Frugal AI Guide

Introduction to the concept of Frugal AI and the main levers of action

Table of Contents

I. Key Concepts

I.1. The cost of AI: a growing environmental impact

> *"AI energy use in data centres is likely to reach 200 – 400 TWh in 2030", IEA [1]*

For comparison, total electricity consumption in France in 2024 was 449 TWh [2]. This projection highlights a **major issue**: the urgency to better **measure, understand, and reduce** AI's energy footprint.

This is the core of Frugal AI: **measure, understand, reduce**.

I.2. Frugal AI: definition

Frugal: *"Living on little, living simply."* [3]

Frugal AI: Performance that meets the need at lower cost/impact.

To design a frugal AI service, it is therefore necessary to [4]:

- demonstrate the necessity of using an AI system ;
- adopt best practices ;
- and strive to stay within planetary boundaries.

I.3. Stakes: not just about ecology

- 🪴 **Environmental impact**: *Greenhouse gases (GHG), rare earths, etc.*
- 💎 **Scarce data**: *Sensitive or unavailable data*
- 🏭 **Limited resources for training**: *Energy consumption, small budgets, etc.*
- 📱 **Limited resources at deployment**: *Edge computing, IoT, etc.*
- 🌏 **Equity**: *Reducing cost (deployment and use)*

II. Orders of magnitude: LLMs, a growing consumption

> *« Shatter the illusion of "free compute time." » [5]*

II.1. Orders of magnitude for LLMs

OpenAI (inference)

- **Point of comparison**: **~0.3 Wh** per Google search (! a figure now being questioned)
- **Estimated inference consumption of GPT-4o [7]**:
  - **1 billion requests/day on ChatGPT** [6]
  - GPT-4o: **[0.42 – 1.79 Wh]** per request *(sometimes much higher depending on sources; these figures only account for energy consumed during the request, not the impact of infrastructure construction or model training)*
  - **Estimated annual consumption for 2025**: **[391 – 463] GWh** (> 35,000 average U.S. households)
- **Estimates from another study [8]**:
  - Study **[8]** evaluates the energy consumption during inference for several language models in a standardized text generation task. The task consists of generating **the next 10 tokens** for each input. Consumption is measured over **1,000 generations per model**.

Evaluated Model	Size (parameters)
gpt2	124 M
gpt2-medium	355 M
gpt2-xl	1.5 B
distilgpt2	82 M
bigscience/bloom-560m	560 M
EleutherAI/gpt-neo-125m	125 M
facebook/opt-1.3b	1.3 B
facebook/opt-6.7b	6.7 B

  - **Average size of tested models**: ~1.35 billion parameters.
  - **Key results**:
    - **Average consumption**: 0.047 kWh ± 0.03 for 1,000 generations of 10 tokens (i.e. **0.047 Wh per generation**).
  - **Extrapolation for very large models**:
    - Assuming a linear relationship between the number of generated tokens and consumption, an estimated consumption of **1 Wh for 200 tokens** is derived.
    - Assuming a linear relationship between model size (number of parameters) and energy consumption…

      > *"Benchmarks have shown impacts are roughly proportional to model size: a model 10 times bigger will generate impacts one order of magnitude larger than a smaller model for the same amount of generated tokens."* ***[10]***

      …we can estimate the consumption for larger models, such as **GPT-4**:
      - **Architecture**: Mixture of Experts (MoE) with a total of **1,700 billion parameters**. Between **220B and 880B active parameters during inference [12]**.
      - **Estimated consumption**: **160 Wh for 200 tokens** (based on 220B active parameters).
    - This estimate contrasts with the **0.42 to 1.79 Wh** reported in the first study **[7]**, highlighting **significant variability** depending on methodologies and assumptions.
  - This estimate remains **approximate** and relies on several assumptions, requiring caution. However, it reveals two major challenges:
    1. The difficulty of accurately assessing the energy consumption of models, particularly due to the **lack of transparency** from key industry players.
    2. The **high variability of results** across studies, complicating comparisons.
- **GPT-4o tests via Ecologits (2025)**:
  - Query [input: ~ 6K tokens, **output: <256 tokens**]: ~ **10g CO_2e** in the worst case

Mistral AI (training)

- Training **Large 2 (123B)** over 18 months:
  - **20,400 tons CO_2e** [9]
- Equivalences:
  - ✈️ **11,500** round trips Paris–New York
  - 🐄 **730 tons** of beef
  - 👥 **2,300 French people** for **1 year**

> *GPT-4 leak: 1.8 trillion parameters (16 × 111B) [https://archive.ph/2RQ8X]*
> *Consumption is roughly proportional to size. [9]*

II.2. LLM impact strongly depends on model and energy mix

For Mistral Large 2 (123B), a request generating one page of text (400 *tokens*) emits about **1.14 g CO_2e** [9].

> *Includes upstream emissions (e.g., server construction), but not the end-user terminal [9].*

For frequent usage, say 50 requests/day for a year: **≈ 20.8 kg CO_2e**

> *Impact remains limited*

Now let's see the same usage for a heavier LLM, based in the USA:

Consider another LLM with a structure similar to GPT-4 [11]:

- **Mixture of Experts (MoE)** 1700B
- **Active parameters at inference:** 220B to 880B

For frequent usage, say 50 requests/day for a year: **≈ 400 kg to 1.6 tons CO_2e!**

> *Approaching the 2 tons CO_2e/year target to limit global warming to +1.5°C*

Assumptions

1. Inference of *Large 2* is performed in Sweden [11] (low-carbon electricity: 36 g CO_2e/kWh, 2024) [12]
2. Carbon intensity in the USA is **384 g CO_2e/kWh (2024)** [12]
3. Energy consumption is roughly proportional to model size [9], including upstream emissions by default (granular emission decomposition unknown)

II.2. A need for transparency

> ***It remains difficult today to precisely measure the real energy consumption of AI***
> *(Financial Times, Sacha Luccioni: "We still don't know how much energy AI consumes")*

- Many studies attempt to estimate the footprint of LLMs, but results vary widely [10]:
  - Different methods
  - Variable scopes
  - Rapidly evolving field

Need: transparency and standardization of practices [4]

III. Orders of magnitude: Giving meaning to frugality

The goal here is to **take a step back**, and better understand what a **Wh** or a **g CO_2e** represents.
Better **interpret numbers** and **put the real energy impact of data science projects in perspective**.

🎯 kWh Quiz!

> 🔌 *With **1 kWh**, I can watch **TV**:*
> - *A. a few minutes*
> - *B. a few hours*
> - *C. a few days*

> 🧺 ***1 washing machine cycle*** *equals:*
> - *A. 1 kWh*
> - *B. 5 kWh*
> - *C. 15 kWh*

🌍 CO_2e Quiz!

> 🥩 ***1 kg of beef*** *equals:*
> - *A. 28 kg CO_2e*
> - *B. 96 kg CO_2e*
> - *C. 233 kg CO_2e*

> 👕 ***1 cotton t-shirt*** *equals:*
> - *A. 209 g CO_2e*
> - *B. 6 kg CO_2e*
> - *C. 58 kg CO_2e*

Answers

- B. a few hours
- A. 1 kWh

- A. 28 kg CO_2e
- B. 6 kg CO_2e

Wh EQUIVALENT

1 kWh is roughly... [13]

- 📺 3–5 h of television
- 🧺 1 washing machine
- 🚿 ½ shower
- 🍗 cook a chicken
- ❄️ 1 day of fridge operation
- 🛁 ¼ bath

CO_2 EQUIVALENT

1 kg CO_2e is roughly... [14][15]

Equivalent	1 kg CO_2e
📧 Emails	406
💾 Downloaded data	105 GB
💻 Video call	17.5 h
🎬 Streaming	15.6 h
🔍 Web searches	813
🍗 Beef meal	0.14
🍽️ Vegetarian meal	1.96
🥖 Traditional baguette	1.29 kg

Manufacturing a laptop is roughly... [14][15]

💻 Laptop	1
Equivalent	193 kg CO_2e
💻 Video calls	3 382 h
🚗 Carpooling (1 passenger)	1 774 km
🚄 High-speed train	65 870 km
✈️ Plane (medium haul)	1 029 km
🥔 Potatoes	273 kg

🔁 kWh ↔ CO_2e Conversion

France (2024): 44 g CO_2e/kWh 👉 **1 kg CO_2e = 22.7 kWh** [12]

🌐 Low-carbon energy mix = low digital impact

- 🇺🇸 **USA**: 384 g CO_2e / kWh [12]
- 🇨🇳 **China**: 560 g CO_2e / kWh [12]

> *Up to **10× more emissions** depending on the country!*

IV. Frugality & *Machine Learning*: Major levers of action [4][5]

1️⃣ Training and awareness

🎯 **Goal:** Raise awareness among all stakeholders about digital sobriety issues to enable informed choices from project conception.

2️⃣ Impact measurement

📊 **Goal:** Estimate environmental footprint at every stage of the AI project lifecycle to guide decisions and justify trade-offs.

- Estimate computing volume and energy consumption from the design phase (tool: Green Algorithms [16]).
- Track and report actual energy consumption (and other environmental indicators) throughout the project (development, inference, etc.) systematically (tools: CodeCarbon [17], Ecologits [18]).
- Analyze results to guide development and maintenance of the solution (e.g., gains achieved after each optimization).

3️⃣ From lightest to heaviest solution

💡 **Goal:** Evaluate the real necessity of using AI and favor simple (*e.g., decision tree*) and frugal approaches whenever possible.

4️⃣ Reuse existing resources

🔁 **Goal:** Promote resource sharing rather than systematic development of new solutions.

- Favor reuse of existing models.

- Share developed solutions (models, code, APIs, documentation).
- Avoid brute-force approaches: consult benchmarks and documentation upfront to limit exploration.

5️⃣ Sobriety in practices

🚫 **Goal:** Reduce unnecessary or excessive computations that needlessly increase environmental footprint.

- Reduce redundant computations: representative subsampling, early stopping, fewer folds in cross-validation, etc.
- Save intermediate checkpoints to avoid recalculating after unexpected interruptions.
- Define clear criteria to justify and trigger retraining once in production.

6️⃣ Technical optimization

⚙️ **Goal:** Use optimization tools and techniques to make solutions more efficient and less energy-intensive.

- Assess profitability of optimizations (e.g., compressing a model depends on expected inference gains).
- Apply compression techniques to limit inference impact (if profitable): quantization, pruning, knowledge distillation, compilation, etc. (tools: PyTorch, ONNX [19], Pruna AI).
- Favor compact, specialized models over oversized generic models.
- Favor recent, optimized libraries.

7️⃣ Rational use of hardware

🖥️ **Goal:** Prioritize using existing infrastructure and optimize its use before investing in new equipment.

- Reduce hardware turnover (high manufacturing impact).
- Adapt solutions to available resources.
- Avoid memory oversizing: energy consumption depends on allocated memory, not used memory! Allocating "just in case" = unnecessary carbon footprint.

8️⃣ Responsible data management

📦 **Goal:** Limit impact through data collection, storage, and processing.

- Use open-source data for prototyping and limit unnecessary collection.
- Favor data quality over quantity.
- Define a rational storage strategy: compress, archive, or delete if needed.

V. When frugality becomes a trap: Limits and points of attention

1️⃣ Take a step back from requirements

A solution can be effective relative to its specifications without being frugal.

To be frugal, it is necessary to **ensure upstream** that the **requirements** are: **technically realistic**, **necessary**, and **aligned with planetary boundaries**.

2️⃣ Simple solutions are sometimes best

Increasing complexity and computation costs should only occur if **simpler methods** cannot achieve the **set objectives (specifications)**.

It is important to **demonstrate the need for this complexity**.

3️⃣ Better is the enemy of good

Every optimization triggers experiments, iterations, and additional training, which may generate significant energy and resource costs.

Ensure optimizations provide a **real, measurable, and necessary benefit** according to specifications, avoiding resource use for marginal gains.

4️⃣ Beware of rebound effects [21]

System efficiency improvements can paradoxically **increase overall impact**, by lowering adoption barriers and massively increasing usage.

"Some environmental gains due [...] to technical improvements are significantly reduced or canceled by increased consumption or changes in usage." [22]

« As it currently stands, all of the scientific progress in terms of energy efficiency for AI will be offset by a rebound effect. » [23]

VI. Measurement tools

Possible measurements

A. Model size

- Number of weights
- Required memory

B. Computation cost

- FLOPs: floating-point operations (addition, multiplication, etc.)
- MACs: multiply-then-add operations (1 MAC = 2 FLOPs) – ptflops library

C. Energy consumption

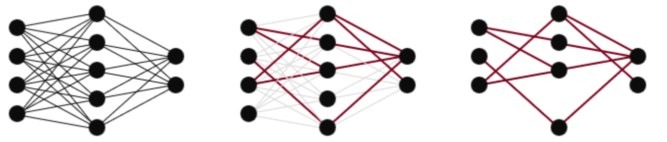
- **Green Algorithms** [16]:
  - Manual assessment of energy impact
  - Resources and publications gathering best practices for sustainable, eco-responsible data science
  - Website: https://www.green-algorithms.org/
- **CodeCarbon** [17]:
  - Measures code energy consumption and associated carbon emissions
  - Requires some access rights (e.g., Intel RAPL file on Linux to measure CPU consumption)
  - Some dependencies depending on OS and need:
    - pynvml to measure Nvidia GPU activity
    - Intel Power Gadget required for CPU on Windows or Mac (manual installation needed [24])
  - Detailed methodology on [17] and GitHub
  - Quickstart: https://mlco2.github.io/codecarbon/usage.html
- **Ecologits** [18]:
  - Measure impact of API calls to LLMs
  - Simple to use, methodology detailed on the website [18]
  - Quickstart: https://ecologits.ai/latest/tutorial/
- Integrated tracking on GCP

VII. Frugal AI and *Deep Learning*

Deep Learning relies on **often very large models**, and reducing their environmental footprint is a crucial challenge with **high impact potential**.

VII.1. Lottery Ticket Hypothesis [25]

> *Even randomly initialized, a dense network hides a performant subnetwork that, trained alone, can match the full model's accuracy with similar training time.*


Entrainement du réseau dense
Identification du bon sous-réseau (pruning)
Réinitialisation des poids et entrainement

Note: The larger the network, the higher the probability of finding a good subnetwork, making it necessary to train large models to extract efficient versions.

Note regarding classical Machine Learning: Be careful with hyperparameter optimization (recommended: Optuna with multi-objective optimization between performance and energy efficiency) and feature selection (favor model-integrated importance if possible, e.g., linear, forest, boosting methods).

VII.2. Lever of action: Neural network compression

By design, Deep Learning creates heavy models, with **more neurons than necessary**.
Tools and techniques exist to reduce this load while maintaining good performance:

- Hybrid architectures
- Adaptive architectures
- Pruning
- Quantization
- Knowledge Distillation
- Low Rank Optimization
- ...

VII.3. Compression: Adaptive architectures

VII.3.A. *Early-exit networks* [26]

Principle

- "Early outputs" at different layers
- Stops inference earlier if confidence is sufficient

Advantages

- Faster inference
- Reduces Overfitting, Overthinking, and Vanishing Gradients

Point of attention

- Number, placement, and structure of exit branches
- Choice of exit criterion (Static VS Dynamic policies)
- Choice of training strategy (Joint, Branch-wise, etc.)

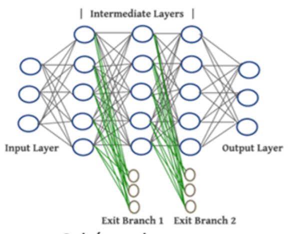
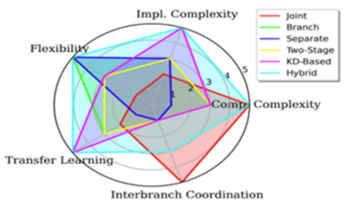


Schéma du concept



Les stratégies d'entrainement

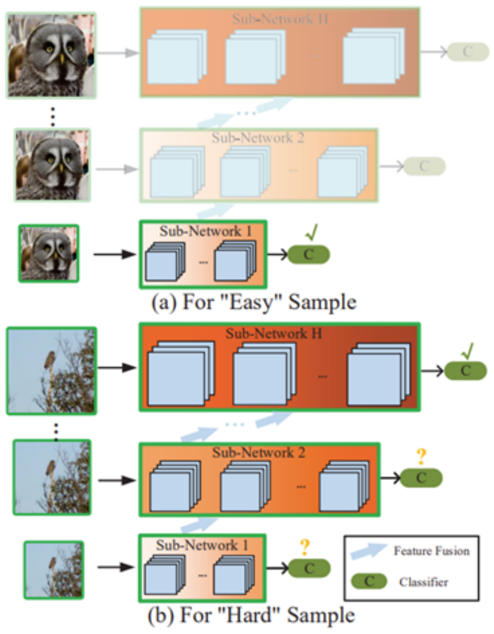
VII.3.B. *Resolution-adaptive networks* [27]

Principle

- Adapt input resolution based on complexity

Advantages

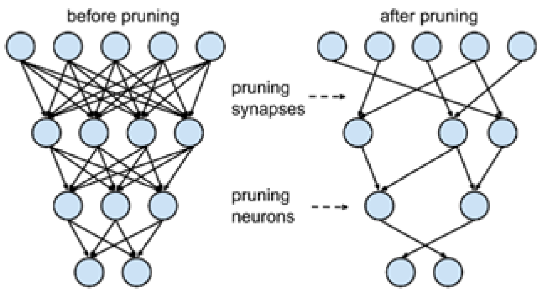
- Faster inference



VII.4. Compression: *Pruning*

VII.4.A. Concept

Pruning? Removing parameters (weights, biases, neurons, etc.) from a neural network (**fine-tuning afterward is recommended**)



Which parameters to remove?

Most common pruning: based on **weight magnitude** (Ln norms, absolute value (L1) for unstructured). Other methods [28]:

- 🎰 random

- 🪞 clustering similar weights
- 🧪 sensitivity analysis, etc.

VII.4.B. Technical approach

Prune all at once or gradually?

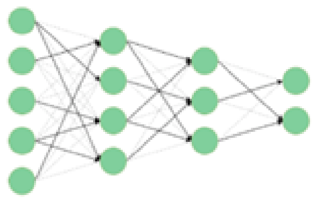
- 1️⃣ **One-shot pruning**: full pruning in one iteration (complexity - ; performance ~)
- 🔄 **Iterative pruning**: gradual pruning, remove a proportion 1 − (1 − k)^(1/n) per iteration, reach k after n steps (complexity + ; performance +)
- 💡 Regularization recommended during fine-tuning (to reduce unnecessary weight magnitudes). L2 norm recommended (see 3.1 of [29]).

Prune 🌏 globally or 🔬 locally (per layer)?

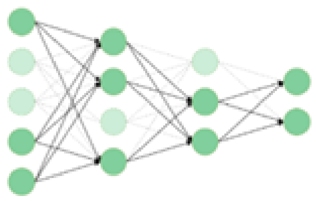
- **Local pruning**:
  - Pruning inside each layer
  - Example: 3-layer network (800 > 180 > 20) → 1000 weights. Local pruning 80% → (160 > 36 > 4)
- **Global pruning**:
  - Pruning across the whole network
  - Example: 3-layer network (800 > 180 > 20) → 1000 weights. Global pruning 80% → may become (190 > 5 > 5)
  - Note: global pruning may achieve better compression for same performance but can over-prune layers, creating bottlenecks detrimental for the network's performance

Structured or unstructured pruning?

- ***Unstructured pruning***



  - Removes individual weights independently, easy to implement
  - Generates **sparse matrices**, compatible with compact representations (e.g., CSR), reduces memory used
  - **Limited impact** (sometimes even negative) on latency depending on hardware
- ***Structured pruning***



- Removes entire structures (e.g., neurons, convolution filters)
- Effectively reduces model size and complexity, **modifies architecture** (sometimes complex)
- Provides **real latency gains** and deployment if the full network saturates hardware

On paper, pruning has potential! But…

- Unstructured pruning only has an impact if the hardware takes advantage of matrix sparsity (rare)
- Structured pruning can quickly become a real headache…

Why is structured pruning complicated?

- Interdependencies between different elements of the neural network: removing a neuron requires adapting adjacent elements, and this can quickly become quite complex depending on the architecture involved [30].

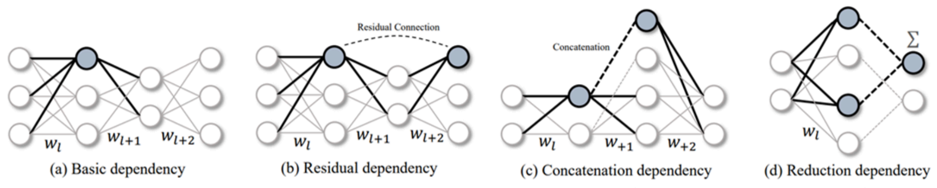


Figure 2[1]: Grouped parameters with inter-dependency in different structures. All highlighted parameters must be pruned simultaneously.

Pour être concaténés, les tenseurs doivent être de dimensions cohérentes

Exemple : *max pooling*

> *Fortunately, it is not necessary to code everything ourselves!*

Examples:

- The Torch-Pruning (TP) library [30] drastically simplifies structured pruning on architectures of varying complexity
- Pruna AI [20]: simplifies model compression: structured pruning (via Torch-Pruning) + many other methods (quantization, compilation caching, batching, etc.)
- Even simpler: use an already compressed model! Pre-trained and compressed models are available: Hugging Face, Pruna, Unslot (LLM) ...

VII.5. Compression: Quantization

VII.5.A. Concept

Quantization? Reduce the precision of weights and activations.
Example: converting from 32-bit to 8-bit:

- memory cost divided by 4
- compute cost divided by 16

Quantization affine (asymétrique) : mapping sur 2^b valeurs

$$\hat{\mathbf{x}} = q(\mathbf{x}; s, z, b) = s\left[\text{clamp}\left(\left\lfloor\frac{\mathbf{x}}{s}\right\rceil + z; 0, 2^b - 1\right) - z\right]$$

Avec l'élément à quantizer ***x***, le facteur d'échelle ***s***, le point zéro ***z*** et le nombre de bits ***b***. ***s***, ***z*** et ***b*** sont à définir.

Intervalle de quantization : $q_{\min} = -sz$ et $q_{\max} = s(2^b - 1 - z)$

💡 Augmenter ***s*** **augmente l'intervalle** & les **erreurs d'arrondi**

Quantization symétrique (***z*** **= 0**) : **- flexible + rapide**

- unsigned integers : distributions unilatérales (e.g. ReLu)
$$\hat{\mathbf{x}} = s.\text{clamp}\left(\left\lfloor\frac{\mathbf{x}}{s}\right\rceil; 0, 2^b - 1\right)$$
- signed integers : distributions symétriques autour de 0
$$\hat{\mathbf{x}} = s.\text{clamp}\left(\left\lfloor\frac{\mathbf{x}}{s}\right\rceil; -2^{b-1}, 2^{b-1} - 1\right)$$

[31]

VII.5.B. Technical approach

⏸️ Post-Training Quantization (PTQ)

- Model first trained in full precision (typically float32), then quantized
- ✅ Easy to implement (no retraining/fine-tuning)
- ⚠️ Possible performance drop

Two approaches:

- Dynamic PTQ: model weights quantized in advance, activations quantized at inference
- Static PTQ: model weights and activations quantized in advance using a calibration set

▶️ Quantization-Aware Training (QAT)

- Quantization simulated during training: model learns to operate in low precision
- ✅ Reduces performance degradation (if PTQ is insufficient)

- ⚠️ Computationally intensive

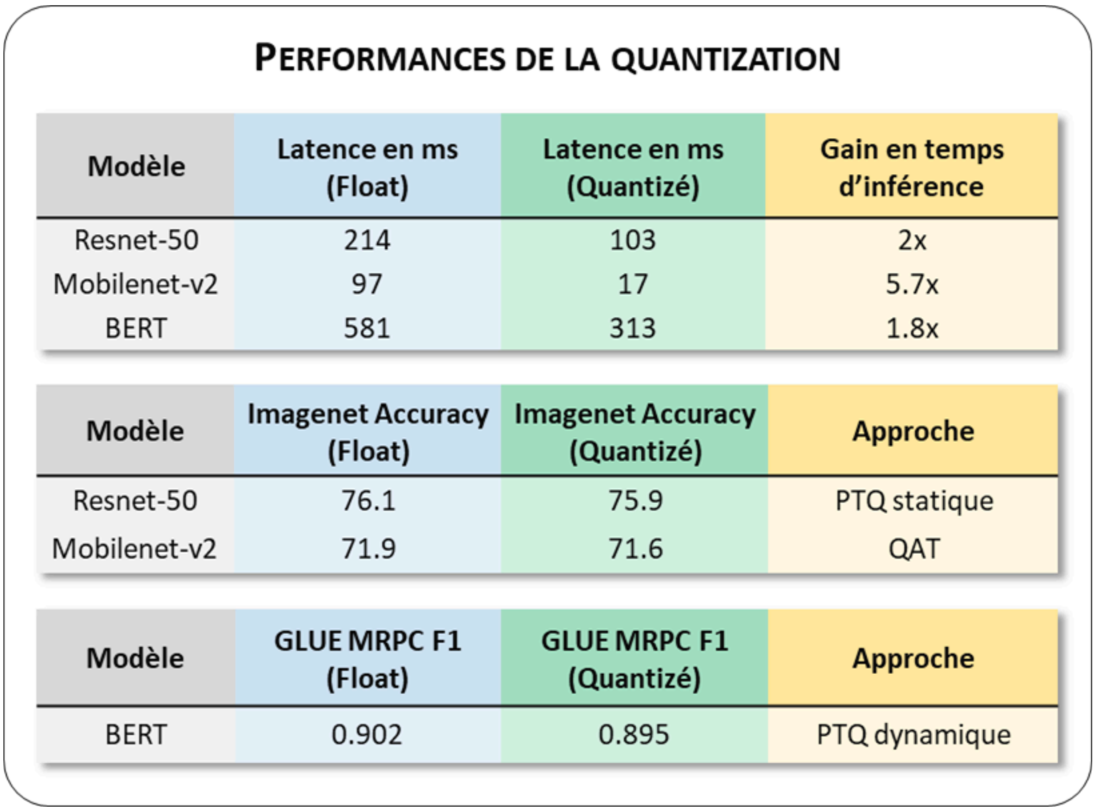
💡 **Layer fusion**

- ⚠️ Apply just before static PTQ or QAT in PyTorch [32]
- 🔗 Combines successive modules (e.g., Conv2d (+ BatchNorm) (+ ReLU), Linear + ReLU, etc.)
- ⚡ Speeds up inference and reduces quantization errors

Efficiency

Quantization provides significant speed gains with minimal impact on performance.

Results available in PyTorch documentation [33].



PERFORMANCES DE LA QUANTIZATION

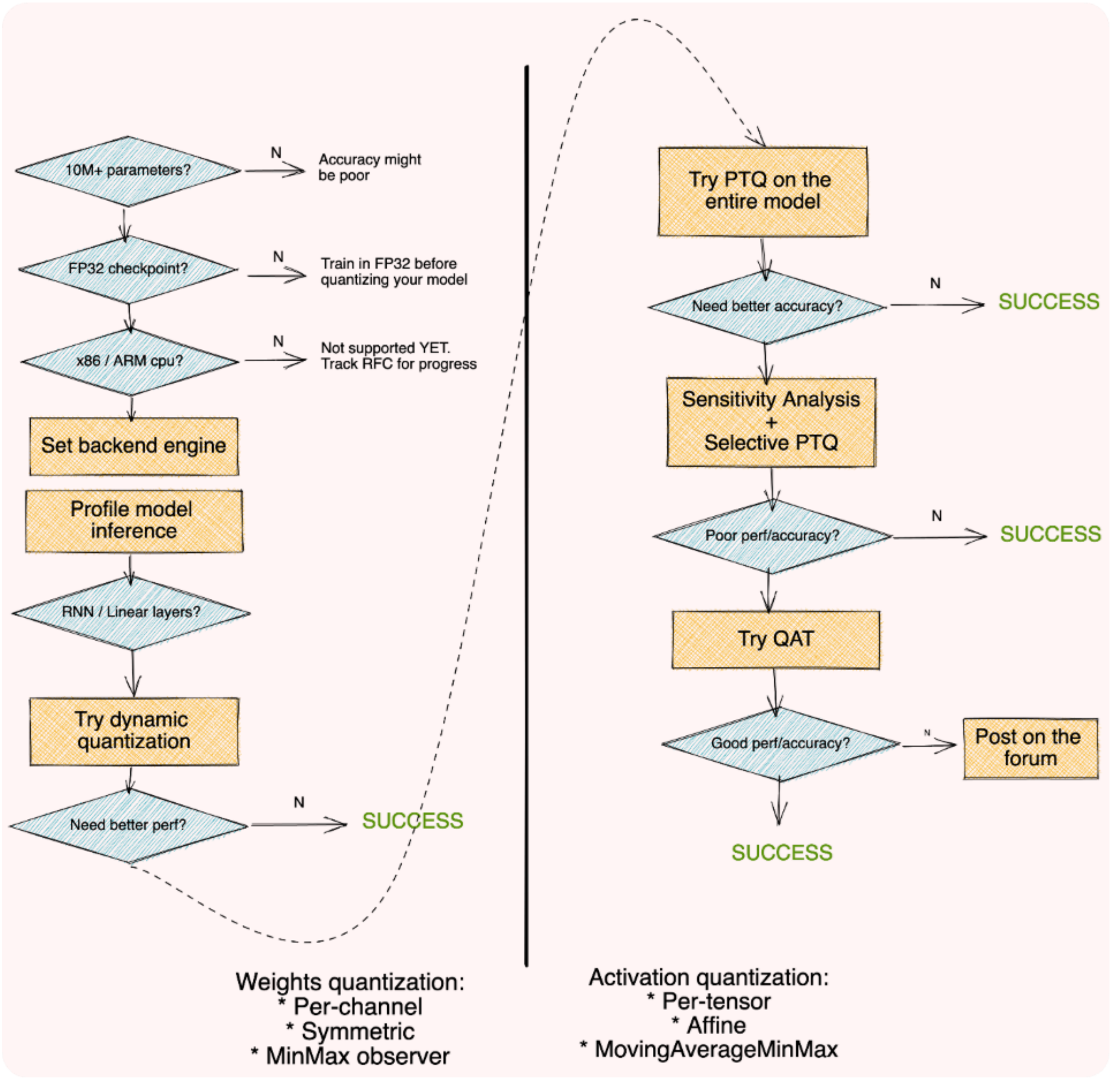
Modèle	Latence en ms (Float)	Latence en ms (Quantizé)	Gain en temps d'inférence
Resnet-50	214	103	2x
Mobilenet-v2	97	17	5.7x
BERT	581	313	1.8x

Modèle	Imagenet Accuracy (Float)	Imagenet Accuracy (Quantizé)	Approche
Resnet-50	76.1	75.9	PTQ statique
Mobilenet-v2	71.9	71.6	QAT

Modèle	GLUE MRPC F1 (Float)	GLUE MRPC F1 (Quantizé)	Approche
BERT	0.902	0.895	PTQ dynamique

https://pytorch.org/blog/introduction-to-quantization-on-pytorch/#performance-results" width="400">

Quantization workflow [32]



https://docs.pytorch.org/assets/images/quantization-practice/quantization-flowchart2.png" width="500">

VII.6. Compression: *Knowledge Distillation*

VII.6.A. Concept

Train a **simple model (student)** to imitate the behavior of a **complex model (teacher)** using true labels and **soft targets** (teacher output probabilities)

Use case: multi-class classification

> *Soft targets provide a lot of information about relationships (distances) between classes.*
> *"An image of a BMW, for example, may only have a very small chance of being mistaken for a garbage truck, but that mistake is still many times more probable than mistaking it for a carrot" [34]*

VII.6.B. Technical approach

🔥 **Temperature importance (in softmax)** [34]

- Smooths teacher predictions to help the student learn class nuances

📉 **Loss function**

$Loss = \alpha.T^2.CE(softmax(z\_s/T), softmax(z\_t/T)) + (1-\alpha).CE(softmax(z\_s),y)$

- z_s = student logits, z_t = teacher logits, y = labels, T = temperature, $\alpha \in [0,1]$ weighting the soft targets
- First term weighted by T^2; temperature affects softmax gradients

- High-entropy soft targets: more information per observation and more stable gradients → allows training on less data, possibly with a higher learning rate

Implementation recommendations

- Prefer Kullback-Leibler divergence over cross-entropy for soft-loss (same gradients but better interpretability) [35]
- Give more weight to the soft-loss ($\alpha > 0.5$) [35]

> *To go furhter [36]*

VIII. Inference optimization

Goal: Improve model efficiency at inference, reducing processing time and energy consumption without modifying the model

Batching

Principle: Group multiple requests for simultaneous processing, useful for high-traffic systems

Optimal batch size:

- too small → no effect
- too large → memory overload

Compilation

Principle: Transform the model into an optimized execution version (compilation or export to an intermediate format, e.g., ONNX [19]).

Example: ONNX exports a model (e.g., PyTorch) to a standard format, then runs efficiently via ONNX Runtime [37] with hardware-specific optimizations.

Useful tools: ONNX Runtime, Pruna, TensorRT, BitsAndBytes, Apache TVM, OpenVINO, etc.

Note: ONNX allows easy framework conversion (e.g., PyTorch → ONNX → TensorFlow).

IX. Conclusions

- Frugality starts with common sense and sobriety!
- Reduction leverage is higher for Deep Learning, but a culture of sobriety and responsible habits is useful in all cases.
- Advanced methods (pruning, quantization, etc.) only have impact when the model is large enough and saturates the hardware used. Otherwise, optimization can even be counterproductive.
- Whenever possible, prefer simple, and/or pre-trained and compressed models!
- Data science obeys the Pareto principle: very often, the majority of gains come from a minority of efforts, and many practitioners waste a lot of time trying to marginally optimise an architecture. In practice, significant performance improvements usually come from adding new data and/or creating new relevant features.
- A good practice for data teams is to identify a "Digital Sustainability Manager". Their role consists, for example, of systematically questioning the purpose of projects, facilitating the implementation of consumption monitoring tools (Green Algortithms, CodeCarbon, EcoLogits), monitoring the evolution of practices, and training employees in the use of these tools.

Sources

[1] Data Centre Energy Use: Critical Review of Models and Results (IEA - 4E), p.33 - https://www.iea-4e.org/wp-content/uploads/2025/05/Data-Centre-Energy-Use-Critical-Review-of-Models-and-Results.pdf

[2] La consommation d'électricité en chiffres (EDF) - https://www.edf.fr/groupe-edf/comprendre/electricite-au-quotidien/usages/consommation-electricite-en-chiffres

[3] Définition tirée du dictionnaire en ligne Larousse - https://www.larousse.fr/dictionnaires/francais/frugal/35451

[4] Référentiel général pour l'IA frugale - https://www.ecologie.gouv.fr/presse/publication-du-referentiel-general-lia-frugale-sattaquer-limpact-environnemental-lia

[5] Ten simple rules to make your computing more environmentally sustainable - https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1009324

[6] Déclaration OpenAI, déc. 2024 – *The Verge* - https://www.theverge.com/2024/12/4/24313097/chatgpt-300-million-weekly-users

[7] *How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference* (15/07/2025) - https://arxiv.org/pdf/2505.09598

[8] *Power Hungry Processing: Watts Driving the Cost of AI Deployment?* (15/10/2024) - https://arxiv.org/pdf/2311.16863

[9] Mistral AI – Our contribution to a global environmental standard for AI - https://mistral.ai/news/our-contribution-to-a-global-environmental-standard-for-ai

[10] Measuring the environmental impact of delivering AI at Google Scale - https://arxiv.org/pdf/2508.15734

[11] Ecologits: About proprietary models (cf. additional document) - https://ecologits.ai/latest/methodology/proprietary_models/

[12] Intensité carbone de la production d'électricité par pays (ourworldindata.org) : il s'agit de la source utilisée par CodeCarbon - https://ourworldindata.org/grapher/carbon-intensity-electricity

[13] Que peut-on faire avec 1 kWh ? (EDF) - https://www.edf.fr/groupe-edf/comprendre/electricite-au-quotidien/usages/que-peut-on-faire-avec-1-kwh

[14] Comparateur carbone (ADEME) - https://impactco2.fr/outils/comparateur

[15] Documentation méthodologique détaillée (ADEME) - https://impactco2.fr/doc/usage-numerique/acv

[16] Green algorithms - https://www.green-algorithms.org/

[17] CodeCarbon - https://mlco2.github.io/codecarbon/index.html

[18] Ecologits - https://ecologits.ai/latest/

[19] ONNX - https://onnx.ai/

[20] Pruna AI - https://docs.pruna.ai/en/stable/index.html#

[21] From efficiency gains to rebound effects (pour aller plus loin) - https://arxiv.org/pdf/2501.16548

[22] Effet rebond : définition du ministère français de la culture - https://www.culture.fr/franceterme/terme/ENVI231

[23] Gilles Sassatelli, research professor at LIRMM on rebound effect - https://news.cnrs.fr/articles/the-challenges-of-frugal-ai

[24] Archive contenant un lien de téléchargement - Intel Power Gadget (plus maintenu, mais fonctionne toujours avec codecarbon) - https://web.archive.org/web/20230325112308/https:/www.intel.com/content/www/us/en/developer/articles/tool/power-gadget.html

[25] The lottery ticket hypothesis: Finding sparse, trainable neural networks - https://arxiv.org/pdf/1803.03635

[26] Early-Exit Deep Neural Network - A Comprehensive Survey - https://dl.acm.org/doi/pdf/10.1145/3698767

[27] Resolution Adaptive Networks for Efficient Inference - https://openaccess.thecvf.com/content_CVPR_2020/papers/Yang_Resolution_Adaptive_Networks_for_Efficient_Inference_CV

[28] Methods for pruning deep neural networks (meta analyse/survey) - https://arxiv.org/pdf/2011.00241

[29] Learning both Weights and Connections for Efficient Neural Networks - https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf

[30] Torch-Pruning (github) - https://github.com/VainF/Torch-Pruning?tab=readme-ov-file#high-level-pruners

[31] A White Paper on Neural Network Quantization - https://arxiv.org/pdf/2106.08295

[32] Practical Quantization in PyTorch - https://pytorch.org/blog/quantization-in-practice/

[33] PyTorch Quantization, Performance Results - https://pytorch.org/blog/introduction-to-quantization-on-pytorch/#performance-results

[34] Distilling the Knowledge in a Neural Network - https://arxiv.org/pdf/1503.02531

[35] Why KL Divergence in Knowledge Distillation? (Medium) - https://medium.com/@buroojghani/why-kl-divergence-in-knowledge-distillation-1375d555a728

[36] Pour aller plus loin : d'autres types de Knowledge Distillation [IBM] - https://www.ibm.com/think/topics/knowledge-distillation

[37] ONNX Runtime - https://onnxruntime.ai/