Proceedings of CONF-CDS 2026 Symposium: Data-Centric AI Security: Securing Models, Learning Agents, and Autonomous Systems
DOI: 10.54254/2755-2721/2026.CH33910

Energy Consumption Measurement and Optimization Strategies for Large Language Model Systems

Jiaran Yun

School of Physical Science and Engineering, Tongji University, Shanghai, China
2351060@tongji.edu.cn

Abstract. Large language models (LLMs) have rapidly increased in scale in recent years, leading to substantial growth in computational demand and energy consumption. This paper investigates the energy consumption mechanisms of large language model systems and analyses the key factors influencing their energy requirements. A review of existing work on AI energy use and carbon footprint estimates is first presented. Next an LLM Energy Consumption Measurement Framework is presented with total energy usage broken down into Computation, Infrastructure Overhead, and Networking Energy Usage Components. After defining the measurement framework, several components that impact energy consumption are explored (such as model size/complexity, duration of training, hardware architecture utilized, and algorithmic efficiency), which demonstrate that the rapid increase in model parameters and training workloads will continue to have a very large impact on Computational Energy Consumption. At the same time, hardware design and algorithmic optimisation play important roles in improving energy efficiency. Finally, potential optimisation strategies are discussed, including model compression, efficient training techniques, specialised AI accelerators, and the integration of renewable energy in data centre operations. These findings provide a systematic perspective for understanding and improving the energy efficiency of large language model systems.

Keywords: Energy consumption, Large language models, AI systems, Data centre efficiency, Model optimisation

1. Introduction

In recent years, artificial intelligence technologies, particularly large language models (LLMs), have exhibited a pronounced trend towards scaling. Model parameter counts have rapidly expanded from millions and hundreds of millions to tens or even hundreds of billions. For instance, GPT-3 contains 175 billion parameters and requires large-scale GPU clusters with substantial computational resources over extended training periods [1]. Empirical studies further show that language model performance scales with a power law with respect to computational budget, dataset size, and parameter scale [2].

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.CH33910

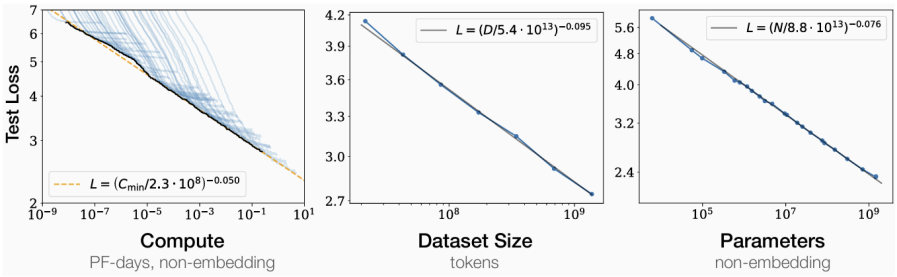


Figure 1. Scaling laws of large language models showing the relationship between test loss, compute, dataset size, and parameter scale [2]

As illustrated in Figure 1, test loss decreases consistently with increasing compute, dataset size, and model parameters, indicating that capability improvements are closely tied to growing computational demand. From an energy perspective, training large-scale models often requires weeks of continuous high-load GPU computation. The electricity consumption of training GPT-3 under best-practice data-centre conditions was around 1,287 MWh [1]. In addition, global data-centre electricity use now accounts for approximately 1%-1.5% of worldwide electricity consumption, and AI workloads are expected to be an important driver of future growth.

Despite growing concern over this issue, current research on AI energy consumption remains fragmented. Existing studies often estimate training energy use from GPU power and runtime, but such approaches may not fully account for cooling systems, power conversion losses, or network transmission. Differences in system boundaries and measurement methods also reduce comparability across studies. It remains unclear how model scale, training duration, hardware architecture, and algorithmic design jointly affect total energy consumption.

To address these issues, a systematic framework is developed to analyse energy consumption in large language model systems. Total energy consumption is divided into three categories: computational energy, infrastructural overhead, and network-related consumption. This study examines the main influencing factors and optimisation strategies for improving energy efficiency.

2. Research progress on energy consumption of AI systems

2.1. Energy consumption of deep learning

The energy consumption of deep learning systems can be analysed through the relationships among computational load, power consumption, and execution time. During training, the dominant cost comes from large-scale matrix multiplication and attention operations. Total computational workload is typically characterised by floating-point operations (FLOPs) and scales with both model size and the number of training tokens. Accordingly, the total energy consumption can be approximated as

$$E \approx P \times t \tag{1}$$

where P denotes the average power consumption and t represents the effective training duration, which is further determined by throughput and the number of training steps.

Data centres have traditionally accounted for about 1% of global energy use, and this proportion has remained fairly stable over the past decade. However, the continued rapid growth of AI-related

DOI: 10.54254/2755-2721/2026.CH33910

workloads may change this trend [3]. The metrics used to measure hardware load in data centres do not always accurately reflect the effective processing power actually being utilised.

Tools such as NVML report kernel occupancy, but cannot distinguish compute-bound from memory-bandwidth-bound conditions. Thus, model FLOPs utilisation (MFU, Achieved/Peak FLOPs) is used to characterise computational efficiency. Measurements show that when MFU decreases by about 30%, GPU power consumption falls by less than 10% [4], indicating significant static power and memory constraints. Under these conditions, energy consumption is not strictly linear with model scale, and operator structure or parallelisation may cause substantial variation.

2.2. Carbon footprint of AI systems

The carbon footprint of AI systems depends not only on computational energy consumption but also on infrastructure-related overhead. In data-centre environments, additional electricity use from cooling, power distribution, and auxiliary facilities is characterised by Power Usage Effectiveness (PUE), which relates IT energy use to total facility energy use. As a result, total operational energy consumption depends on both computational load and data-centre efficiency.

Carbon emissions are typically estimated by combining total electricity consumption with the carbon intensity (CI) of the regional power grid. Since CI varies significantly across regions, identical computational workloads may produce very different emissions. At the macro level, data centres already account for about 1%–1.5% of global electricity consumption, and the share linked to AI workloads continues to grow. This highlights the need to consider both infrastructure efficiency and energy sources when evaluating AI's environmental impact.

2.3. Green AI and sustainable AI

Green AI stresses the value of considering both computational cost and resource use in model evaluation, rather than focusing only on performance. Instead of pursuing the highest accuracy, it promotes acceptable performance under constrained budgets to improve energy efficiency.

Typical approaches include model compression, low-precision computation, and efficient training strategies, which reduce computational load or improve hardware utilisation. However, such gains may encourage wider deployment, partly offsetting energy savings. Model performance also shows diminishing returns. Case studies further indicate that architectural and algorithmic optimisation can improve the relationship between model scale and energy consumption [3].

Nevertheless, existing research mainly focuses on individual optimisation techniques or localised measurements, and still lacks a unified description of energy consumption across the full training process. This limitation motivates the need for a systematic framework to analyse energy consumption and its influencing factors, which is addressed in the following section.

3. Energy consumption measurement framework

3.1. Energy consumption model

To avoid systematic underestimation from approximations based solely on 'GPU power consumption × time', a theoretical expression may be established to decompose LLM training (or inference) energy consumption into three components: IT-side computational energy, data-centre infrastructure energy, and communication energy:

$$E_{total} = E_{IT} + E_{cool} + E_{net} \tag{2}$$

DOI: 10.54254/2755-2721/2026.CH33910

The term 'IT' encompasses accelerators (GPUs/TPUs), host CPUs, memory, and other devices directly involved in computation. Existing approaches often directly convert IT energy consumption into total facility-level electricity consumption using PUE:

$$\mathrm{E_{facility}} \approx \mathrm{PUE} \cdot \mathrm{E_{IT}} \tag{3}$$

Accordingly, an expression for operational energy consumption is derived (e.g., by integrating phased power and multiplying by PUE) [4]. The carbon footprint depends not only on model scale and service efficiency, but also on data centre performance metrics (such as PUE).

If only GPU local power consumption is considered, while neglecting additional power use from the CPU, memory, and cooling systems, the results reflect only the compute node's energy requirements. Such estimates cannot adequately represent total energy consumption at the data-centre level. Therefore, system boundaries must be clearly defined, with reasonable corrections for non-computational energy use. This paper uses 'PUE' to account for$E_{cool}$and retainsE_{net}as an independent term to characterise interconnect and data transmission overheads. For carbon accounting, it further incorporates the Carbon Intensity (CI) of the power grid using the formula [4]:

$$\mathrm{C} = \mathrm{E_{op}} \cdot \mathrm{CI} \tag{4}$$

3.2. GPU-based power estimation

During the training phase, the energy consumption on the GPU side may be regarded as the integral of power over time. Let the instantaneous power of the k-th GPU be denoted as$\mathrm{P_k(t)}$. The total energy may then be expressed as:

$$\mathrm{E} = \sum_{\mathrm{k}} \int_{\mathrm{t_0}}^{\mathrm{t_0}} \mathrm{P_k}\,(\mathrm{t})\mathrm{dt} \tag{5}$$

If discrete sampling data is employed, it may be approximated as

$$\mathrm{E} \approx \sum_{\mathrm{k}} \overline{\mathrm{P}}_{\mathrm{k}} \cdot \mathrm{T} \tag{6}$$

Where$\mathrm{T} = \mathrm{t_1} - \mathrm{t_0}$, and $\overline{\mathrm{P}}_{\mathrm{k}}$denotes the time-averaged power within the training interval. To ensure the average value has physical significance, the statistical interval should be clearly defined, and time-weighting should be used to eliminate bias caused by uneven sampling intervals.

$$\overline{\mathrm{P}} = \frac{\sum_{\mathrm{i}} \mathrm{P_i} \Delta \mathrm{t_i}}{\sum_{\mathrm{i}} \Delta \mathrm{t_i}} \tag{7}$$

Applying time-weighting to power signals during resampling or aggregation constitutes a necessary step [4].

Power data is typically derived from GPU board-level telemetry, such as NVML board power. However, utilisation metrics do not directly reflect effective computation, and power consumption is not linearly related to utilisation. When computational resource usage is measured by MFU (Model FLOPs Utilisation), a 30% decrease in MFU results in less than a 10% reduction in power consumption [4]. This indicates significant static power baselines and memory access bottlenecks. Therefore, treating power simply as TDP may lead to phase-specific errors in power estimation.

DOI: 10.54254/2755-2721/2026.CH33910

Statistics solely based on GPU-side power integration:

$$E = \sum_k \int P_{k(t)} dt \tag{8}$$

Excludes ancillary components such as the CPU, memory, interconnects, and cooling systems. This approach facilitates comparative analysis across identical hardware platforms, though the results should not be directly equated with the total energy consumption at the data centre level.

3.3. Data center infrastructure energy correction

Merely calculating GPU power consumption is insufficient to reflect total energy expenditure in training. In real deployment environments, infrastructure such as cooling and power distribution adds energy costs. Therefore, GPU-side energy consumption should be adjusted using data-centre efficiency metrics. An approach is to introduce PUE (Power Usage Effectiveness), defined as:

$$PUE = \frac{E_{total}}{E_{IT}} \tag{9}$$

E_total denotes the overall energy consumption of the data centre facility, whileE_{IT}represents the energy consumed by IT equipment such as servers, storage, and network devices [5].

Therefore, having obtained the GPU training energy consumption E_GPU, the infrastructure energy consumption may be estimated using the following equation:

$$E_{total} \approx PUE \times E_{GPU} \tag{10}$$

This methodology incorporates facility loads, such as cooling, UPS systems, distribution losses, and lighting, into the statistical scope. Many traditional data centres have a PUE of 1.3–3.0 [5], showing that non-IT infrastructure accounts for a significant share of energy use.

PUE describes infrastructure efficiency rather than computational productivity. In training energy estimators, it is better used as an infrastructure-level correction factor to extrapolate equipment-side energy to facility-level energy use.

4. Key influencing factors

4.1. Model size(parameter scale)

The model size N exhibits a stable power-law relationship with performance. Under conditions of sufficient data and training to convergence, the test loss can be expressed as

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N} \tag{11}$$

Where$\alpha_N \approx 0.076$, $N_c \approx 8.8 \times 10^{13}$ [2]. This relationship indicates that increasing model parameters improves performance with diminishing returns.

The computational cost of training can be approximated as:

$$C \approx 6ND \tag{12}$$

DOI: 10.54254/2755-2721/2026.CH33910

Where D denotes the number of training tokens [2]. With data volume remaining constant, linear parameter growth will directly inflate total FLOPs, thereby increasing energy consumption.

Under a fixed computational budget C, the optimal model size satisfies the following relationship with data volume:

$$\begin{cases} N_{\{opt\}} \propto C^{a} \\ D_{\{opt\}} \propto C^{b} \end{cases} \tag{13}$$

Where$a \approx 0.50$, $b \approx 0.50$ [6]. This implies that when computational power increases tenfold, model scale and training data should expand roughly proportionally. If parameters increase without enough data, the model deviates from the computationally optimal frontier. This reduces performance gains per unit of compute, leading to lower marginal energy efficiency.

4.2. Training duration

Training duration is reflected either in parameter updates per step or in the total token count, which directly determines computational volume. In data-constrained scenarios, the relationship between loss and data scale satisfies:

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D} \tag{14}$$

Where$\alpha_D \approx 0.095$, $D_c \approx 5.4 \times 10^{13}$tokens [2]. Furthermore, the total computational effort for training can be expressed as

$$C = 6NBS \tag{15}$$

Where S denotes the number of parameter update steps [2]. When hardware power remains approximately constant, energy expenditure is nearly proportional to training steps or token count.

Large-scale models are typically trained on10^{11} – 10^{12} tokens. For example, a 70-billion-parameter model is trained on hundreds of billions of tokens [6]. Insufficient or excessive training both reduce energy efficiency, either through underused compute or diminishing returns.

4.3. Hardware architecture

Hardware architecture directly affects energy consumption through differences in computational efficiency and system design. Large-scale training mainly relies on GPUs or TPUs. Dedicated accelerators improve performance per watt through optimised matrix units, memory hierarchy, and interconnects. TPU v4 achieves about 2.7× higher performance per watt [7].

Process technology affects energy efficiency by enabling lower energy per operation through finer fabrication nodes, but at the same time, increasing integration density. In distributed training systems, the interconnect bandwidth and the communication topology come into play affecting the overall energy consumed. Energy optimized network structures, like reconfigurable interconnects, can reduce communication overhead and increase system level energy efficiency [7].

DOI: 10.54254/2755-2721/2026.CH33910

4.4. Algorithm efficiency

Algorithmic efficiency determines the effective computational cost under fixed hardware conditions. Training workloads are dominated by matrix multiplication and convolution operations, which account for the majority of execution time [8].

Low-precision computation, such as mixed-precision training, reduces memory bandwidth requirements and improves computational throughput. Experiments show that mixed-precision training can improve both throughput and energy efficiency by over 20% [8].

In addition, techniques such as model sparsification and computational optimisation reduce effective FLOPs, thereby lowering energy consumption. These approaches improve energy efficiency primarily by reducing redundant computation or increasing hardware utilisation.

5. Optimisation strategies

5.1. Model compression

Model compression lowers computational cost while maintaining acceptable performance. Common approaches include pruning, quantisation, and distillation.

Pruning removes redundant weights or structures in neural networks, reducing both model size and computational overhead. In large convolutional models, parameter counts can be reduced by over 75%, with corresponding reductions in computational cost [9]. As model sizes continue to grow, pruning has become a key direction in model compression [10].

Quantisation reduces the numerical precision of model parameters by converting floating-point values to lower-bit representations, thereby reducing memory usage and computational cost. In this way, the model becomes more efficient while maintaining relatively stable overall performance.

Knowledge distillation transfers knowledge from a large model to a smaller one. By learning from the larger model's outputs or internal features, the smaller model can achieve good performance with fewer parameters. Comparative studies indicate that quantisation offers favourable efficiency gains, while distillation is effective in reducing inference latency [11].

5.2. Efficient training techniques

Efficient training reduces energy use through improved hardware utilisation and shorter training times. The most common approach is mixed-precision training, in which inference uses float16 and backpropagation uses float32 to avoid numerical instability. On GPUs with TensorCore support, mixed-precision training can accelerate training by 1.9× with minimal effect on accuracy [12].

Gradient checkpointing reduces memory consumption by recomputing activations during backpropagation, enabling larger models to be trained under limited hardware resources [13].

Batch size selection also affects computational efficiency. Larger batch sizes improve hardware parallelism and throughput, while excessively large batches may degrade convergence performance. In practice, training efficiency is determined by balancing throughput and optimisation stability.

5.3. Hardware optimisation

Hardware optimisation improves energy efficiency by enhancing performance-per-watt and reducing data movement overhead. Specialised accelerators integrate dedicated tensor computation units, high-bandwidth memory, and optimised dataflow architectures.

DOI: 10.54254/2755-2721/2026.CH33910

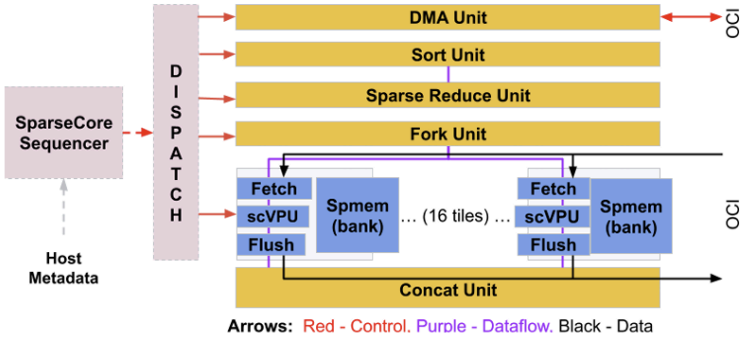


Figure 2. Architecture of a sparse-computing AI accelerator [7]

Figure 2 shows a sparse-computing accelerator architecture in which specialised functional modules improve parallel processing efficiency and reduce data movement overhead.

The TPU v4 achieves approximately 2.7× improvement in performance-per-watt in large-scale training scenarios [7].

Edge AI shifts computation closer to data sources. Edge devices operate under strict resource constraints (e.g., less than 1 MB storage and limited memory) [14], reducing data transmission and overall system energy consumption.

6. Conclusion

The rapid scaling of large language models has led to a noticeable increase in computational demand and energy consumption. This paper devises a decomposition that disentangles total energy use into computational energy, infrastructure overhead, and network-related energy, allowing for a fuller description of LLM system energy consumption behaviour. Using this decomposition, it is shown that model size, training duration, hardware architecture and algorithmic efficiency jointly determine total energy demand, with effects not strictly linear due to utilisation limits and system considerations.

From an optimisation perspective, improvements cannot rely on a single level. Model compression and efficient training methods help reduce computational load, while hardware design and specialised accelerators improve performance per watt. At the system level, data centre efficiency and energy sources also play an important role. Overall, improving energy efficiency requires coordinated optimisation across model design, hardware, and infrastructure, providing a practical direction for the sustainable development of large-scale AI systems.

References

[1] Patterson D, Gonzalez J, Le Q, et al. Carbon emissions and large neural network training [J]. arXiv preprint arXiv: 2104.10350, 2021.
[2] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models [J]. arXiv preprint arXiv: 2001.08361, 2020.
[3] De Vries A. The growing energy footprint of artificial intelligence [J]. Joule, 2023, 7(10): 2191-2194.
[4] Özcan M, Wiesner P, Weiß P, et al. Quantifying the energy consumption and carbon emissions of LLM inference via simulations [J]. arXiv preprint arXiv: 2507.11417, 2025.
[5] Avelar V, Azevedo D, French A, et al. PUE: a comprehensive examination of the metric [J]. White paper, 2012, 49: 52.
[6] Hoffmann J, Borgeaud S, Mensch A, et al. Training compute-optimal large language models [J]. arXiv preprint arXiv: 2203.15556, 2022, 10.

DOI: 10.54254/2755-2721/2026.CH33910

[7] Jouppi N, Kurian G, Li S, et al. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings [C]//Proceedings of the 50th annual international symposium on computer architecture. 2023: 1-14.
[8] He X, Sun J, Chen H, et al. Campo: {Cost-Aware} performance optimization for {Mixed-Precision} neural network training [C]//2022 USENIX Annual Technical Conference (USENIX ATC 22). 2022: 505-518.
[9] Cheng Y, Wang D, Zhou P, et al. A survey of model compression and acceleration for deep neural networks [J]. arXiv preprint arXiv: 1710.09282, 2017.
[10] Cheng H, Zhang M, Shi J Q. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46(12): 10558-10578.
[11] Von Rad J, Cao Y, Geiger A. UniComp: A Unified Evaluation of Large Language Model Compression via Pruning, Quantization and Distillation [J]. arXiv preprint arXiv: 2602.09130, 2026.
[12] Dörrich M, Fan M, Kist A M. Impact of mixed precision techniques on training and inference efficiency of deep neural networks [J]. IEEE Access, 2023, 11: 57627-57634.
[13] Yang Y. Large Language Models for Energy and Carbon Footprint Optimization: A Comprehensive Survey [J].
[14] Lamaakal I, Yahyati C, Ouahbi I, et al. A survey of model compression techniques for TinyML applications [C]//2025 International Conference on Circuit, Systems and Communication (ICCSC). IEEE, 2025: 1-6.