Middle East Research Journal of Engineering and Technology
ISSN: 2789-7737 (Print) ISSN: 2958-2059 (Online)
Frequency: Bi-Monthly
DOI: https://doi.org/10.36348/merjet.2026.v06i01.001

KUWAIT SCHOLARS PUBLISHER
Website: http://www.kspublisher.com/
Email: office@kspublisher.com

Modeling Energy Use Per Generative AI Task: A Simplified Disaggregated Octave Framework Across End-User, Network, and Cloud Layers

Anders S.G. Andrae[1*]
[1]Huawei Technologies Sweden AB, Sweden, Skalholtsgatan 9, 16494 Kista, Sweden

Abstract: The energy consumption of AI and especially individual AI tasks is complex to measure. A critical aspect of the energy evaluation of AI systems is the precise definition of both the scope and methodology. It is not evident if the differentiation should occur at the task level or model level. Here it is argued that full task is the best entity for functional unit setting for LCA of AI systems. An example of data analysis is provided to show the usefulness and reasonability of the conceptual and analytical framework which helps identify hidden drivers. The proposed framework reveals that time-extended service phases are energy drivers which remain invisible in both interference-only and average LCA approaches. Main contributions are interaction-level energy accounting, theoretical expansion of existing LCA and scaling approaches and identification of dominate non-compute energy drivers.

Keywords: Cloud, Energy, Electricity, Generative AI, task.

Research Paper

***Corresponding Author:**
Anders S.G. Andrae
Huawei Technologies Sweden AB, Sweden, Skalholtsgatan 9, 16494 Kista, Sweden

How to cite this paper:
Anders S.G. Andrae (2026). Modeling Energy Use Per Generative AI Task: A Simplified Disaggregated Octave Framework Across End-User, Network, and Cloud Layers. *Middle East Res J. Eng. Technol, 6*(1): 1-10.

Article History:
| Submit: 07.12.2025 |
| Accepted: 05.01.2026 |
| Published: 09.01.2026 |

INTRODUCTION

The data centers (DCs) are using ever more power due to servers and others (Andrae, 2023). It has been estimated that AI DCs could be 20% of United States DCs electricity use in 2028 (≈100TWh) and all US DCs up to 12% of US total electricity use (≈500 TWh) (Shehabi *et al.,* 2024). Generally, the energy consumption of particular AI systems is complex to measure (Berthelot *et al.,* 2024) and simplifications are necessary, similar to software systems (Andrae 2024a). A critical aspect of energy evaluations of AI systems is the precise definition of both the scope and methodology. It is not evident if the functional unit (f.u.) in an AI Life Cycle Assessment (LCA) should be defined on full task level or model level. For example, f.u. such as the number of prompt tokens for text generation, the number of bytes for image generation, the number of bytes for audio recording, the number of bytes for video are inappropriate as f.u. for AI systems.

Here the definition of a full GenAI task is: a single user-initiated interaction that triggers the complete lifecycle of a service request including all associated compute, memory, network and service overheads required to fulfill that interaction.

Table 1 explains why task is better than bytes as f.u. for AI LCA.

Table 1: Criteria for functional unit setting in AI LCA

Criterion	F.u. Task	F.u. Bytes
Represents user function	Yes	No
Works across modalities	Yes	No
Normalizes environmental data	Yes	No
Scales with complexity	Yes	No
Promotes useful benchmarking	Yes	No

Moreover, the impact of Traditional AI (single models) and Generative AI (GenAI), featuring variation of tasks, are different. GenAI tasks have more significant impact (Desroches, 2025).

The present research is based on reasonable assumptions and probabilities adapting the method for data analysis software (Andrae, 2024a) for AI tasks. Therefore, the present study will only offer an initial suggestion for energy modeling of AI tasks. Extending the present research beyond the use stage to LCA is considered trivial.

In summary, for the first time a framework is presented which include:

1. Time-extended services phases
2. Separation of inference compute vs serving overhead
3. Training amortization tied structurally to parameters
4. Sensitivity analysis on system behavior

Experimental Section/Material and Methods

Apart from (Andrae, 2024a) the implementation is based on (ITU, 2022; Andrae, 2024b)

$$E_{tr} = C \times V^2 + \frac{1}{sp \times Clock_{chip}} \times I_{off} \times V \quad (1)$$

$$E_{tr} = E_{factor} \times k_b \times T \quad (2)$$

$$E_{factor} = \frac{C \times V^2 + \frac{1}{sp \times Clock_{chip}} \times I_{off} \times V}{k_b \times T} \quad (3)$$

$$W_{chip} = N_{transistors,chip} \times Clock_{chip} \times E_{tr} \times CompUE \quad (4)$$

$$E_{FLOP} = \frac{1}{\left(\frac{FLOPS_{chip}}{W_{chip}}\right)} \quad (5)$$

where

E_{tr}= Dynamic switching energy (J/transistor, J/erased bit).
C = Load Capacitance (As/V)
V = Voltage across the gate, (V)
sp = switching probability
$Clock_{chip}$= Clock frequency (1/s)
I_{off}= Leaking current drawn by each switch in the off-state (A)
E_{factor}= Dimensionless primary energy/enthropy factor
k_b= Boltzmann's constant (J/K)
T = Temperature at which the transistor is operating (K)
W_{chip} = Power consumption of one chip (W), energy
$N_{transistors,chip}$ = Number of transistors in one chip (#)
$CompUE$= Computational use effectiveness.
E_{FLOP} = energy use per floating point operation (J/FLOP)
$FLOPS_{chip}$ = floating point operations per second performance per chip (FLOPs/s)

Equations (4) and (5) are used in the GenAI calculations for Model Interference in section C.

The same case study as (Andrae, 2024a) is used however with GenAI features for the SW analytics. Similarly, the scope is end-user, network, and cloud overhead.

The functional unit is "The execution of one GenAI-assisted analysis task by an individual knowledge worker, generating a visual analytical output using cloud-hosted Large Language Model (LLM) infrastructure in 2024."

A. End-user Hardware use

This entity of the task energy model is about the energy used by the end-user device inputting the query and accessing the output.

It is assumed as in (Andrae 2024a) that the end-user is using a laptop or desktop for the GenAI analytics session. These components are included in local device-side computation and display:

- CPU/GPU usage (light computation, rendering)
- Memory and disk
- Screen
- Network interface (Wi-Fi, Ethernet)
- Browser (e.g., chart visualizations)

Table 2 shows the assumptions for power draw.

Table 2: Typical Power Draw of Devices and energy use for typical GenAI session

Component of end-user device	Power (W)	Time (s)	Energy (Wh)	Reference
CPU active	~15 W	60 s	~0.25 Wh	Cabaret *et al.*, 2025
Display (LCD/LED)	~8 W	120 s	~0.267 Wh	Huang *et al.,* 2025
Disk I/O	~2 W	10 s	~0.006 Wh	Ishengoma 2025
Memory/network	~3 W	30 s	~0.025 Wh	Caiazza *et al.*, 2024
Chart rendering	~5 W	40 s	~0.056 Wh	Dornauer and Felderer 2023; Horn *et al.*, 2023
			~0.6 Wh TOTAL	

B. Network Transfer

This entity of the task energy model is about the energy used when data are transmitted between the end-user devices and the cloud service including both the upload of the user prompt and the download of the model's output.

The amount of Wh for network transfer is uncertain both from Wh/MB and for amounts of MB viewpoint.

In some cases (with large outputs or explanations), the size may go up to 10 MB/analysis task. Network transfer should besides transmission also include:

- Application Programming Interface (API) routing latency
- Caching/storage overhead
- Control flow, encryption, security layers
- Often inflated due to cloud architecture inefficiencies (e.g., API gateways, containerized LLM orchestration)

- Multiple network hops (user ↔ API gateway ↔ inference server ↔ postprocessing)

So, network transfer should include more than raw *data movement*. It also reflects network-layer overhead in *practical GenAI inference systems*.

Wh/MB vary with network type but cloud + broadband is most common (Guennebaud and Bugeau, 2024). Full top-down view may use 0.22 Wh/MB, fixed optical 0.03, mobile 0.04, and data centers 0.006 Wh/MB, (Andrae, 2020).

MB/task is likely 0.2 – 0.5 for low tier tasks and 5 - 10 for multimodal high tier tasks. Hence the minimum energy use is 0.2×(0.03+0.006)=0.0072 Wh and maximum is 10×0.22=2.2 Wh. Table 3 shows typical size estimates for data transfer.

Table 3: Typical data size estimate for data transfer

Component of network transfer	Size estimate (MB)	Reference
User prompt → API	~0.01–0.05	Koneva *et al*., 2025
LLM Model output	~0.05–0.2	Perez *et al.,* 2025
Chart code	~0.2–0.6	Andersson and Grandin, 2025
Orchestration payloads	~0.1–0.3	Andersson and Grandin, 2025
Streaming	~0.5–1	Mukherjee, 2024; Koneva *et al,* 2025
Total transfer	**~1–2 MB** TOTAL	

According to LCA best practice a conservative estimate is to be applied so 0.22 Wh/MB is chosen: Energy (Wh)=Electricity use data transfer, practical (Wh/MB)×Data transferred (MB) = 0.22 Wh/MB × 2 MB/task = 0.44 Wh.

C. Cloud Use

Cloud use for GenAI is here assumed to consist of

- Model Inference
- API/LLM latency
- Memory Overhead
- Query Execution
- Training

1), Model Interference

This entity of cloud use is about the core computation process where a trained AI model generates an output. Interference is about using the already trained AI model to make predictions or generate outputs, i.e. reasoning by which conclusions are derived from known premises. The same task can include multiple interferences. Sometimes one interference is equal to one task. Here the interference energy is part of the task energy.

300–600 W per GPU is assumed (Gregersen *et al.*, 2024). In GenAI analytics the code + explanation + chart creation generate ~500–2000 tokens (Hedderich *et al.,* 2025). A token is a chunk of data used by the AI model, typically a word or piece of a word. Depending on latency the inference time is around 5–10 seconds **(**Argerich and Patiño-Martínez, 2024; Bian *et al.*, 2025). Hence, the energy use for hardware power draw is 400 W×0.00278 h = 1.1 Wh. This reflects moderate prompt length (~1000–2000 tokens), a single-user batch (not large-scale interference) and possibly multi-GPU context window handling. GPUs are often underutilized, but still draw power. So 1.1 Wh per inference is a conservative average for GPT-3.5 and GPT-4 class models.

An alternative method for calculating the energy use of model interference is to include parameters and sequence length and combine with equations (4) and (5). A parameter is an internal variable of a model that affects how it computes its outputs. The reason is that parameters is suggested as a very important driver for interference energy use per task. It is assumed 39.6 billion parameters (Gonzalez-Agirre *et al.*, 2025) and 1000 tokens of sequence length (Hedderich *et al*, 2025) → 2 (multiplication and addition in multi-add operation, 2 FLOPs)×39.6 billion×1000 = 7.92×10^{13} FLOPs.

Assumed $FLOPS_{chip}/W_{chip}$ = 1 TFLOPS/W and W_{chip} = 400 W.
Time = 7.92×10^{13} FLOPs /(400 W × 1×10^{12} W/FLOPS) = 0.198 seconds.
Energy (idealistic for pure GPU only) = 400 W×0.198/3600 h = 0.022 Wh.

However, 0.022 Wh only includes the matrix multiplications while the memory access, network stacking, cooling, load balancing, etc are excluded. Due to whole system power draw in data centers, a system overhead multiplier must be added. Memory+scheduling could add ≈4 times (Yoon *et al.,* 2025), API latency ≈3 times (Nõu *et al.*, 2025), Query orchestration overhead ≈20% (Hammad *et al.,* 2025), PUE 50% (Horner and Azevedo, 2016) and additional system idle variability ≈2 times (Jin *et al.*, 2020). All in all, the cumulative effect of these overheads could reach ≈50 times. That is 0.022 Wh more realistically has to be increased to ≈1.1 Wh for interference.

2). API/LLM Latency

This entity of cloud use is about the overhead energy use associated with running the AI model as a service.

The API/LLM latency represents the section where the interference service is active between request initiation and completion. Table 4 shows examples of power use for API LLM related components.

Table 4: Examples of power use for API LLM related components

Component	Power (W)	Time (s)	Energy (Wh)	Explanation	Reference
Container standby (warm state)	150 W	24 s	1.00 Wh	Cloud instance or container kept warm while awaiting user input or returning results	Raza 2021
Token streaming delay + I/O	120 W	15 s	0.50 Wh	Slow return of generated text tokens over WebSocket or API	Katal *et al.*, 2022
Prompt context preload	200 W	10 s	0.56 Wh	Video Random-Access Memory (VRAM) preloading of long prompts or embeddings before generation starts	Jin *et al.*, 2020
Retry + orchestration fallback	100 W	10 s	0.28 Wh	Sometimes prompts fail or are retried with fallback chains or formats	Jin *et al.,* 2020
Residual idle / buffer overhead	50 W	20 s	0.28 Wh	Idle waiting or orchestration-related polling	Katal *et al.,* 2022
Total Wh			**2.26 Wh**		

3). Memory Overhead

This entity of the cloud use is about the additional energy used to keep the AI model and related data loaded into memory also when the model is not actively computing.

Memory overhead includes large VRAM allocation to hold context (prompt + embeddings), persistent memory during LLM session even when not actively computing and use of GPU RAM and/or TPU memory and temporary storage of intermediate representations.

As far as power sources a single GPU is assumed to use in idle VRAM state: ~100–150 W and partially loaded state (holding prompt but not generating): ~200–250 W (Ikram *et al.*, 2017). Regarding time, 10–30 seconds is assumed while holding prompt context in memory. This leads to 200 × 25/3600 = 1.39 Wh is used. This means that the present model assigns the entire power use to one task despite of what else the GPU is handling.

4). Query Execution

This entity of the cloud use is about the final stage of processing a GenAI task where the system post-processes, formats, and delivers the model's output to the end-user. Table 5 shows examples of power use for query execution related components.

Table 5: Examples of power use for query execution related components

Backend Type	Typical use	Active Power (W)	Reference
vCPU	CPU use for parsing, planning and execution	~10–50 W	Katal, *et al*., 2022, Choochotkaew, *et al*., 2025
Memory	Buffer pool, caching, joins, sorting	~5–30 W	Legler, *et al*., 2025; Centofanti, *et al*.. 2024
Disk I/O (SSD)	Read/writes from local/remote storage	~5–20 W	Centofanti, *et al.,* 2024
Network	If distributed query (e.g. cloud DB)	~2–10 W	Guo, *et al*., 2022; Legler, *et al.,* 2025; Katal *et al.,* 2022
Container overhead	Scheduling, runtime, orchestration overhead	~5–10 W	Katal, *et al*., 2022; Centofanti, *et al.,* 2024
TOTAL		~27–120 W (median 73.5 W)	

It is assumed that a query runs between 10 and 20 seconds (He *et al*. 2024), and the mean 15 seconds is used**:**

Energy = 73.5 W × 15/3600 = 0.3 Wh

For many GenAI analytics queries, ~0.3 Wh is a reasonable average to allocate to query execution in the cloud.

5). Training

This entity of the cloud use is about the initial process where an LLM or GenAI model learns from massive datasets by adjusting its parameters over many cycles (epochs). An epoch is the time the model sees every training sample once. Training is assumed to be run on more optimized and newer hardware than e.g. the model interference.

Assumptions: Training tokens 300 billion (Brown *et al*. 2020), Epochs 3 (Prapas *et al*., 2021), interference tasks 30 billion (Schwartz *et al.,* 2020).

Training FLOPs: C×Parameters×Training tokens×Epochs = 6 × 39.6 billion × 300 billion × 3 = 2.14×10^{23} FLOPs
C = Architecture-specific constant for FLOPs/token, 6 (Hoffmann *et al.* 2022)
Assumptions for GPU: 1.2 TFLOPS/W (Khan *et al*. 2025) and power 700 W (Sun *et al.* 2021, Espenshade *et al*. 2024).
Time to execute those FLOPs: {700 W × 1.2×10^{12} FLOPS/W = 8.4×10^{14} FLOPS} 2.14×10^{23} FLOPs/8.4×10^{14} FLOPS = 2.54×10^{8} seconds
Compute energy used: (700 W×2.54×10^{8} seconds)/3600 = 49.5 MWh
Training energy per task: 49.5 MWh/30 billion = 1.65×10^{-3} Wh/task

The training energy is modeled as proportional to the number of model parameters, training tokens and epochs which is consistent with e.g. (Douwes and Serizel, 2024).

A. Code in GNU Octave for implementation and chart creation

The following code is used in GNU Octave (Park, 2021) to generate Figure 1 and Figure 2.

```
% genai_energy_minimal_with_values.m
% Minimal + robust: always saves PNGs, also shows figures if GUI works.
% Adds VALUE LABELS on BOTH plots.
clc; clear; close all;
outdir = fullfile(pwd,'out');
if ~exist(outdir,'dir'), mkdir(outdir); end
% ============================
% ENERGY MODEL
% ============================
EndUser = (15*60 + 8*120 + 2*10 + 3*30 + 5*40)/3600; % Wh
Network = 0.22 * 2; % Wh
parameters = 39.6e9; tokens = 1000;
GPU_power = 400; overhead = 50;
FLOPs = 2*parameters*tokens;
GPU_time = FLOPs/(GPU_power*1e12);
ModelInf = (GPU_power*GPU_time)/3600 * overhead; % Wh
API = (150*24 + 120*15 + 200*10 + 100*10 + 50*20)/3600; % Wh
MemOvh= (200*25)/3600; % Wh
Query = (73.5*15)/3600; % Wh

training_FLOPs = 6*parameters*300e9*3;
training_perf = 700*1.2*1e12;
training_time = training_FLOPs/training_perf;
TrainTask = ((700*training_time)/3600) / 30e9; % Wh/task
Cloud = ModelInf + API + MemOvh + Query + TrainTask;
Total = EndUser + Network + Cloud;

% ============================
% PRINT
% ============================
fprintf('\n=== ENERGY PER GENAI TASK ===\n\n');
fprintf('End-user HW: %.4f Wh\n', EndUser);
fprintf('Network: %.4f Wh\n', Network);
fprintf('Model inference: %.4f Wh\n', ModelInf);
fprintf('API latency: %.4f Wh\n', API);
fprintf('Memory overhead: %.4f Wh\n', MemOvh);
fprintf('Query execution: %.4f Wh\n', Query);
fprintf('Training (task): %.2e Wh\n', TrainTask);
fprintf('TOTAL ENERGY: %.4f Wh\n\n', Total);

% ============================
% PLOT 1 (breakdown) + VALUES + SAVE
% ============================
labels1 = {'End-user HW','Network','Model inference','API latency','Memory overhead','Query execution','Training'};
vals1 = [EndUser, Network, ModelInf, API, MemOvh, Query, TrainTask];

figure('Color','w','Position',[100 100 1200 600]);
barh(vals1); grid on;
ax = gca;
set(ax,'YDir','reverse','YTick',1:numel(labels1),'YTickLabel',labels1,'FontSize',25);
xlabel('Energy per task (Wh)'); title('Energy per GenAI task');

xmax = max(vals1);
xoff = 0.02*(xmax + eps);
for i=1:numel(vals1)
if i == numel(vals1)
t = sprintf('%.2e', vals1(i));
else
t = sprintf('%.4f', vals1(i));
end
text(vals1(i)+xoff, i, t, 'VerticalAlignment','middle','FontSize',25);
end
xlim([0, xmax*1.25 + eps]);

drawnow;
print(fullfile(outdir,'genai_energy_breakdown.png'),'-dpng','-r200');

% ============================
% SENSITIVITY (Top 10 absolute elasticities)
% ============================
delta = 0.01;
baseE = Total;

P = {
'tokens', tokens
'parameters', parameters
```

```
'GPU_power', GPU_power
'overhead', overhead
'Wh_per_MB', 0.22
'MB_per_task', 2
'cont_time', 24
'context_time', 10
'mem_ovh_time', 25
'query_time', 15
'training_tokens', 300e9
'inference_tasks', 30e9
};

S = zeros(size(P,1),1);
lab2 = cell(size(P,1),1);

for i=1:size(P,1)
name = P{i,1};
x0 = P{i,2};
x1 = x0*(1+delta);

tok=tokens; par=parameters; gp=GPU_power; ov=overhead;
whmb=0.22; mb=2; ct=24; cxt=10; mot=25; qt=15; ttok=300e9; it=30e9;

Switch name
Case 'tokens', tok=x1;
Case 'parameters', par=x1;
Case 'GPU_power', gp=x1;
Case 'overhead', ov=x1;
Case 'Wh_per_MB', whmb=x1;
Case 'MB_per_task', mb=x1;
Case 'cont_time', ct=x1;
Case 'context_time', cxt=x1;
Case 'mem_ovh_time', mot=x1;
Case 'query_time', qt=x1;
Case 'training_tokens', ttok=x1;
Case 'inference_tasks', it=x1;
End

Network2 = whmb*mb;
FLOPs2 = 2*par*tok;
GPU_time2 = FLOPs2/(gp*1e12);
ModelInf2 = (gp*GPU_time2)/3600 * ov;

API2 = (150*ct + 120*15 + 200*cxt + 100*10 + 50*20)/3600;
MemOvh2= (200*mot)/3600;
Query2 = (73.5*qt)/3600;

training_FLOPs2 = 6*par*ttok*3;
training_perf2 = 700*1.2*1e12;
training_time2 = training_FLOPs2/training_perf2;
TrainTask2 = ((700*training_time2)/3600)/it;
Total2 = EndUser + Network2 + (ModelInf2 + API2 + MemOvh2 + Query2 + TrainTask2);
S(i) = abs(((Total2-baseE)/baseE)/delta);
lab2{i} = strrep(name,'_',' ');
end
 [~,ord] = sort(S,'descend');
k = min(10,numel(S));
topS = S(ord(1:k));
topL = lab2(ord(1:k));

% ============================
% PLOT 2 (sensitivity) + VALUES + SAVE
% ============================
figure('Color','w','Position',[120 120 1300 650]);
barh(topS); grid on;
ax = gca;
set(ax,'YDir','reverse','YTick',1:k,'YTickLabel',topL,'FontSize',25);
xlabel('|(ΔE/E)/(Δx/x)|'); title('Sensitivity (Top inputs, absolute)');

xmax = max(topS);
xoff = 0.02*(xmax + eps);
for i=1:k
t = sprintf('%.3g', topS(i));
text(topS(i)+xoff, i, t, 'VerticalAlignment','middle','FontSize',25);
end
xlim([0, xmax*1.25 + eps]);

drawnow;
print(fullfile(outdir,'genai_sensitivity_top10.png'),'-dpng','-r200');

fprintf('Saved PNGs in: %s\n', outdir);
```

RESULTS AND DISCUSSION

As shown in Figure 1 the API/LLM Latency component is the largest share of the present task. Usually interference and training get a lot of attention when AI energy is discussed.

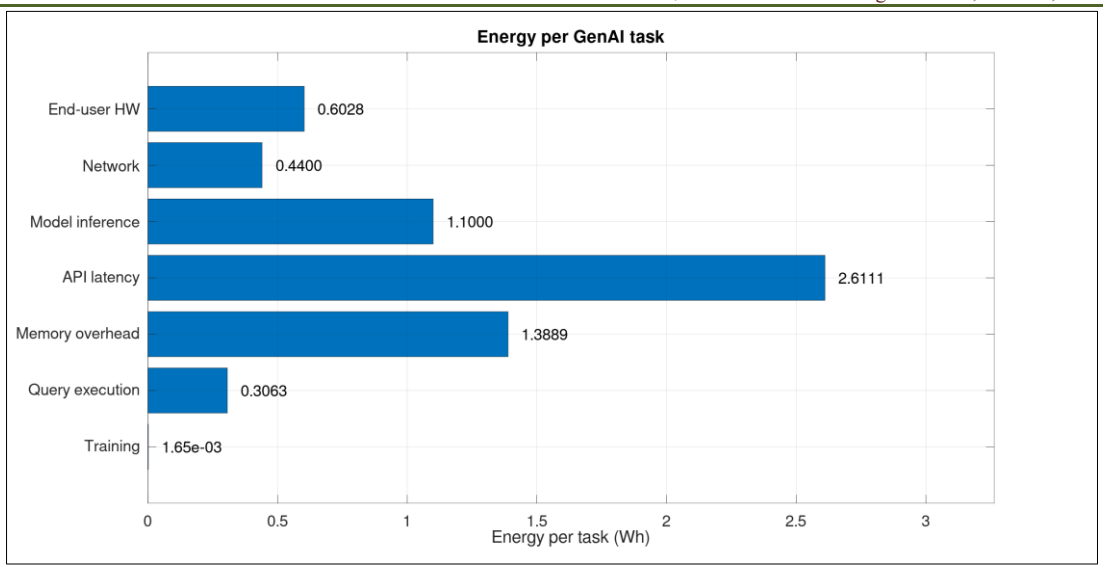

Figure 1: Energy use per GenAI task by component.

In total ≈6.45 Wh per functional unit is used.

Figure 1 is adequately consistent with (Figure 2a for GWP in Berthelot *et al.*, 2024) regarding the relative shares of the end-user, network and the cloud entities. This suggests that the proposed simplified framework is a useful development of state-of-the-art.

Figure 2 shows the result of the sensitivity analysis.

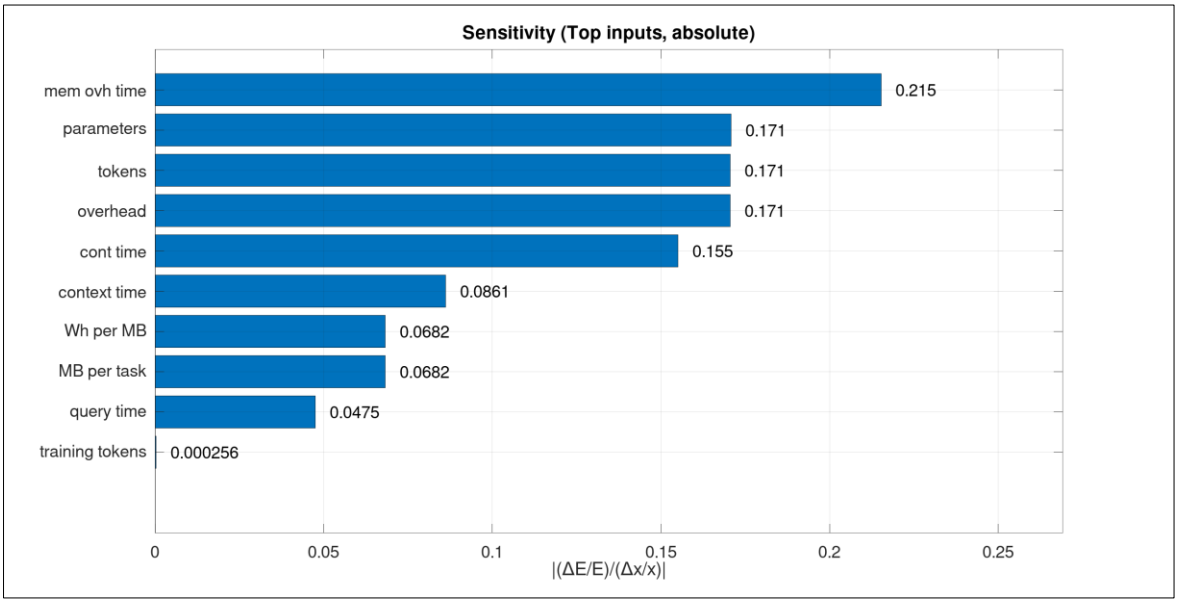

Figure 2: Sensitivity of Energy use per GenAI task by component

The sensitivity analysis is focused on cloud-side factors which are directly related to model architecture. Hence, end-user device energy is driven mainly by user behavior and device features and is therefore excluded from the input sensitivity ranking. The time assumed for memory overhead (25 s) being the most sensitive factor could be an over-simplification as it does not consider how GPUs actually schedule memory. Next follows the 39.6 billion parameters used both in model interference and training and the number of parameters is therefore and structural driver. The final result is also sensitive for the 1000 tokens assumption for model interference. After this the container standby time (24 s active per request) and prompt context preload time (10 s per request) in the API LLM segment affect the 6.45 Wh/task the most.

Can tasks be put into a larger context? How many are performed per year? In 2023 total AI share of data center electricity use was still limited (Shehabi *et al.*, 2024). Most installed servers were non-accelerated and storage and networking were dominated by non-AI traffic. For AI, non-AI dominated over GenAI. Due to the lack of data, a scenario-based allocation of the 176 TWh (approximate data center electricity use in the

United States in 2023), the dominance of non-accelerated servers, and the scale of traditional workloads relative to GenAI, it is assumed as in Table 6.

Table 6: Data Center tasks and electricity reasonable average intensities in 2024 to achieve total TWh

Task type in data centers	Tasks (trillions)	Wh/task, average (entire data center including overhead)	TWh	Share
non-GenAI	8.33	1.75	15	9%
GenAI	0.278	18	5	3%
Other DC (cloud computing, crypto, traditional workloads)	173	0.9	156	88%
			176	

Table 6 suggests that the numbers of tasks are counted in trillions and the table may be used to extrapolate and estimate future electricity demands of data centers and beyond.

The present investigation offers a preliminary signal, but the low number of datapoints precludes firm conclusions about absolute Wh/task values. Anyway, the analysis code can be reused for further modeling.

The result ≈6.45 Wh/task is comparable to (Berthelot *et al*., 2024) which presented a GenAI LCA of picture generation. Berthelot *et al*., 2024, looking at GenAI and LCA, found that a person visiting the website and submitting a prompt - generating four images - caused 7.84 g CO2e for the task, and when using 0.5 gCO2e/Wh, ≈15.6 Wh/task. Another source for GenAI benchmarks (Table 1 in Desroches *et al*., 2025) mentions 0.093 Wh/interference task (Low Model size and Chat use case) to 95.8 Wh/interference task (High Model size and Agents use case). These numbers do not refer to the electricity consumption across the whole system caused by one GenAI interaction. The present method accounts for time-extended service-level overheads associated with serving a user request.

CONCLUSION

For the first time, an Octave implementation for simplified GenAI task energy estimation is developed capturing the temporal structure of real user interactions. The implementation is applied to a limited application suggesting that further research with larger samples is required to substantiate the conclusion that time-dominated overheads can outweigh compute-dominated phases. The implementation works well. Task based functional units seem most appropriate for specific AI LCA case studies. However, even though the Wh/task seems reasonable, more studies including more GenAI task types are necessary to clearly establish the driving forces of energy use of individual GenAI tasks and their overall energy use.

REFERENCES

- Andersson, D., Grandin, P. (2025). Energy Consumption of GraphQL APIs: Analyzing the Impact of Optimization Techniques, Workload & Overfetching. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1970088
- Andrae, A. (2024b). Towards hundred thousand-fold improvement in energy performance for the coming ronnabyte era? *International Journal of Advanced Research in Engineering & Management (IJAREM)*, *10*(4), 1–13. http://www.ijarem.org/papers/v10-i4/1.IJAREMG7320.pdf
- Andrae, A.S.G. (2020). New perspectives on internet electricity use in 2030. *Engineering and Applied Science Letter*, 3(2), 19-31. DOI: 10.30538/psrp-easl2020.0038
- Andrae, A.S.G. (2023). From an Environmental Viewpoint Large ICT Networks Infrastructure Equipment must not be Reused. *WSEAS Transactions on Environment and Development*, *19*, 375–382. DOI: 10.37394/232015.2023.19.34
- Andrae, A.S.G. (2024a). Method for calculating the uncertainty range of avoided primary energy consumption and environmental impact applied to data analysis software services and solar electricity. *International Journal of Environmental Engineering and Development*, DOI: 10.37394/232033.2024.2.25
- Argerich, M. F., & Patiño-Martínez, M. (2024). Measuring and improving the energy efficiency of large language models inference. *IEEE Access*, *12*, 80194-80207. DOI: 10.1109/ACCESS.2024.3409745
- Berthelot, A., Caron, E., Jay, M., & Lefèvre, L. (2024). Estimating the environmental impact of Generative-AI services using an LCA-based methodology. *Procedia CIRP*, *122*, 707-712. DOI: 10.1016/j.procir.2024.01.098
- Bian, S., Yan, M., Jayarajan, A., Pekhimenko, G., & Venkataraman, S. (2025). What Limits Agentic Systems Efficiency?. *arXiv preprint arXiv:2510.16276*.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901. DOI: 10.5555/3495724.3495886

- Cabaret, L., Hudelot, C., Pierrard, R., & Poli, J. P. (2025). Efficient Parallel Fuzzy Dilation for Visual Reasoning on Edge: Leveraging ARM. In *Architecture of Computing Systems: 38th International Conference, ARCS 2025, Kiel, Germany, April 22–24, 2025, Proceedings* (p. 18). Springer Nature. DOI: https://doi.org/10.1007/978-3-032-03281-2_2
- Caiazza, C., Luconi, V., & Vecchio, A. (2024). Energy consumption of smartphones and IoT devices when using different versions of the HTTP protocol. *Pervasive and Mobile Computing*, *97*, 101871. DOI: 10.1016/j.pmcj.2024.101871
- Centofanti, C., Santos, J., Gudepu, V., & Kondepu, K. (2024). Impact of power consumption in containerized clouds: A comprehensive analysis of open-source power measurement tools. *Computer Networks*, *245*, 110371. DOI: 10.1016/j.comnet.2024.110371
- Choochotkaew, S., Wang, C., Chen, H., Chiba, T., Amaral, M., Lee, E. K., & Eilam, T. (2024). A Robust Power Model Training Framework for Cloud Native Runtime Energy Metric Exporter. *arXiv preprint arXiv:2407.00878*.
- Desroches, C., Chauvin, M., Ladan, L., Vateau, C., Gosset, S., & Cordier, P. (2025). Exploring the sustainable scaling of AI dilemma: A projective study of corporations' AI environmental impacts. *arXiv preprint arXiv:2501.14334*.
- Dornauer, B., & Felderer, M. (2023). Energy-saving strategies for mobile web apps and their measurement: Results from a decade of research. In *2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft)* (pp. 75-86). IEEE. DOI: 10.1109/MOBILESoft55845.2023.00014
- Douwes, C., & Serizel, R. (2024). From computation to consumption: Exploring the compute-energy link for training and testing neural networks for sed systems. *arXiv preprint arXiv:2409.05080*.
- Espenshade, C., Peng, R., Hong, E., Calman, M., Zhu, Y., Parida, P., ... & Kim, M. A. (2024, April). Characterizing training performance and energy for foundation models and image classifiers on multi-instance GPUs. In *Proceedings of the 4th Workshop on Machine Learning and Systems* (pp. 47-55). DOI: 10.1145/3634265.3634312
- Gonzalez-Agirre, A., Pàmies, M., Llop, J., Baucells, I., Da Dalt, S., Tamayo, D., ... & Villegas, M. (2025). Salamandra technical report. *arXiv preprint arXiv:2502.08489*.
- Gregersen, T., Patel, P., & Choukse, E. (2024). Input-dependent power usage in gpus. In SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1872-1877). IEEE. DOI: 10.1109/SC24-W58165.2024.00199
- Guennebaud, G., & Bugeau, A. (2024). Energy consumption of data transfer: Intensity indicators versus absolute estimates. *Journal of Industrial Ecology*, *28*(4), 996-1008. DOI: 10.1111/jiec.13499
- Guo, B., Yu, J., Yang, D., Leng, H., & Liao, B. (2022). Energy-efficient database systems: A systematic survey. *ACM Computing Surveys*, *55*(6), 1-53. DOI: 10.1145/3502958
- Hammad, Y., Ahmad, A. A. S., & Andras, P. (2025). An empirical study on the performance overhead of code instrumentation in containerised microservices. *Journal of Systems and Software*, 112573. DOI: 10.1016/j.jss.2025.112573
- He, Z., Yu, J., Gu, T., & Yang, D. (2024). Query execution time estimation in graph databases based on graph neural networks. *Journal of King Saud University-Computer and Information Sciences*, *36*(4), 102018. DOI: 10.1016/j.jksuci.2024.03.014
- Hedderich, M. A., Wang, A., Zhao, R., Eichin, F., Fischer, J., & Plank, B. (2025). What's the Difference? Supporting Users in Identifying the Effects of Prompt and Model Changes Through Token Patterns. *arXiv preprint arXiv:2504.15815*.
- Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... & Sifre, L. (2022). Training compute-optimal large language models. arXiv preprint arXiv:2203.15556.
- Horn, R., Lahnaoui, A., Reinoso, E., Peng, S., Isakov, V., Islam, T., & Malavolta, I. (2023). Native vs web apps: Comparing the energy consumption and performance of android apps and their web counterparts. In 2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft) (pp. 44-54). IEEE. DOI: 10.1109/MOBILESoft55845.2023.00012
- Horner, N., & Azevedo, I. (2016). Power usage effectiveness in data centers: overloaded and underachieving. *The Electricity Journal*, *29*(4), 61-69. DOI: 10.1016/j.tej.2016.04.004
- Huang, S., Guo, H., Xia, P., Sun, H., Lu, C., Feng, Y., ... & Wang, C. (2025). Integrated device of luminescent solar concentrators and electrochromic supercapacitors for self-powered smart window and display. *Nature Communications*, *16*(1), 2085. DOI: 10.1038/s41467-025-10549-5
- Ikram, M. J., Abulnaja, O. A., Saleh, M. E., & Al-Hashimi, M. A. (2017). Measuring power and energy consumption of programs running on kepler GPUs. In *2017 Intl Conf on Advanced Control Circuits Systems (ACCS) Systems & 2017 Intl Conf on New Paradigms in Electronics & Information Technology (PEIT)* (pp. 18-25). IEEE. DOI: 10.1109/ACCS-PEIT.2017.8303038
- International Telecommunication Union. (2022). *ITU-T L.1318 (08/2022): Q factor: A fundamental metric expressing integrated circuit energy efficiency*. https://handle.itu.int/11.1002/1000/15027

- Ishengoma, F. (2025). Enhancing performance of E-Government information systems with SSD-based Hadoop mapreduce. *Scientific Reports*, *15*(1), 1-15. DOI: 10.1038/s41598-025-15854-y
- Jin, C., Bai, X., Yang, C., Mao, W., & Xu, X. (2020). A review of power consumption models of servers in data centers. *Applied Energy*, *265*, 114806. DOI: 10.1016/j.apenergy.2020.114806
- Kansal, A., Zhao, F., Liu, J., Kothari, N., & Bhattacharya, A. A. (2010). Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing* (pp. 39-50). DOI: 10.1145/1807128.1807135
- Katal, A., Dahiya, S., & Choudhury, T. (2023). Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing*, *26*(3), 1845-1875. DOI: 10.1007/s10586-023-03685-0
- Khan, S., Naz, N. S., Mazhar, T., Tariq, M. U., Shahzad, T., Guizani, S., & Hamam, H. (2025). Green AI Techniques for Reducing Energy Consumption in AI Systems. *Array*, 100652. DOI: 10.1016/j.array.2025.100652
- Koneva, N., Navarro, A. L. G., Sánchez-Macián, A., Hernández, J. A., Zukerman, M., & de Dios, Ó. G. (2025). Introducing Large Language Models as the Next Challenging Internet Traffic Source. *arXiv preprint arXiv:2504.10688*.
- Legler, J., Werner, S., Borges, M. C., & Tai, S. (2025). Service-Level Energy Modeling and Experimentation for Cloud-Native Microservices. *arXiv preprint arXiv:2510.13447*.
- Mukherjee, D., Sandur, A., Mechitov, K., Lahiri, P., & Agha, G. (2024). eScope: A Fine-Grained Power Prediction Mechanism for Mobile Applications. *arXiv preprint arXiv:2405.08819*.
- Nõu, A., Talluri, S., Iosup, A., & Bonetta, D. (2025). Investigating Performance Overhead of Distributed Tracing in Microservices and Serverless Systems. In *Companion of the 16th ACM/SPEC International Conference on Performance Engineering* (pp. 162-166). DOI: 10.1145/3622028.3622563
- Park, Y. (2021). An automatic program of generation of equation of motion and dynamic analysis for multi-body mechanical system using GNU octave. *Journal of Applied and Computational Mechanics*, *7*(3), 1687–1697. DOI: 10.22055/jacm.2021.19251.2347
- Perez-Ramirez, D. F., Kostic, D., & Boman, M. (2025). CASTILLO: Characterizing Response Length Distributions of Large Language Models. *arXiv preprint arXiv:2505.16881*.
- Prapas, I., Derakhshan, B., Mahdiraji, A. R., & Markl, V. (2021). Continuous training and deployment of deep learning models. *Datenbank-Spektrum*, *21*(3), 203-212. DOI: 10.1007/s13222-021-00402-4
- Raza, S. M., Jeong, J., Kim, M., Kang, B., & Choo, H. (2021). Empirical performance and energy consumption evaluation of container solutions on resource constrained IoT gateways. *Sensors*, *21*(4), 1378. DOI: 10.3390/s21041378
- Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green ai. *Communications of the ACM*, *63*(12), 54-63. DOI: 10.1145/3385128
- Shehabi, A., et al. (2024). *2024 United States data center energy usage report***.** Lawrence Berkeley National Laboratory. DOI: https://doi.org/10.71468/P1WC7Q
- Sun, Y., Ou, Z., Chen, J., Qi, X., Guo, Y., Cai, S., & Yan, X. (2021). Evaluating performance, power and energy of deep neural networks on CPUs and GPUs. In *National conference of theoretical computer science* (pp. 196-221). Singapore: Springer Singapore. DOI: 10.1007/978-981-16-5673-5_12
- Yoon, I., Mun, J., & Min, K. S. (2025). Comparative Study on Energy Consumption of Neural Networks by Scaling of Weight-Memory Energy Versus Computing Energy for Implementing Low-Power Edge Intelligence. *Electronics*, *14*(13), 2718. DOI: 10.3390/electronics14132718.