represent the number of heads. With this approach, we have an attention process for each head which allow us to capture multiples levels of relation between the tokens of the sequence, each head is so called an attention head. In our approach to compute a lower bound of the number of $FLOPs$ for 1 inference process, we assume that the computation of each attention head are made in parallel, meaning that we estimate the amount of $FLOPs$ requires for the Multi-head attention process as the number of $FLOPs$ requires to 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 token. 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 relation between tokens, the model doesn't take into account the token after the current word, this masking process is not considered in our estimation due to its low computational cost.

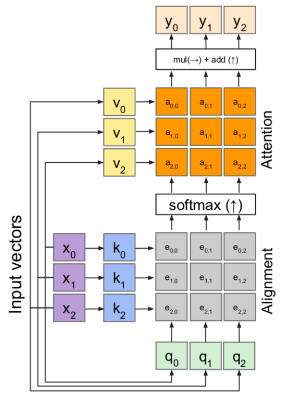

Figure 3: Graphical representation of the attention process taken from lecture slides of Xavier Alameda-Pineda.

Feed forward block

The second important block of the transformer-decoder architecture is the feed forward block which correspond 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. 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 connections 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 bias. The forward pass of the FFN is done independently for each token of the sequence, in our 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 as lower bound of the FFN this complexity.

We can see on Table 4 the architecture of the StarCoder model where we can retreive the different variables that we definned :

- 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 matrix 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 head. Because we previously consider that the attention computation of each head is parallelized, this difference don't change our lower bound. So we can now compute the lower bound of $FLOPs$ of the architecture :

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

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

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

Hyperparameter	**SantaCoder**	**StarCoder**
Hidden size	2048	6144
Intermediate size	8192	24576
Max. position embeddings	2048	8192
Num. of attention heads	16	48
Num. of hidden layers	24	40
Attention	Multi-query	Multi-query
Num. of parameters	$\approx$ 1.1B	$\approx$15.5B

Figure 4: Table which present the characteristics of the architecture of the StarCoder model [LAZ+23].

Roofline model

In order to compare the deployment models that we want to study (edge and large scale) in a computational point of view, and to estimate a lower bounds on the execution time require for 1 inference in both deployment context, 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 $FP32$ so we consider the numbers of bytes transfer as : $15.5 \times 10^9 \times 4 \approx 57.74GB$.
Regarding the hardware specification[3] requires 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 memory bandwidth capacity of 136.5 GB/s, a peak computing performance of 1410 $GFLOPS$, a die area of 350 mm and a Thermal Design Power (TDP) of 30 W.

Large scale computing deployment : We consider that the inference process is performed by using 1 NVIDIA A100 SMX4 40G with memory bandwidth capacity of 1.56 TB/s, a peak computing performance of 19.49 $TFLOPS$, a die area of 826 mm and a TDP of 400 W.
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 on Figure 5

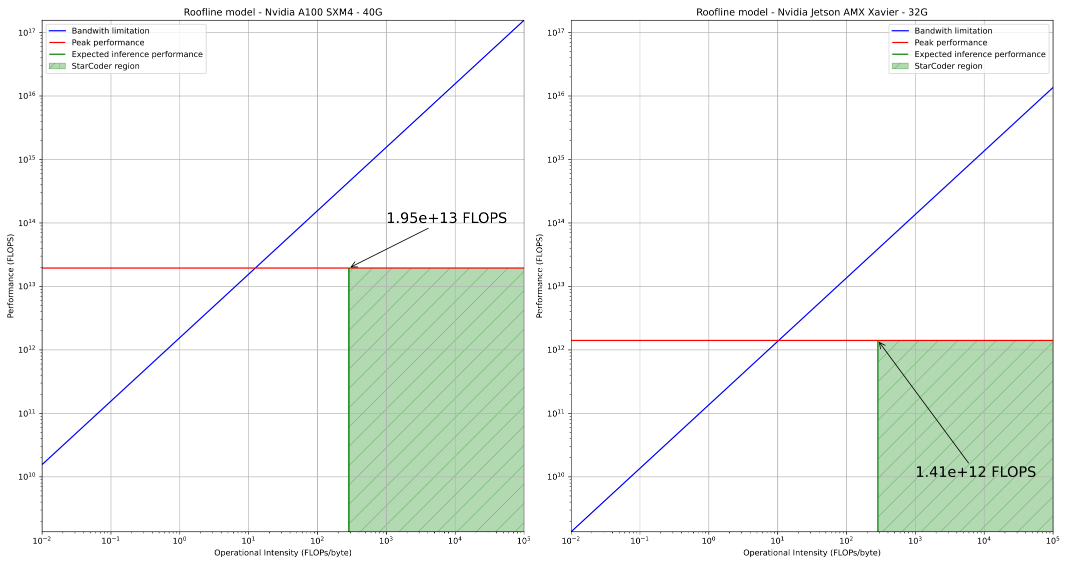

Figure 5: Graphs of the roofline model for Large-scale deployment (left) and edge device deployment (right).

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 complete 100% of the peak performance of the

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

hardware in both settings (reach the red line in Figure 5) :

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

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

QoE evaluation

With the approximations defined in the previous sections, we can apply the QoE methodology previously defined. By knowing that the same model is used in both deployment's contexts, our approach will not consider the precision of the model.

Environmental impact estimation

With our previous approach, we defined lower bounds of the execution time of the inference process, we based our approximation of the environmental impacts on these estimations combined with the hardware specifications of both deployment's contexts. To implement our comparative analysis, we will consider the following assumptions :

- Both settings are powered by the electricity production of the same geographical region g where $EGM_g = 1$ for ADP, GWP and PE.
- Both settings have an ideal $PUE = 1$.
- $a_e(t)$ can be considered as t.
- $C_{i,e}$ can be considered as the TDP of the hardware used.
- $I_{die\ area}$ and $I_{memory\ size}$ are equal to 1 for ADP, GWP and PE.
- the base impact I_ϵ is equal to 0 for ADP, GWP and PE.

With these assumptions, we can easily compute the environmental impacts of one inference for both settings as :

$$I_{Edge} = TDP_{Edge} + Execution\ time_{Edge} \times (die\ area_{Edge} + memory\ size_{Edge}) \approx 834$$

$$I_{Large\ scale} = TDP_{Large\ scale} + Execution\ time_{Large\ scale} \times (die\ area_{Large\ scale} + memory\ size_{Large\ scale}) \approx 522.6$$

Discussions

Actually, we can't infer anything's from these results due to the huge amount of assumption made. In my point of view, the next step will consist in defining more precisely the method to compute the environmental impact (with fewer assumptions) and after that, defining an experimental methodology to measure the environmental impact of inference on real edge/large scale platforms.

References

[BCJL24] 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.

[CQR24] 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.

[DZF$^+$20] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020.

[LAZ$^+$23] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

[MLN24a] 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.

[MLN24b] 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.

[Vas17] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.