

Review

A Survey on Energy-Efficiency Mechanisms for Large Language Model Training: Measurement, Optimization Mechanisms, Evidence Boundaries, and Future Research Directions

Prudhvi Raj Nelapatla * , Elijah Starkey and Yi Zhou 

TSYS School of Computer Science, Columbus State University, Columbus, GA 31907, USA; starkey_elijah@students.columbusstate.edu (E.S.); zhou_yi@columbusstate.edu (Y.Z.)

* Correspondence: nelapatla_prudhviraj@students.columbusstate.edu

Abstract

The rapid growth of large language models (LLMs) has made training energy efficiency a major systems and sustainability challenge. Training requires substantial computation, memory, communication, and electrical energy, yet the literature often treats runtime, floating-point operations, memory, communication, cost, energy, and carbon as interchangeable indicators of efficiency. This survey critically reviews 54 research papers and one supporting software tool across measurement and carbon accounting, model and numerical efficiency, memory and communication, distributed planning, GPU power control, carbon-aware scheduling, lifecycle design, and reliability. We classify evidence as direct, modeled, simulated, reported-comparison, or enabling, and interpret results only within their stated workload, hardware, quality condition, and measurement boundary. The synthesis shows that individual mechanisms are mature, yet cross-paper comparison remains difficult because studies use inconsistent boundaries, quality targets, platforms, and accounting methods. We, therefore, propose a quality-aware reporting framework centered on energy to target quality, explicit boundaries, uncertainty, and reproducibility. The survey establishes an evidence-disciplined framework for evaluating energy-efficient LLM training and identifies integrated, quality-aware, whole-system energy optimization as the principal research direction.

Keywords: large language model training; energy efficiency; distributed training; power management; energy measurement; carbon-aware computing; quality-aware evaluation; sustainable AI



Academic Editor: George Angelos Papadopoulos

Received: 12 August 2026

Revised: 14 September 2026

Accepted: 18 September 2026

Published: 29 September 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

1.1. Motivation and Scope

The rapid growth of large language models has made training efficiency a major systems and sustainability concern because a single training program can occupy many accelerators for long periods while also consuming host-compute, memory, interconnect, storage, cooling, and datacenter electricity. Training compute for landmark machine-learning systems has grown by many orders of magnitude over the past decade, with an estimated doubling time of a few months in the recent large-scale era [1]. These resources create financial and environmental costs, but they are not represented by one interchangeable metric. A method can reduce runtime, floating-point operations, memory, or communication without necessarily reducing total energy.

Early work on the energy and policy implications of deep learning for natural language processing helped establish the need to report computational demand, financial cost, and environmental impact alongside model performance [2].

Research on LLM-training efficiency spans multiple layers. Model and data methods change the amount of learning work; mixed precision and efficient kernels change numerical execution; checkpointing and sharding alter memory and communication; parallelism planners divide work across devices; GPU controllers change power and frequency; carbon-aware schedulers shift workloads across time or location and reliability systems reduce lost work. This survey focuses specifically on how these mechanisms affect the energy efficiency of LLM training and how their evidence should be interpreted.

1.2. Related Surveys

To position the contribution of this survey, we first review representative prior survey efforts. Existing surveys provide valuable but complementary views of LLM efficiency. Hardware-oriented surveys review accelerator architectures and compare GPUs, TPUs, FPGAs, and domain-specific accelerators [3]. Distributed-training surveys focus on large-scale infrastructure, including parallelism, memory optimization, communication reduction, and fault tolerance [4]. Other surveys emphasize algorithmic efficiency through scaling laws, model compression, sparse architectures, parameter-efficient fine-tuning, and inference optimization [5–7].

Although these surveys provide important foundations, most are not organized around energy-efficient LLM training as the primary objective. Energy is often discussed alongside throughput, latency, FLOPs, or memory rather than treated as a distinct evaluation target. Measurement boundaries, quality conditions, carbon signals, and reliability costs also vary across studies, which complicates direct comparison. Table 1 summarizes representative related surveys and positions the narrower contribution of this work; the comparison dimensions are our synthesis of the cited surveys.

Table 1. Comparison of representative surveys related to efficient LLM systems and training.

Survey	Primary Scope	Training-Energy Treatment	Systems/Sustainability Coverage	Main Distinction Relative to This Survey
Hardware accelerator survey [3]	Accelerator architectures and reported speed/energy-efficiency comparisons	Partial; substantial inference emphasis and no unified training-energy benchmark	Distributed: limited; carbon: no; reliability: no	Hardware-centered; inconsistent baselines limit direct comparison
Distributed LLM training survey [4]	Infrastructure, parallelism, memory, communication, and fault tolerance	Limited; mostly utilization, throughput, memory, and communication	Distributed: strong; carbon: limited; reliability: strong	Closest systems survey, but not energy-centered
Compression and tuning survey [5]	Model compression (pruning, quantization, distillation, low-rank) and parameter-efficient tuning	Limited; mainly compression ratio, latency, and memory	Distributed: no; carbon: no; reliability: no	Compression- and tuning-centered, not training-energy-centered

Table 1. *Cont.*

Survey	Primary Scope	Training-Energy Treatment	Systems/Sustainability Coverage	Main Distinction Relative to This Survey
Efficient LLM survey [6]	Model-, data-, and framework-centric efficiency across the lifecycle	Limited; energy and carbon are not primary organizing dimensions	Distributed: strong; carbon: limited; reliability: limited	Broad taxonomy, but not energy-centered
Resource-efficient foundation-model survey [7]	Computation, memory, energy, cost, and communication across the lifecycle	Moderate; energy is included but evidence is largely secondary	Distributed: strong; carbon: partial; reliability: limited	Closest multi-resource framing, but broader than training energy
This survey	Training-energy measurement and mechanisms across model, memory, communication, planning, power, carbon, and reliability	Central; separates direct, modeled, simulated, reported-comparison, and enabling evidence	Distributed: strong; carbon: central; reliability: explicit	Evidence-centered comparison by measurement boundary, quality condition, and reproducibility

1.3. Contributions

This survey makes the following contributions:

1. It reviews 54 research papers and one separately audited software tool spanning energy measurement, model and numerical efficiency, memory, communication, distributed training, GPU power control, carbon-aware scheduling, and reliability.
2. It organizes the literature through a mechanism-oriented taxonomy rather than publication chronology.
3. It distinguishes direct, modeled, simulated, reported-comparison, and enabling evidence and preserves each result's workload, hardware, quality condition, and measurement boundary.
4. It compares mechanisms across common dimensions, including quality evidence, distributed scope, runtime adaptation, scalability, carbon accounting, reliability, and reproducibility.
5. It identifies recurring research gaps in measurement consistency, cross-paper comparability, mechanism integration, reliability, carbon accounting, and reproducibility.
6. It derives an energy-aware self-designing distributed training system as a long-term conceptual research direction rather than an implemented contribution of this paper.

1.4. Organization and Reading Guide

The manuscript intentionally progresses from foundations and review methodology, through mechanism-level evidence, toward cross-mechanism synthesis and future research directions. Section 2 introduces the LLM-training and distributed-systems concepts required for the survey. Section 3 describes the review methodology and evidence hierarchy. Section 4 defines metrics, evidence classes, and measurement boundaries. Sections 5–11 review energy measurement, model and numerical efficiency, memory and communication, distributed planning, GPU power control, carbon-aware scheduling, and reliability. Section 12 synthesizes evidence across mechanism categories. Section 13 identifies research gaps and develops the survey-derived agenda. Section 14 presents the future direction toward energy-aware self-designing distributed training. Sections 15–17 provide the recommended evaluation framework, threats to validity, and conclusion.

The overall organization of this survey is summarized in Figure 1.

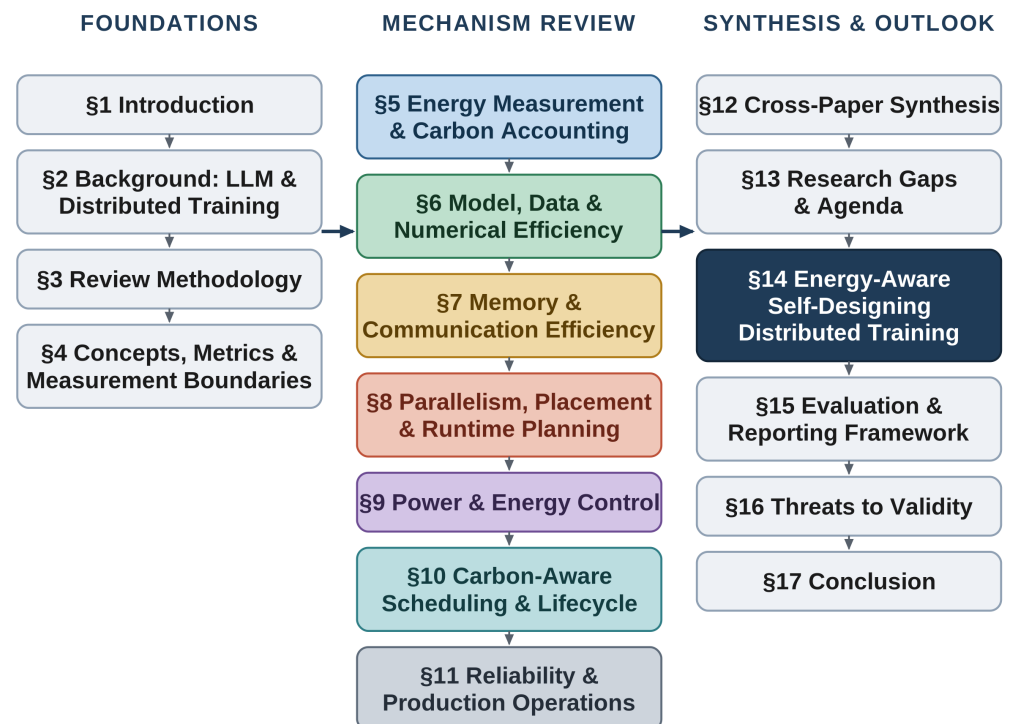


Figure 1. Organization of the survey across foundations, mechanism review, and synthesis and outlook. The roadmap aligns Sections 1–17 with the manuscript's evidence-centered progression.

2. Background: LLM and Distributed Training Fundamentals

2.1. LLM Training Process

LLM training repeatedly performs a forward pass, computes a loss, performs a backward pass, and updates model parameters. During the forward pass, each Transformer layer produces intermediate values called activations. During the backward pass, gradients quantify how each parameter should change to reduce the loss, and an optimizer applies those updates [8].

2.2. Quality Metrics

Energy can be compared fairly only when competing methods achieve comparable model quality. For language-model pretraining, common metrics include validation loss and perplexity; lower values generally indicate better next-token prediction. Task-specific studies may instead use accuracy, F1, exact match, pass rate, or benchmark scores. The preferred comparison in this survey is energy to a predeclared target quality, or a clearly justified non-inferiority margin when exact equality is unrealistic [9,10].

2.3. Distributed Training

When a model or batch is too large for one accelerator, training is distributed across multiple devices. Data parallelism replicates the model and divides data; tensor parallelism divides mathematical operations; pipeline parallelism assigns different layers to different stages; sequence and expert parallelism divide sequence or mixture-of-experts work. These strategies solve different scalability bottlenecks but introduce new system trade-offs, including synchronization, communication, idle time, and additional failure modes [4,11–13].

2.4. Power, Energy, and Carbon

Power is the instantaneous rate of electricity use, measured in watts. Energy accumulates power over time and is measured in joules, watt-hours, or kilowatt-hours. Carbon emissions are derived by combining energy with an electricity carbon-intensity signal. A lower power setting can still consume more energy if it increases runtime substantially, and lower operational carbon does not necessarily mean lower energy [14].

2.5. Why Mechanisms Must Be Compared Carefully

The mechanisms surveyed in this paper operate at different layers and report different outcomes. Reduced FLOPs, memory, communication, cost, or runtime are valuable enabling results, but they are not direct proof of energy savings unless energy is measured or estimated through a disclosed model. Similarly, quality, hardware generation, model scale, and measurement boundary must be aligned before results can be compared [15,16].

Figure 2 shows the LLM training lifecycle and the stage-specific and cross-cutting efficiency mechanisms reviewed in this survey.

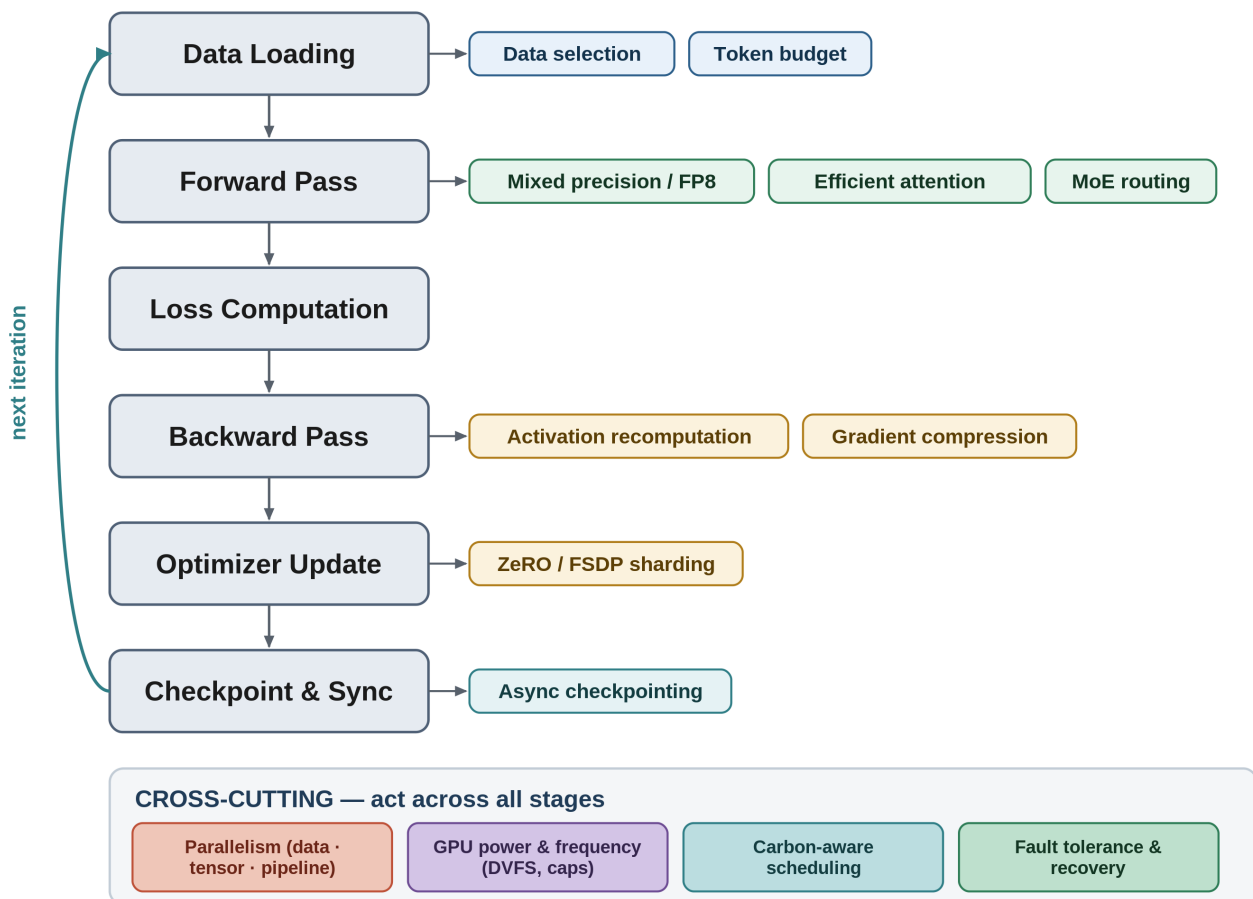


Figure 2. LLM training lifecycle and the stage-specific and cross-cutting efficiency mechanisms reviewed in this survey.

3. Review Methodology

This review was designed as an evidence-disciplined critical survey rather than as a completed PRISMA systematic review. Its objective is to catalog and compare energy-efficiency mechanisms while determining exactly what each study measures, which system boundary it covers, whether model quality is preserved, and whether its reported result constitutes direct energy evidence, modeled evidence, simulation, a reported comparison, or an enabling improvement. Reductions in floating-point operations, runtime, memory

footprint, communication volume, monetary cost, or accelerator utilization do not by themselves establish lower energy.

3.1. Research Questions

The review is organized around six research questions (RQ1–RQ6). They determine what was extracted from each study and how the evidence was synthesized, and each is answered in the sections indicated:

- RQ1.** Which mechanisms across the training stack affect the energy efficiency of LLM training, and how can they be organized into a coherent, mechanism-oriented taxonomy? (Sections 5–11)
- RQ2.** For each mechanism, what quantity does the primary study actually report—energy, power, or carbon, as opposed to runtime, FLOPs, memory, communication, or cost—and within which measurement boundary? (Section 4)
- RQ3.** What class of evidence—direct, modeled, simulated, reported-comparison, or enabling—supports each retained result? (Evidence hierarchy defined in this section and applied throughout)
- RQ4.** Is model quality preserved, and can the result be expressed as energy to a declared target quality rather than as an unnormalized saving? (Sections 2.2 and 4)
- RQ5.** How comparable are results across studies given differences in hardware, workload, measurement boundary, and carbon accounting, and what does a capability-level cross-paper synthesis reveal? (Section 12)
- RQ6.** Which research gaps remain, and what future research direction and evaluation standard follow from the synthesized evidence? (Sections 13–15)

3.2. Review Scope and Canonical Corpus

The canonical corpus contains 55 reviewed entries: 54 research papers and one supporting software tool, CodeCarbon (version 3.2.2). The original submission listed 60 entries. Two redundant arXiv preprints (P02 and P07) were withdrawn in the first revision. To remove all preprints, three further arXiv-only entries were withdrawn in the second revision, with their claims rebased on peer-reviewed sources, and the arXiv FP8-formats entry was replaced by a peer-reviewed FP8 training study. The reference list, therefore, contains no preprints. The papers span energy and carbon accounting, GPU power management, model and data scaling, numerical precision, activation memory, communication, distributed parallelism, scheduling, heterogeneous and geo-distributed training, carbon-aware computing, lifecycle design, reliability, hardware acceleration, and resource-efficiency surveys. CodeCarbon was retained as software evidence because it represents a widely used practical instrumentation layer, but it was not counted as a research paper. Each entry was assigned a stable identifier, P01 through P60, and tracked in a canonical registry to prevent duplicate counting, filename drift, and numbering changes during the review.

The corpus is intended to provide broad and technically deep coverage of mechanisms relevant to energy-efficient large-model training. Papers were selected because they contributed primary evidence or foundational mechanisms relevant to this objective, rather than because of publication date or popularity. The corpus is not claimed to be an exhaustive census of all publications in machine-learning efficiency. The scope was deliberately restricted to work that contributes at least one of the following: direct energy or carbon evidence; an optimization mechanism that could affect training energy; a measurement or accounting method; a distributed-training systems capability; production-scale reliability evidence; or a survey taxonomy necessary to identify the state of the field. Inference-only papers were retained only when they offered direct hardware or measurement evidence useful for defining system boundaries; their findings were not generalized to training.

To prevent over-generalization across regimes, every corpus entry is additionally stratified by training regime—LLM pretraining, LLM fine-tuning, generic deep-neural-network or machine-learning training, inference or hardware characterization, and contextual or survey evidence—and this stratum is recorded in the Supplementary Materials. Evidence from one stratum is not treated as direct proof for another: fine-tuning, generic-DNN, and inference results are read as transferable hypotheses whose applicability to large-model pretraining depends on scale, workload structure, and measurement boundary, and they are flagged as such wherever they inform a pretraining-oriented conclusion.

3.3. Documented Update Search and Screening

The original corpus was assembled by expert, snowball-style curation of primary mechanism papers rather than by an exhaustive database query, and a complete historical retrieval log was not kept. To make selection reproducible and to test the corpus for recent and competing work, a documented update search was executed on 25 August 2026 across three public bibliographic services with programmatic interfaces: Crossref, OpenAlex, and DBLP. Five predefined queries were used: (Q1) “energy efficient large language model training”; (Q2) “carbon aware large language model training”; (Q3) “GPU power energy large model training”; (Q4) “distributed large language model training energy” and (Q5) “metamaterial metasurface optical neural network energy”. For each query and service, the top 100 relevance-ranked records were retrieved and archived, producing 1002 raw records. Records were deduplicated by DOI where present, and otherwise by normalized title, leaving 789 unique records. A reproducible priority rule—peer-reviewed venue, publication year 2023–2026, and relevance of energy, power, or carbon to large-model training—selected 184 records for title and metadata screening, from which eight peer-reviewed items were retained after source verification (seven from the database search and one from citation chaining). The 100-record-per-query API cap and the automatic priority rule are disclosed limitations that can miss papers with atypical terminology; the raw responses, the deduplicated screening ledger, exclusion reasons, and inclusion decisions are provided in Supplementary File S1 so that the search can be reproduced and audited.

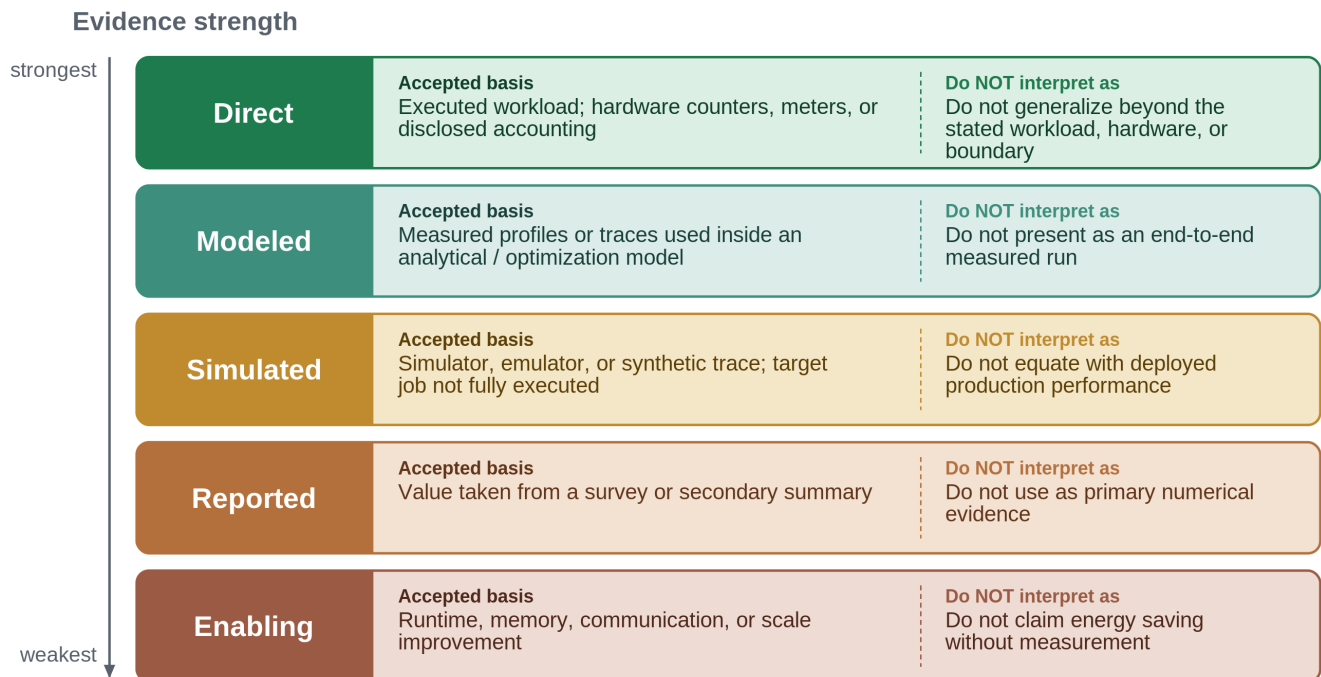
3.4. Evidence Hierarchy

Every retained claim was classified by evidence type. Direct evidence denotes energy, power, or carbon values obtained from an executed workload using hardware counters, external meters, datacenter telemetry, or an explicitly described accounting pipeline. Modeled evidence uses measured profiles or traces as inputs to an analytical or optimization model. Simulated evidence reports results from a simulator, emulator, or synthetic workload trace without executing the complete target training job. Reported-comparison evidence appears in a survey or secondary source and was not accepted as independently verified numerical proof. Enabling evidence includes improvements in runtime, memory, communication, scalability, or feasibility that may contribute to energy efficiency but do not directly measure energy.

3.5. Data Extraction and Structured Review

Each paper was reviewed using a common extraction schema. The schema recorded bibliographic metadata, study objective, workload, model scale, hardware and topology, optimization variables, comparison baseline, measurement method, system boundary, quality or convergence evidence, numerical results, limitations, and relevance to the six research questions (RQ1–RQ6) defined at the start of this section. Particular attention was paid to whether a result was measured or projected, whether the comparator used equal quality or equal completion time, and whether the reported energy covered only accelerators or the wider host, cluster, datacenter, and lifecycle boundary. Figure 3 presents

the evidence-strength framework used to distinguish direct, modeled, simulated, reported, and enabling claims.



Classes indicate how closely a claim is tied to an executed, measured training workload — not overall paper quality.

Figure 3. Evidence-strength framework used to distinguish direct, modeled, simulated, reported, and enabling claims and to constrain their interpretation.

For systems papers, the review also recorded whether the method was static or runtime-adaptive; which dimensions it controlled, such as data, tensor, pipeline, sequence, or expert parallelism; whether it changed batch size, learning rate, precision, recomputation, communication, placement, or power state and whether it considered failures, deadlines, carbon intensity, or final model quality. This structured extraction allowed comparisons to be based on decision scope and evidence strength rather than on headline speedup alone.

3.6. Claim-Level Verification and Second-Pass Audit

Numerical and technical claims were not retained from abstracts or surveys without checking the original source location. For every deeply reviewed entry, claims were tied to a page, section, table, figure, equation, or source-code path. Ambiguous claims were either narrowed to the exact condition supported by the source or excluded. Examples of corrections included distinguishing projected accelerator performance from measured hardware, separating site-month datacenter carbon from single-model carbon, identifying simulation-based scalability, and preserving the difference between operational and embodied emissions.

Each paper then underwent a second-pass audit. The second pass rechecked metadata, numerical values, comparator conditions, measurement boundaries, quality evidence, and limitations. The complete evidence corpus comprised 54 peer-reviewed research papers, together with one software artifact (CodeCarbon). All corpus entries underwent structured extraction and staged second-pass verification using a common verification protocol. Verification records captured bibliographic metadata, source locations (e.g., page, table, figure, or equation where applicable), measurement boundaries, retained numerical claims, limitations, and cross-paper consistency checks. Every entry was verified

with the same protocol, and each entry's retained claims, evidence class, measurement boundary, and source locations are recorded in the registry, extraction matrix, and claim log provided in Supplementary File S1, so that every classification can be traced and independently rechecked.

Extraction and evidence classification were performed by a single reviewer applying the fixed rubric in Table 2, under which each retained result is assigned to exactly one evidence class according to its measurement basis. To reduce single-rater error, every entry underwent an independent second pass on a later date. This pass re-derived each classification from the cited source location rather than from the first-pass record. Disagreements between passes were resolved by narrowing the claim to the exact condition supported by the source, or by excluding it. Single-reviewer classification is acknowledged as a limitation in Section 16. To make the process auditable rather than to assert inter-rater agreement, the full rubric, the per-entry evidence classifications, and their source locations are released as Supplementary Materials so that any reader can reproduce or contest an individual assignment.

Table 2. Evidence hierarchy used to interpret retained claims.

Evidence Class	Accepted Basis	Allowed Interpretation	Prohibited Interpretation
Direct	Executed workload with described measurement	Energy/power/carbon result within stated boundary	Generalization beyond workload, hardware, or boundary
Modeled	Measured profiles or traces used in a model	Model outcome under declared assumptions	Presentation as an end-to-end measured run
Simulated	Simulator, emulation, or synthetic trace	Comparative result in simulated setting	Equivalence to deployed production performance
Reported comparison	Survey or secondary summary	Taxonomy and candidate identification	Use as primary numerical evidence
Enabling	Runtime, memory, communication, or scale result	Mechanism may alter resource demand	Claim of energy saving without measurement
Inference	Author synthesis derived from multiple sources	Clearly labeled interpretation supported by cited evidence	Presentation as a directly measured or source-stated result

3.7. Cross-Paper Synthesis and Novelty Check

After completing the canonical corpus, the evidence was synthesized by capability rather than by publication chronology. The synthesis examined measurement, power control, model and token allocation, precision, memory, communication, parallelism, placement, carbon scheduling, reliability, hardware, and multi-resource optimization. A capability was marked as established only when supported by a primary paper or audited software evidence; a survey alone could not establish that a capability was solved. The methodology intentionally prioritizes evidence quality over publication chronology so that comparisons remain scientifically interpretable.

Because novelty in systems research is time-sensitive, a targeted 2025–2026 competitor search was performed after the corpus review. This search focused on systems that could invalidate the proposed gap: energy-aware large-model training, carbon-aware distributed pretraining, multi-objective cluster control, cross-datacenter planning, and energy-aware serving. It confirms several recent peer-reviewed competitors. Perseus reduces training-energy bloat without throughput loss [17]; GREEN performs carbon-aware scheduling for machine-learning clusters [18]; recent measurement studies characterize distributed

LLM-training energy on multi-instance GPUs and under immersion cooling [19–21] and renewable-aware client selection has been applied to federated LLM training [22]. Kareus remains a verified partial competitor [23]. None of these systems individually performs joint selection of model, precision, memory, parallelism, placement, and power state under measured whole-system energy with quality, reliability, and carbon constraints. Consequently, broad claims such as being the first energy-aware trainer, or the first carbon-aware distributed LLM training system, were rejected. Within the reviewed corpus and the documented competitor search, the surviving gap is the absence of a verified runtime that jointly selects model and token budget, precision, memory and recomputation, distributed parallelism and communication, placement, and GPU power state using measured whole-system energy while adapting to failures and carbon intensity under final-quality and deadline constraints.

3.8. Validity and Reproducibility of the Review

The review process reduces several common sources of error, but it does not eliminate them. The current corpus is broad rather than exhaustive. The original corpus was curated by expert selection rather than by an exhaustive systematic query, and a complete historical retrieval log was not kept. To mitigate this, a documented update search with dated queries, deduplication, and explicit screening counts was executed; it is reported in Section 3.3 and Supplementary File S1. That update search was capped at the top 100 records per query and service, so it strengthens but does not fully replace an exhaustive systematic review. The corpus may also omit newly released, proprietary, or differently indexed work. Hardware generations, grid conditions, software frameworks, and workloads vary across studies, limiting direct numerical comparison.

Reproducibility is supported through the canonical registry, structured evidence matrix, claim log, second-pass audits, source hashes, competitor matrix, planned tables and figures, proposed architecture, evaluation protocol, and verified bibliography maintained in the project workbook. The documented update search in Section 3.3 provides dated queries and explicit screening counts; the present manuscript does not claim that the original corpus was assembled by that search, only that the search was used to test the corpus for completeness and recency.

4. Concepts, Metrics, and Measurement Boundaries

Energy-efficient large-model training is frequently discussed using terms that are related but not interchangeable. Runtime, throughput, floating-point operations, memory footprint, communication volume, accelerator utilization, monetary cost, power, energy, and carbon emissions describe different properties and must be reported separately [15,24].

4.1. Power and Energy

As defined in Section 2.4, energy is the integral of power over the execution interval. Consequently, a faster run may consume less, equal, or more energy depending on the power trajectory, and a lower power cap may reduce instantaneous power while increasing execution time enough to leave total energy unchanged or higher.

This distinction is especially important in GPU training. Several reviewed systems directly optimize device power, frequency, or energy, but their measurement boundaries differ. Some report GPU-only energy, some report server power, and others use modeled cluster energy derived from device profiles [25–27].

4.2. Runtime, Throughput, and Utilization

Runtime and throughput are performance metrics. Runtime measures elapsed completion time, whereas throughput measures work completed per unit time, such as tokens

per second. Utilization describes how fully a resource is occupied, but high utilization does not guarantee high energy efficiency. A system can increase utilization by keeping more hardware active, which may raise total power and energy even when throughput improves. Conversely, a low-utilization system may waste static power over a longer duration.

The reviewed literature contains many important throughput and utilization improvements in pipeline parallelism, sharding, communication, attention kernels, heterogeneous placement, and production-scale execution. These results are treated as enabling evidence unless the study also reports power or energy. This prevents a common but invalid inference in which speedup is automatically relabeled as energy reduction.

4.3. FLOPs, Model Size, and Memory

Floating-point operation counts characterize computational work under an abstract operation model. They are useful for scaling analysis, but they do not capture data movement, memory access, communication, hardware efficiency, idle power, or control overhead [9,10]. The reviewed carbon-accounting literature, therefore, treats FLOPs as an enabling metric rather than a complete energy measurement [14,16].

Model size and memory footprint similarly describe capacity rather than electrical consumption. Quantization, pruning, activation checkpointing, fully sharded training, and low-rank adaptation can reduce memory demand or make a model feasible on fewer devices [12,28–30]. These changes may reduce energy, but the result depends on recomputation, communication, hardware count, kernels, and runtime.

4.4. Communication Volume and Network Cost

Communication volume is generally reported in bytes, messages, synchronization frequency, or effective bandwidth. Reducing communication can improve runtime and scalability, particularly on bandwidth-constrained clusters or wide-area links [31–33]. However, network energy depends on topology, link technology, switch utilization, retransmissions, and idle power.

For this reason, communication-oriented studies are classified as direct network-energy evidence only when NIC, switch, or calibrated network energy is measured. Otherwise, they remain enabling evidence. This boundary is particularly important for geodistributed and low-frequency-synchronization systems, where communication savings may be substantial, but energy and carbon outcomes also depend on region, routing, waiting time, and additional local computation.

4.5. Monetary Cost

Cloud or cluster cost is influenced by energy but also by pricing policy, reserved capacity, accelerator scarcity, network charges, storage charges, and organizational accounting. Lower monetary cost may result from moving to cheaper regions, using spot instances, or improving scheduling without reducing energy. Conversely, a lower-energy configuration can cost more if it uses premium hardware or finishes during an expensive tariff period. Cost is, therefore, reported as a separate outcome rather than a substitute for energy or carbon.

4.6. Operational and Embodied Carbon

Operational carbon emissions are commonly calculated by multiplying consumed electrical energy by a carbon-intensity factor. The interpretation depends on the signal used. Average grid intensity, marginal emissions, cloud-provider regional factors, renewable-curtailment signals, and market-based accounting can produce different decisions [34,35].

Embodied carbon covers emissions associated with manufacturing, transport, and infrastructure construction. It is a lifecycle quantity rather than an instantaneous runtime signal. The reviewed evidence supports using embodied carbon in procurement and

capacity-planning decisions, but cautions against repeatedly charging sunk manufacturing emissions to runtime scheduling decisions [16,36,37].

4.7. Quality, Convergence, and Energy to Target

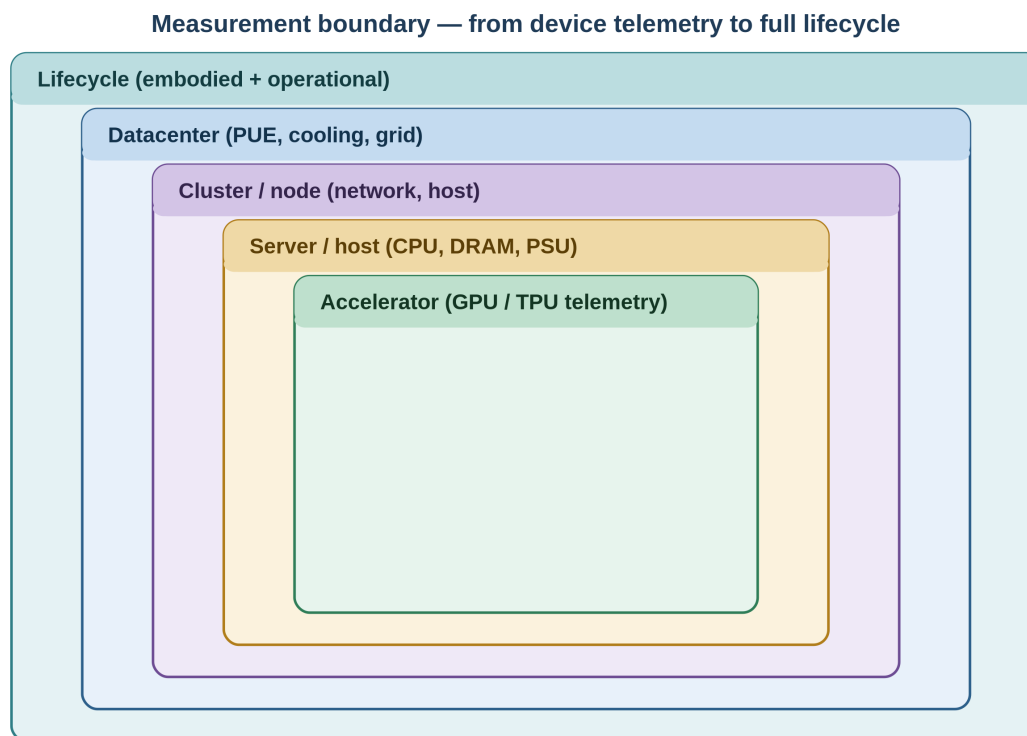
Energy comparisons are meaningful only when the trained models satisfy a common quality requirement or when the quality trade-off is explicitly reported. A configuration that stops early, uses fewer tokens, lowers precision, compresses gradients, or changes synchronization can reduce resource use while also altering the final model. The preferred primary metric for training-system evaluation is, therefore, energy to a declared target quality, accompanied by time to target and final quality.

Target quality may be expressed through validation loss, perplexity, task performance, or a predeclared non-inferiority margin. The target, validation cadence, dataset, and uncertainty should be reported. This requirement also prevents improvements in hardware throughput from being compared with runs that optimize different training objectives or terminate at different stages of convergence.

4.8. Measurement Boundaries

A reported energy value is inseparable from its measurement boundary. The narrowest common boundary is accelerator-only energy. Wider boundaries may include host CPUs, RAM, NICs, storage, power-supply losses, cooling, datacenter overhead through power usage effectiveness, and electricity generation. Lifecycle boundaries additionally include manufacturing and infrastructure [14,16,38].

Figure 4 illustrates how measurement boundaries become more complete as they expand from accelerator-level telemetry to host, cluster, datacenter, grid-carbon, and lifecycle accounting.



Reported energy is valid only within its stated boundary; broader boundaries capture more of the true footprint.

Figure 4. Measurement boundaries become more complete as they expand from accelerator-level telemetry to host, cluster, datacenter, grid-carbon, and lifecycle accounting.

Table 3 summarizes measurement boundaries and the valid interpretation of reported energy values.

Table 3. Measurement boundaries and valid interpretation of reported energy values.

Boundary	Typically Included	Common Omissions	Valid Interpretation
Accelerator	GPU/TPU device power or energy	CPU, RAM, NIC, storage, cooling	Device-level efficiency only
Server	Accelerator, CPU, RAM, local components, PSU losses	Shared network, facility cooling	Host-level workload energy
Cluster	Allocated servers and sometimes network	Shared infrastructure and cooling allocation	Job or cluster operational energy
Datacenter	IT load and facility overhead	Embodied emissions unless added	Site operational energy/carbon
Lifecycle	Operational plus embodied components	Often incomplete supply-chain detail	Declared lifecycle footprint

4.9. Direct Measurement, Estimation, and Uncertainty

Measurement methods also differ in evidentiary strength. Hardware energy counters and calibrated external meters provide direct observations within their stated boundary. Instantaneous power sensors require temporal integration. TDP-based methods, utilization-scaled models, and generic memory assumptions are estimates and should be labeled accordingly [14,15].

CodeCarbon illustrates this mixed measurement model. It prefers hardware counters for supported GPUs and CPUs, but may use modeled fallbacks when counters are unavailable. Its process mode estimates CPU and RAM attribution, while GPU energy remains device-level. It also excludes network, storage, cooling, and embodied emissions unless the user adds those boundaries explicitly [39].

4.10. Reporting Standard Adopted in This Survey

The remainder of this survey reports efficiency as a vector rather than a single score. Where available, the vector includes energy, power, runtime, throughput, memory, communication, carbon, cost, final quality, measurement boundary, and evidence type. Numerical comparisons are retained in their native units and original conditions. Cross-paper normalization is avoided when hardware, workload, quality target, or accounting boundary differ materially.

This reporting standard supports two goals. First, it prevents an enabling mechanism from being overstated as direct energy evidence. Second, it reveals the integration problem that motivates the research agenda: efficient large-model training requires simultaneous reasoning about quality, time, memory, communication, power, reliability, and carbon rather than independent optimization of one proxy.

Figure 5 presents the taxonomy of energy-efficiency mechanisms used to organize the mechanism-level review.

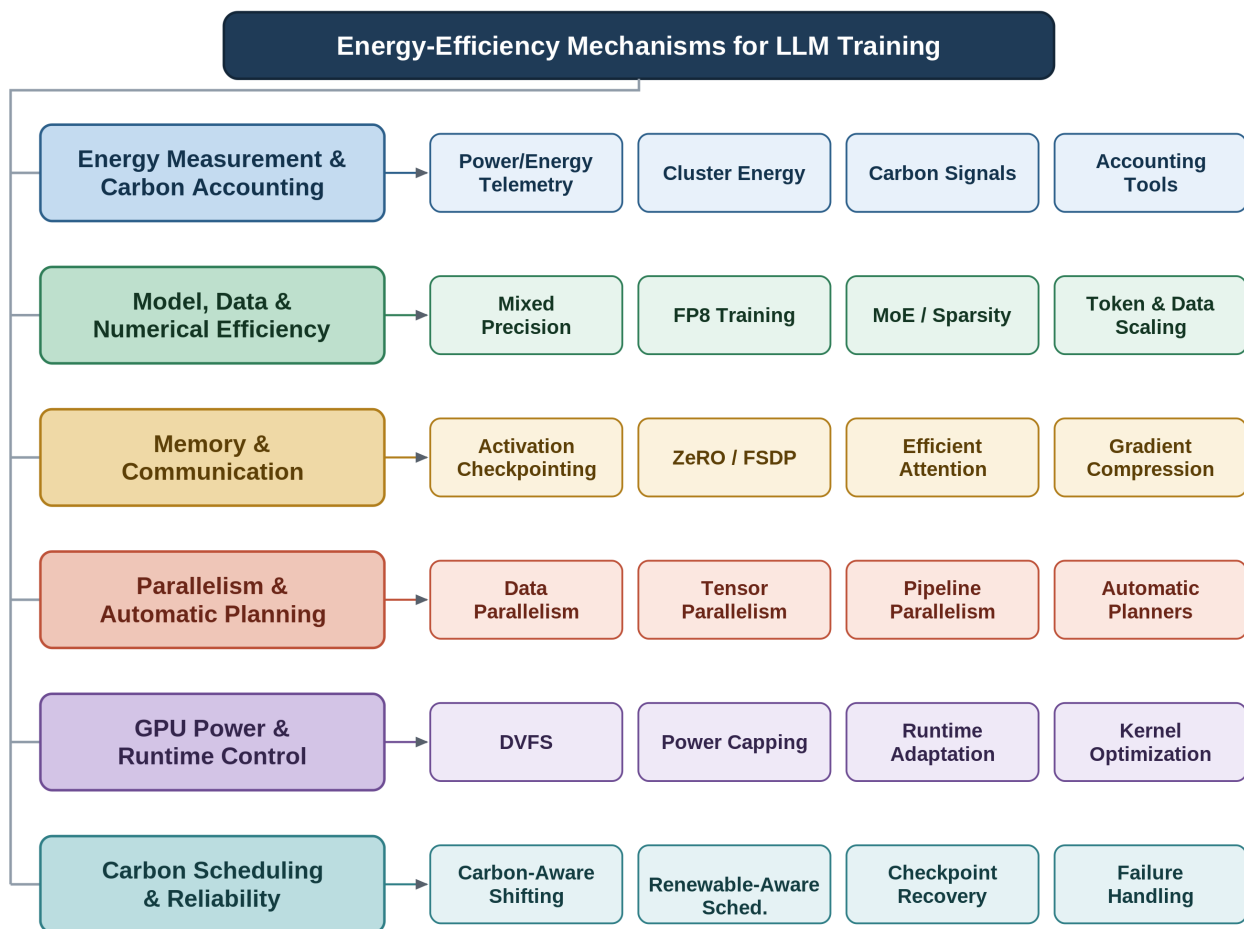


Figure 5. Taxonomy of energy-efficiency mechanisms for LLM training used to organize the mechanism-level review.

5. Energy Measurement and Carbon Accounting

Reliable optimization begins with reliable measurement. Large-model training spans accelerators, host processors, memory, interconnects, storage, cooling, and grid electricity, yet most published measurements cover only a subset of this stack. The reviewed literature, therefore, reveals two parallel challenges: collecting accurate telemetry and defining a transparent accounting boundary [15,16].

5.1. Hardware Counters and Device-Level Telemetry

The most direct software-accessible measurements come from hardware counters or vendor telemetry. Modern accelerators may expose instantaneous power, accumulated energy, utilization, clock frequency, memory activity, and temperature. CPUs may expose energy through interfaces such as running-average power limit counters, sometimes with separate package and memory domains. These interfaces support low-overhead sampling during training and are the foundation of many software energy trackers.

Device-level telemetry is valuable but incomplete. Accelerator counters normally represent the whole device rather than one process, rank, or container. Shared devices, therefore, create attribution ambiguity. Counter availability also depends on driver, operating system, permissions, and hardware generation. Energy counters may reset, wrap, or differ in resolution, while instantaneous power requires numerical integration over time. A defensible study must report the exact backend, sampling interval, device coverage, and treatment of missing samples.

5.2. External Power Meters and Server-Level Measurement

External meters, intelligent power-distribution units, board management controllers, and datacenter telemetry can capture a wider boundary than device counters. They may include accelerator, CPU, memory, storage, power-supply losses, and other server components. The TPU datacenter analysis is a useful hardware-context example because it compares performance and power at the deployed-system level, although it evaluates inference rather than LLM pretraining [40].

External measurement introduces its own design requirements. The instrument must be calibrated, synchronized with workload events, and sampled at a rate that captures transient phases. Shared-rack or shared-PDU measurements require a transparent attribution method. Idle power should be reported rather than silently subtracted, because idle allocation is part of the operational cost when resources are reserved for a job. When direct whole-server metering is available, it should be used to validate the sum of software counters and modeled components.

5.3. Software Tracking Frameworks

Software frameworks lower the barrier to collecting energy and carbon metadata. Experiment-impact-tracker and Carbontracker were designed to record training resource use and, in the case of Carbontracker, predict total energy, carbon, and duration from early epochs. CodeCarbon (version 3.2.2) provides Python and command-line integration with hardware-counter and modeled backends [15,39,41].

These tools differ in measurement backend, prediction model, and scope. CodeCarbon prefers direct counters where available but falls back to utilization-scaled or hardware-table estimates for unsupported systems. Its process mode estimates CPU and RAM attribution, while GPU energy remains device-level. Carbontracker's reported prediction error varies across workloads and metrics, which demonstrates that early-run forecasting should carry uncertainty rather than be presented as exact accounting. Software trackers should, therefore, record whether each component is directly measured, integrated from power samples, or estimated.

Table 4 compares software energy and carbon tracking approaches.

Table 4. Comparison of software energy and carbon tracking approaches.

Tool	Primary Method	Carbon Support	Typical Boundary	Main Limitation
Experiment-impact-tracker	Runtime resource and energy tracking	Yes	Process/device with modeled components	Coverage depends on available counters and system metadata
Carbontracker	Early-run prediction plus final accounting	Yes	Training workload with modeled carbon conversion	Forecast accuracy varies when later phases differ from early epochs
CodeCarbon	Hardware counters with modeled fallbacks	Yes	GPU device; estimated CPU/RAM attribution	Network, storage, cooling, and embodied emissions are excluded by default

5.4. Predictive Measurement and Early-Run Estimation

Predictive tools estimate final energy or carbon before training completes. This is useful for budgeting, early stopping, and policy enforcement, but prediction quality depends on whether early epochs represent later behavior. Large-model training can change phase

because of sequence length, activation memory, communication, checkpointing, evaluation, or failures [41,42].

Forecasting and accounting should also be separated. A predicted total can guide a decision, but the final publication result should be recomputed from observed measurements over the completed run. When prediction triggers early termination, the resulting model quality and saved energy must both be reported.

5.5. Carbon Conversion and Signal Semantics

Operational carbon is commonly calculated by multiplying energy consumed during each interval by a corresponding carbon-intensity value. The accounting quality, therefore, depends on temporal resolution, geographic mapping, and the semantics of the carbon signal [14].

Static regional averages are simple and reproducible, but they may obscure hourly variation and marginal grid response. Marginal signals are better aligned with the consequence of adding or shifting load, yet they are harder to obtain and forecast. Curtailment signals identify otherwise-wasted renewable generation, but their availability is geographically and temporally constrained. Carbon-aware studies must identify the signal type, provider, timestamp, region, update interval, and any forecasting assumptions.

5.6. Power Usage Effectiveness and Facility Allocation

Datacenter operational energy exceeds IT-device energy because of cooling, power conversion, lighting, and other facility loads. Power usage effectiveness is commonly used to allocate a facility multiplier to IT energy. Although useful, a single annual average PUE can misrepresent the marginal overhead of a specific training run [14].

PUE also does not capture every lifecycle effect. It excludes accelerator manufacturing, building construction, replacement cycles, and many supply-chain emissions. A study that multiplies GPU energy by PUE and grid carbon intensity is reporting operational datacenter carbon, not a complete lifecycle footprint.

5.7. Lifecycle and Embodied Accounting

Lifecycle studies extend the boundary beyond electricity use. The BLOOM analysis combined dynamic training energy, idle infrastructure, and embodied equipment emissions, demonstrating that a single accelerator-energy number can omit substantial sources [38]. Acme similarly provides evidence that production LLM development includes evaluation, failures, checkpointing, and underutilized components beyond successful training steps [42].

Embodied emissions require assumptions about manufacturing inventories, allocation across workloads, hardware lifetime, utilization, and replacement policy. Because these emissions are largely incurred at procurement time, runtime schedulers should not treat them as if they were newly emitted whenever a job is assigned to an already-owned server. The correct decision timescale is important: procurement may optimize lifecycle carbon, while runtime control optimizes operational energy and carbon under the installed fleet.

5.8. Large-Model and Datacenter Evidence

The reviewed large-model studies show the value of reporting both training-run and organizational boundaries. BLOOM provides a detailed model-level operational and lifecycle accounting example [38]. Earlier large-model carbon studies compare models such as T5, Meena, GShard, Switch Transformer, and GPT-3 and show how model, machine, datacenter, and location choices affect estimates [14,16].

Acme provides a complementary production perspective through a six-month trace from a 4704-GPU LLM datacenter. Its measurements show that training, evaluation, failures,

checkpointing, and underutilized components all influence datacenter resource use [42]. Such traces are particularly important because successful-run energy alone is insufficient for organizational planning.

5.9. Measurement Uncertainty and Error Propagation

Every measurement pipeline contains uncertainty. Device counters have resolution and coverage limits; external meters have calibration error; component attribution can be ambiguous; PUE is often an average; carbon factors may be stale or geographically coarse and embodied estimates depend on lifecycle assumptions. When carbon is derived from energy, uncertainty from both quantities propagates into the result.

At minimum, studies should report the measurement source, sampling rate, calibration method, missing-data treatment, modeled components, carbon-signal semantics, and boundary. Repeated measurements should be used to quantify run-to-run variation. Sensitivity analysis is required when network energy, cooling allocation, or embodied carbon is estimated rather than directly measured.

5.10. Recommended Measurement Protocol

This protocol does not require every study to own a full datacenter meter. It requires transparent disclosure and validation at the widest feasible boundary. A GPU-only study can still be valuable when labeled correctly; the problem arises when its result is generalized to whole-system or lifecycle efficiency. Table 5 presents the recommended measurement and carbon-accounting protocol.

Table 5. Recommended measurement and carbon-accounting protocol.

Stage	Required Action	Minimum Disclosure	Failure Condition
Calibration	Compare counters with an external or trusted reference	Instrument, date, workload, error	No calibration or unexplained drift
Boundary definition	Declare included and omitted components	GPU, CPU, RAM, NIC, storage, cooling, embodied	Whole-system claim with omitted boundary
Runtime collection	Record synchronized energy, power, progress, and events	Sampling interval, backend, missing samples	Unlabeled fallback or unsynchronized traces
Carbon conversion	Apply timestamped and typed carbon signal	Average/marginal/curtailment, source, region, timezone	Unknown signal semantics
Quality alignment	Measure until declared target quality	Metric, dataset, cadence, non-inferiority margin	Energy result without comparable quality
Uncertainty	Quantify measurement and run variation	Confidence interval and sensitivity analysis	Point estimate presented as exact
Reproducibility	Release manifests, traces, code, and formulas where possible	Versions, hashes, configuration, exclusions	Irreproducible accounting path

5.11. Implications for an Energy-Aware Self-Designing Trainer

The measurement review directly shapes the proposed research agenda. A self-designing trainer cannot optimize what it cannot observe, and it should not optimize a modeled quantity without tracking model confidence. The minimum telemetry layer should combine accelerator and host counters with per-rank identity, communication and checkpoint events, carbon-signal provenance, and an uncertainty flag for every estimated component.

The remaining measurement gap is, therefore, not the absence of software trackers. It is the absence of validated per-job, whole-system, distributed attribution that is sufficiently

accurate and timely for online control. This distinction prevents the proposed system from overstating novelty while identifying a concrete engineering and scientific contribution.

6. Model, Data, and Numerical Efficiency

Large-model training efficiency can be altered before the distributed runtime is chosen. Model size, token budget, data selection, sparsity, numerical precision, parameter adaptation, and architecture all change the amount and structure of computation presented to the system. These mechanisms are central to any future self-designing trainer because they define the algorithmic action space. However, most reviewed studies measure loss, perplexity, memory, throughput, or feasibility rather than training energy. This section, therefore, treats them primarily as enabling evidence and preserves the distinction between lower computational demand and directly measured energy reduction.

6.1. Scaling Laws and Model–Data Allocation

Scaling-law studies establish that model quality depends predictably on model size, data, and compute over substantial ranges, following approximately power-law relationships [10]. Early scaling analyses favored allocating additional compute mainly to larger models, whereas Chinchilla revised the compute-optimal balance toward substantially more training tokens for a given model size [9].

For energy-aware training, the main implication is not that compute-optimal is automatically energy-optimal. The scaling-law objective is loss under a FLOP budget, not measured energy under hardware, communication, memory, carbon, reliability, and deadline constraints. The energy-optimal allocation may differ across accelerator generations, sequence lengths, interconnects, and power states. A future planner should, therefore, use scaling relations as a quality model while learning a separate system-energy model.

6.2. Data Efficiency and Token Budget

Data efficiency affects how many useful training tokens are required to reach a target quality. Data deduplication, selection, curriculum, importance sampling, and synthetic or augmented data can alter convergence and total computation. The reviewed surveys identify these as major efficiency dimensions, but the primary corpus contains less direct energy evidence for data-policy changes than for systems mechanisms.

The central evaluation requirement is, therefore, quality-normalized accounting. Reducing tokens is beneficial only when the final model reaches the declared target or when the trade-off is explicitly quantified. Token count should be treated as a decision variable coupled to model size and data quality, not as an isolated proxy for energy.

6.3. Mixture-of-Experts and Conditional Computation

Mixture-of-experts architectures increase parameter capacity while activating only a subset of experts for each token. This can improve the relationship between model capacity and active computation. Large-model case studies such as GLaM and Switch Transformer show that sparse architectures can reach very large parameter counts with lower active computation than dense models of comparable capacity [43,44].

Conditional computation nevertheless shifts cost into routing, expert imbalance, communication, memory capacity, and fault sensitivity. Parameter count, active FLOPs, and training energy are, therefore, different quantities. Sparse models may reduce compute per token but require more devices or network traffic to host and exchange expert states. Energy claims require whole-system measurement under the actual expert-parallel topology.

6.4. Mixed-Precision Training

Mixed-precision training is one of the most established numerical-efficiency mechanisms. The reviewed work shows that lower-precision tensor operations can reduce memory use and accelerate supported operations while preserving training quality through FP32 master weights, loss scaling, and FP32 accumulation. On Volta-generation hardware, it reports operation-level speedups of approximately 2–6 times and roughly half the memory footprint for selected workloads [45].

These results demonstrate feasibility and performance, not an end-to-end energy reduction. Tensor-core acceleration can lower time and potentially energy, but the effect depends on achieved utilization, memory behavior, power draw, communication, and any stability overhead. Equal-quality full-training measurements are needed before interpreting mixed precision as an energy result.

6.5. FP8 Training

FP8 extends the mixed-precision approach by introducing dedicated low-precision formats (such as E4M3 and E5M2) for different tensor types. A peer-reviewed FP8 training study performs all hidden linear-layer computations in FP8 and trains language models from one to thirteen billion parameters without dynamic scaling factors or special hyperparameters, reporting quality competitive with a higher-precision comparator [46].

The evidence boundary is important: the study demonstrates FP8 training on hardware and reports quality, but it does not report measured whole-system energy. It, therefore, supports numerical viability and efficiency rather than a direct, quality-normalized energy result, which still requires primary end-to-end measurement.

6.6. Post-Training Sparsity

SparseGPT demonstrates that very large language models can be pruned in one shot without retraining. For OPT and BLOOM models, the paper reports that approximately 50% unstructured sparsity can be achieved with minor quality degradation, and it evaluates structured patterns such as 4:8 and 2:4 sparsity. The method can process OPT-175B on one A100 80 GB GPU in approximately four hours [29].

This is strong post-training compression evidence, but it does not reduce the pretraining energy already spent. Its training-energy relevance is indirect: sparse models may reduce later inference or adaptation cost, and sparsity-aware training could become part of a future design space. Real benefits also depend on kernels and hardware that can exploit the sparsity pattern.

6.7. Quantized Low-Rank Adaptation

QLoRA combines a frozen 4-bit quantized base model with trainable low-rank adapters and higher-precision computation. It introduces the NF4 data type, double quantization, and paged optimizers. The paper reports that a 65-billion-parameter model can be fine-tuned in less than 48 GB of memory on one GPU, whereas conventional 16-bit fine-tuning would require far more memory [28].

The evidence shows a major feasibility improvement for adaptation. It does not establish lower pretraining energy, and the reported quality comparisons apply to fine-tuning and instruction-following evaluation. QLoRA should, therefore, be positioned as an adaptation mechanism that may reduce hardware count, memory, and cost for a specific lifecycle stage.

6.8. Pruning, Quantization, and Adaptation as Lifecycle Choices

Compression and parameter-efficient adaptation affect different lifecycle stages. One-shot pruning occurs after pretraining; QLoRA reduces the resource requirements of fine-tuning; mixed precision and FP8 change numerical representation during training and mixture-of-experts changes model structure through conditional computation. These categories should not be collapsed into one efficiency claim because they change different resources at different points in the model lifecycle.

A publication-grade comparison should, therefore, report the stage at which savings occur, the previously incurred cost, whether the base model is frozen, the quality comparator, and whether specialized kernels are required. This stage-aware view is also necessary for deciding which variables belong in an online training controller and which belong in an offline model-selection or procurement workflow.

6.9. Interaction with Distributed Systems

Algorithmic choices alter systems behavior. Lower precision reduces tensor size and can change communication volume, kernel selection, memory headroom, and batch size. Activation sparsity and mixture-of-experts change load balance and routing. Quantization can introduce dequantization overhead. A smaller model may need more tokens to reach a target, while a larger model may create more communication and memory pressure.

These interactions explain why an energy-aware planner cannot select model, data, and precision independently of parallelism, recomputation, placement, and power state. The relevant outcome is the energy and carbon required to reach target quality under the complete distributed plan.

6.10. Comparative Evidence Summary

Table 6 summarizes comparative evidence for model, data, and numerical mechanisms.

Table 6. Comparative evidence for model, data, and numerical mechanisms.

Mechanism	Lifecycle Stage	Primary Measured Outcome	Quality Evidence	Direct Training Energy?	Main Limitation
Scaling laws	Pretraining design	Loss versus model/data/compute	Direct loss curves	No	FLOP objective, not system energy
Mixture-of-experts	Pretraining architecture	Capacity and active compute	Model-quality comparisons	Usually no	Routing, memory, and communication costs
Mixed precision	Training	Memory and operation speed	Generally matches FP32 with safeguards	No in reviewed paper	Hardware- and stability-dependent
FP8	Training	Simulated low-precision quality	Close to bfloat16 comparator	No	Simulated rather than measured hardware
SparseGPT	Post-training	Sparsity, perplexity, pruning time	Minor degradation near 50% sparsity	No	Does not recover pretraining energy
QLoRA	Fine-tuning	Memory feasibility, runtime, task quality	Strong fine-tuning comparisons	No	Adaptation only, not pretraining

6.11. Implications for Self-Designing Training

The reviewed evidence shows that the model and numerical action space are already rich. A proposed system cannot claim novelty merely by combining mixed precision, pruning, quantization, scaling laws, or parameter-efficient adaptation. The unresolved contribution is to select among these mechanisms jointly with distributed-system controls using measured energy and a final-quality constraint.

A practical first implementation should restrict the online action space to reversible and quality-safe variables such as supported precision modes, batch and accumulation, recomputation, parallelism, placement, and GPU power state. Model architecture and token-budget decisions are better made before training or at coarse checkpoints because they can change the scientific objective. This separation between offline design and online control improves safety while retaining the full-stack research agenda.

7. Memory and Communication Efficiency

Memory and communication mechanisms determine whether a model can be trained at all and how efficiently a distributed system uses its devices. They are among the strongest enabling technologies in the reviewed corpus, but their energy consequences are rarely measured directly. These mechanisms primarily improve feasibility and scalability rather than directly reducing measured training energy, making careful evidence interpretation particularly important. The central systems trade-off is that reducing one resource can increase another: recomputation saves activation memory but adds arithmetic, sharding saves device memory but adds collectives, and compression saves bytes but may add local computation.

7.1. Activation Recomputation

Activation checkpointing trades memory for additional forward computation. Memory-efficient recomputation establishes schedules that reduce activation storage—down to logarithmic memory in the extreme—at correspondingly greater recomputation cost [47]. Selective recomputation later targets expensive Transformer activations and reports large memory reductions with improved throughput relative to full recomputation [30]. These methods are not monotonically energy-saving: recomputation raises arithmetic work, while lower memory pressure can enable larger batches, fewer devices, or faster kernels. Whether recomputation reduces total training energy depends on the balance between additional arithmetic and reductions in memory pressure, hardware requirements, or execution time.

7.2. Sharding and Memory Distribution

Fully sharded data parallelism and ZeRO-family methods distribute parameters, gradients, and optimizer state across workers [12,32,48]. They make very large models feasible and can improve throughput under constrained memory, but sharding introduces collective communication, metadata, and coordination overhead. The magnitude of these overheads depends on model scale, cluster topology, collective implementation, and available network bandwidth.

7.3. Attention and Kernel Efficiency

FlashAttention introduced an exact, IO-aware attention algorithm that reduces high-bandwidth-memory transfers, while FlashAttention-2 improved parallelism and work partitioning for higher throughput [49,50]. FlashAttention-2 reports substantial throughput gains on A100-class hardware, but does not directly measure energy. Kernel-level improvements are expected to influence energy because data movement is expensive; nevertheless, device power, host overhead, and end-to-end quality-normalized training must be measured before assigning a net energy effect.

7.4. Gradient and Collective Compression

Deep Gradient Compression reduces transmitted gradients by hundreds of times while preserving task quality through momentum correction, clipping, masking, and warm-up [31]. ZeRO++ reduces communication volume through quantization and hierarchical partitioning [32], while DiLoCo reduces synchronization frequency across workers [33].

These studies demonstrate that communication can be compressed or avoided, but bytes saved are not network energy saved. Modern LLM validation, NIC and switch telemetry, local compute overhead, and topology are required for direct energy conclusions.

7.5. Cross-Mechanism Synthesis

Memory and communication are coupled through batch size, recomputation, sharding, parallelism, and checkpointing. A planner that treats them independently can select a memory-feasible plan that is communication-bound or a low-communication plan that uses more devices. The appropriate objective is whole-system energy to target quality under capacity and deadline constraints, with separate reporting of GPU, host, and network components. This interaction is the main synthesis from Section 7: feasibility mechanisms should be evaluated as part of a complete distributed plan rather than ranked independently.

Table 7 summarizes comparative evidence for memory and communication mechanisms.

Table 7. Comparative evidence for memory and communication mechanisms.

Mechanism	Primary Benefit	Measured Outcome	Quality Condition	Direct Training-Energy Evidence?	Main Caveat
Sublinear checkpointing	Activation memory	Memory/runtime	Exact gradients	No	Additional forward work
Selective recomputation	Memory and throughput	Memory/throughput	Training semantics	No	Model/hardware dependent
FSDP/ZeRO	State sharding	Memory/throughput/bytes	Semantic equivalence	No	Collective overhead
FlashAttention-2	Lower memory traffic	Throughput	Equivalent attention	No	Kernel and GPU specific
DGC	Gradient bytes	Communication and task quality	Safeguards required	No	Old CNN/RNN workloads
DiLoCo	Synchronization count	Perplexity and communication	Small-model validation	No	Limited scale and hardware detail

8. Parallelism, Placement, and Runtime Planning

Distributed-training planning has progressed from manually selected parallelism strategies to automatic search over stage partitions, sharding, replication, placement, batch configuration, and heterogeneous resources. These systems provide strong evidence for feasibility, throughput, and scalability, but comparatively little direct whole-system energy evidence. Automatic parallelism itself, therefore, cannot support a novelty claim; the remaining question is how plans should be selected and adapted under energy, quality, reliability, and carbon constraints.

8.1. Pipeline and Hybrid Parallelism

GPipe combines synchronous micro-batching with rematerialization and demonstrates very large capacity, although its largest table entries are feasibility limits rather than fully trained models [13]. PipeDream profiles a model and uses dynamic programming to choose stage partitions, replication, and in-flight minibatches, then applies 1F1B scheduling with weight stashing [8]. These systems establish automatic or semi-automatic pipeline planning and time-to-accuracy comparisons on earlier model families.

8.2. Automatic and Heterogeneous Parallelism Planning

FlexFlow searches the SOAP parallelization space [51], Alpa automates inter- and intra-operator parallelism [52], and Galvatron combines multiple parallelism dimensions [53]. Whale addresses heterogeneous resources [54], while MegaScale demonstrates the oper-

ational realities of training across more than 10,000 GPUs [55]. Together, these systems cover much of the plan-generation space needed by a self-designing trainer, but their objectives are dominated by throughput, memory, or scalability rather than measured whole-system energy.

8.3. Training and Cluster Co-Adaptation

Pollux jointly adapts GPU allocation, placement, batch size, gradient accumulation, and learning rate through a goodput objective that includes statistical efficiency [56]. Carbon- and energy-aware cluster schedulers extend this decision space toward explicit energy budgets and grid-carbon signals [18]. The proposed research must explicitly extend both directions: Pollux provides executed quality-aware co-adaptation through its goodput objective, while cluster-level energy and carbon scheduling contributes an energy-constrained search space that still requires measured, quality-aware whole-system evaluation.

8.4. Heterogeneous, Decentralized, and Runtime-Adaptive Placement

Heterogeneous planners and decentralized foundation-model training show that placement can account for unequal compute and communication [54,57]. The reviewed canonical corpus does not contain a fully verified primary source for the CrossPipe and 2026 curtailment-aware claims that appeared in the earlier draft; those unsupported claims are, therefore, removed from this revision.

8.5. Cross-Mechanism Synthesis

A critical distinction is whether a system searches once before training or replans during execution. Static planners can be highly sophisticated yet cannot respond to power events, carbon changes, failures, thermal drift, or evolving bottlenecks. Runtime adaptation introduces transition overhead, stability risks, and checkpoint requirements. A publication claim should, therefore, report the decision timescale, supported transition graph, rollback behavior, and measured net benefit. The literature establishes both automatic planning and runtime co-adaptation; the open problem is safe, quality-aware adaptation using measured whole-system energy. Automatic parallelism, heterogeneous placement, and runtime co-adaptation are established capabilities, but their dominant objectives remain throughput, memory feasibility, goodput, or modeled device energy. The unresolved problem is safe runtime selection and adaptation of distributed plans using measured whole-system energy while preserving target quality, deadlines, reliability, and transition safety.

9. Power and Energy Control

Power-control research provides some of the closest direct evidence for energy-aware training. GPU dynamic voltage and frequency scaling, power caps, phase-aware control, slack exploitation, and energy-budget schedulers have all been explored [17,23,25–27,58–61]. The evidence is strongest at the accelerator boundary and weaker for host, network, cooling, and quality-normalized end-to-end energy. Consequently, accelerator-level savings should not be generalized to whole-system energy without explicit measurement of the wider boundary.

9.1. Frequency and Power-Cap Control

GPU power states expose a trade-off between instantaneous power and execution time. Effective controllers profile or learn the workload response rather than assuming that the lowest frequency minimizes energy. The reviewed studies report meaningful GPU-energy reductions with bounded slowdown in their tested settings, but frequently use single-GPU or GPU-only boundaries. The energy-minimizing operating point depends on workload

phase, hardware generation, control range, and permitted slowdown; the lowest frequency or cap is not necessarily optimal.

9.2. Phase- and Slack-Aware Control

Training alternates among compute, memory, communication, synchronization, evaluation, and checkpointing phases. Phase-aware controllers reduce frequency during slack or communication without slowing the critical path. This mechanism is established, but modern distributed LLMs introduce nested collectives, pipeline bubbles, checkpointing, asynchronous work, and transition overhead that complicate phase detection and net-benefit accounting.

9.3. GPOEO and Kareus

GPOEO contributes runtime GPU energy optimization, and Kareus jointly changes kernel scheduling and GPU frequency for large-model training. Kareus reports up to 28.3% lower energy at equal time or 27.5% lower time at equal energy in its evaluation. These systems invalidate any claim that joint scheduling and GPU-frequency control is new. The remaining extension is full-stack, whole-system, quality- and failure-aware control. Kareus’s numerical results should be interpreted only within its evaluated workloads, hardware, and equal-time or equal-energy comparison conditions.

9.4. Evidence Boundary and Synthesis

GPU energy, server energy, and cluster energy must remain separate. Equal-time, equal-energy, and equal-quality comparisons must also remain separate. The proposed evaluation, therefore, treats whole-system energy to target quality as the primary outcome and reports accelerator energy as a component. This evidence boundary prevents a GPU-only saving from being generalized to the full distributed training system. Controller overhead, transition cost, and slowdown must be included when reporting the net benefit of runtime power control.

Table 8 summarizes evidence boundaries and unresolved extensions in power-control systems.

Table 8. Evidence boundaries and unresolved extensions in power-control systems.

System Family	Variables	Boundary	Adaptation	Quality/Time Condition	Unresolved Extension
DVFS/power-cap controllers	Frequency, cap	Mostly GPU	Runtime	Bounded slowdown	Joint plan and whole-system energy
Phase/slack controllers	Frequency by phase	Mostly GPU	Runtime	Critical-path preservation	LLM-scale phase transitions
Kareus	Kernel schedule and frequency	Training energy	Runtime	Equal-time/equal-energy	Carbon, failure, and broader action space

10. Carbon-Aware Scheduling and Lifecycle Design

Carbon-aware computing changes when, where, or how workloads run in response to electricity-emissions signals. Lower operational carbon does not necessarily imply lower energy because carbon-aware actions may delay execution, move work, change scale, or use additional infrastructure. Outcomes depend on carbon-signal semantics, forecast quality, regional capacity, workload flexibility, deadlines, and policy constraints. The reviewed work spans deployed virtual-capacity curves, elastic scheduling, renewable and battery design, global trace analysis, procurement metrics, and recent carbon-aware LLM training.

10.1. Temporal and Spatial Shifting

Google's virtual-capacity-curve system demonstrates operational reduction in flexible load and power during high-carbon periods [34]. CarbonScaler jointly adjusts scale and timing under deadlines [62]. Earlier work on temporal workload shifting similarly showed that flexible cloud workloads can be delayed toward lower-carbon periods, while emphasizing the importance of workload deadlines and carbon-intensity variation [63]. These systems show that carbon-aware control is deployable, but neither provides full LLM configuration planning or quality modeling. These results establish deployable carbon-aware scheduling, not a general reduction in training energy.

10.2. Practical Limits

The carbon-shifting limits study analyzes 123 regions and shows that ideal global benefits can be much larger than practical benefits once capacity, latency, privacy, and workload constraints are applied [35]. More than 70% of regions exhibit limited daily carbon-intensity variation, and benefits decline as grids decarbonize. Carbon-aware scheduling should, therefore, be treated as one objective, not the sole sustainability strategy. Reported outcomes must also distinguish average carbon intensity, marginal emissions, and renewable-curtailment signals because these signals represent different accounting and control objectives.

10.3. Lifecycle Design and the Sunk-Carbon Problem

Carbon Explorer jointly considers renewables, batteries, server capacity, scheduling, operational carbon, and embodied carbon at design time [36]. The sunk-carbon analysis argues that embodied emissions should guide procurement but should not be repeatedly charged as a marginal runtime cost among already-owned machines [37]. The survey, therefore, separates fleet design from runtime placement. Runtime controllers should optimize controllable operational quantities, while embodied-carbon allocation should primarily inform procurement and fleet-design decisions.

10.4. Positioning Against Recent Carbon-Aware Training Work

The verified corpus establishes temporal and spatial workload shifting, elastic carbon-aware scheduling, and lifecycle-aware datacenter design, but it does not contain a fully verified primary source for every recent system named in earlier drafts. This revision, therefore, avoids unsupported claims about GREEN, SustainFed-LLM, CrossPipe, Optimus, or a 2026 curtailment-window trainer. Before submission, a fresh documented competitor search should identify and cite any such primary papers. The defensible conclusion from the current evidence is narrower: a future system cannot claim novelty for carbon-aware scheduling alone and must demonstrate joint LLM configuration, measured whole-system energy, quality preservation, and safe runtime adaptation. Any integrated future system should also record carbon-signal provenance, including source, region, timestamp, update interval, and signal semantics.

10.5. Evidence Boundary and Cross-Layer Synthesis

Carbon-aware scheduling and lifecycle design are established research areas, but they operate under different accounting signals and decision timescales. The unresolved integration challenge is to coordinate carbon-aware execution with measured whole-system energy, target quality, reliability, transition cost, and operational constraints. This connection motivates the reliability discussion in Section 11 because failures and recovery can change both completion energy and the timing of carbon emissions.

11. Reliability and Production Operations

Reliability is a fundamental component of energy-efficient large-model training because failures increase the energy required to reach target quality through lost computation, repeated data processing, checkpoint recovery, and delayed completion. Although reliability studies rarely optimize energy directly, they strongly influence the operational energy efficiency of large-scale training systems.

11.1. *MegaScale*

MegaScale demonstrates 175-billion-parameter training at 12,288 GPUs, with production handling of more than 100 restarts and high rates of automatic fault diagnosis and remediation [55]. Its contribution is operational scalability and reliability rather than energy optimization. The reported operational experience also shows that reliability mechanisms become increasingly important as cluster scale grows because even low component-level failure rates can produce frequent system-level interruptions.

11.2. *Acme*

Acme analyzes approximately 1.09 million jobs over six months in a 4704-A100 data-center [42]. It separates pretraining, evaluation, failures, checkpointing, utilization, server-component power, and site-level carbon. The trace shows why successful-run energy alone is insufficient: failed jobs, evaluation, and infrastructure overhead materially affect the organization-level footprint. Organizational energy accounting should, therefore, include unsuccessful executions, validation workloads, checkpoint overhead, and infrastructure utilization rather than only completed training jobs.

11.3. *Failure-Aware Energy Optimization*

A full-stack optimizer should minimize expected remaining energy to target quality, including failure probability, lost work, checkpoint cost, and recovery. This requires a failure model and a safe rollback manager. Fault injection, recovery-energy accounting, and completion probability should be reported; fewer failures must not be presented as energy savings unless reductions in repeated computation and recovery overhead are measured within the stated boundary while final quality is preserved.

11.4. *Cross-Mechanism Synthesis*

The reviewed evidence demonstrates that production-scale reliability is well established, but reliability remains largely disconnected from energy-optimization objectives. Future systems should jointly optimize expected remaining energy, target quality, recovery cost, checkpoint strategy, and completion probability rather than treating failures solely as operational events. This integration is the principal research opportunity identified from the reliability literature.

12. Cross-Paper Synthesis

The reviewed evidence consistently indicates that the research problem is not a shortage of individual efficiency mechanisms. It is a coordination, measurement, and validation problem across layers. Model and numerical methods change the amount and form of learning work; memory and communication methods determine feasibility and distributed overhead; planners select parallelism and placement; power controllers change device behavior; carbon schedulers change execution time or location and reliability mechanisms determine how much work must be repeated. These controls interact, and a choice that improves one proxy can worsen total energy or final quality. To our knowledge, no reviewed

system simultaneously optimizes all of these mechanisms using measured whole-system energy while enforcing target-quality, reliability, and policy constraints.

Figure 6 presents the evidence-aware trade-off matrix across mechanism classes.

Mechanism	Direct energy evidence?	Likely net energy effect
Mixed precision / FP8	Limited	Workload-dependent
Activation recomputation	No	Boundary-dependent
MoE / sparsity	Limited	Workload-dependent
Gradient / comm. compression	No	Boundary-dependent
Power capping / DVFS	Yes	Measured lower in selected studies
Automatic parallelism	No	Not established
Carbon-aware scheduling	Limited	Not established

Direct evidence: Yes = training energy directly measured · Limited = some studies only · No = not measured.
Speed or memory gains are not energy savings unless energy is measured at the relevant boundary.

Figure 6. Evidence-aware trade-off matrix showing that direct training-energy support and likely net energy effects vary substantially across mechanism classes.

Figure 7 summarizes cross-mechanism capability coverage and the remaining gaps.

To keep the capability-coverage labels reproducible, the qualitative terms in Table 9 are defined by the number of corpus studies supplying *direct* measured energy or carbon evidence for a capability: *Mature* denotes five or more such studies, *Some*, *Partial*, or *Limited* denotes two to four, *Rare* denotes at most one, and *No* denotes none. The same thresholds apply to the coverage labels in Table 1, where *Limited* similarly indicates partial coverage supported by at most a few studies. These labels describe the strength of *direct* evidence only, not the overall importance of a mechanism, several of which are widely deployed on the basis of enabling evidence.

Across the corpus, direct energy evidence is concentrated in measurement, GPU power control, and selected carbon-aware systems. Model, memory, communication, and automatic-parallelism studies provide mostly enabling evidence. Quality is often evaluated, but rarely together with whole-system energy and runtime adaptation. Reliability is demonstrated at production scale, yet lost-work energy is usually absent from the optimization objective. The strongest defensible survey conclusion is, therefore, that integrated evaluation is less mature than the component mechanisms themselves. Survey-derived research gap: existing work provides mature solutions for individual optimization mechanisms, but, within the reviewed corpus and the documented search, we found no verified system that integrates model configuration, distributed planning, power control, carbon awareness, reliability, and measured whole-system energy under explicit target-quality constraints.

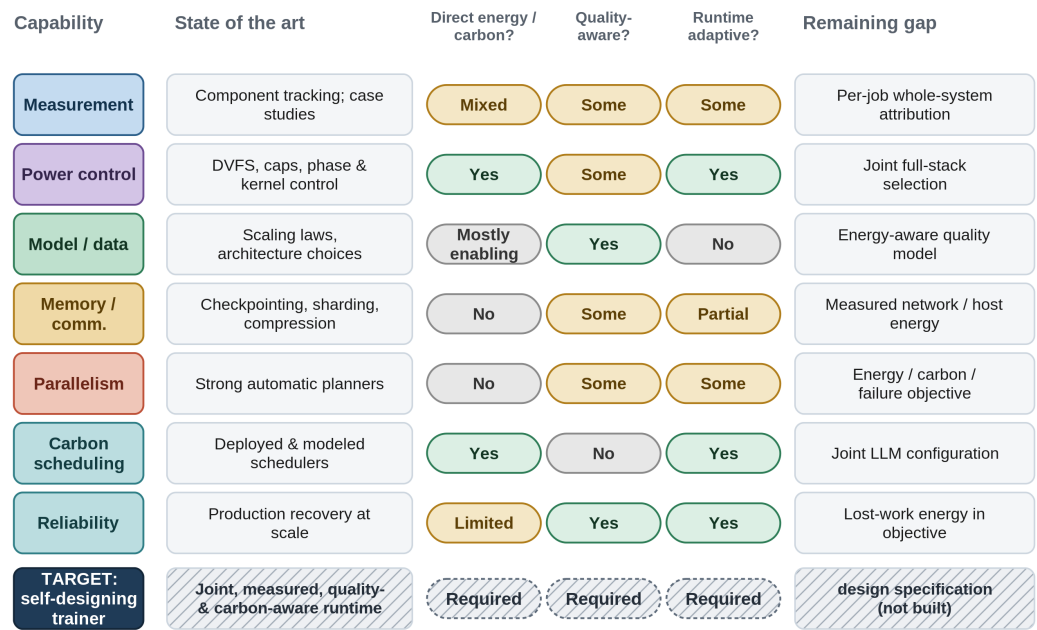


Figure 7. Cross-mechanism capability coverage and remaining gaps. The target self-designing trainer row is a research specification, not an implemented system.

Table 9. Cross-mechanism capability coverage and remaining gaps.

Capability	State of the Art	Direct Measured Energy/Carbon Evidence?	Quality-Aware?	Runtime Adaptive?	Remaining Gap
Measurement	Mature component tracking and case studies	Mixed	Usually observational	Monitoring	Per-job distributed whole-system attribution
Power control	DVFS, caps, phase and kernel control	Yes, often GPU	Time constraints	Yes	Joint full-stack selection
Model / data	Scaling and architecture choices	Mostly no	Yes	Mostly static	Energy-aware quality model
Memory / communication	Checkpointing, sharding, compression	Rare	Often	Partial	Measured network/host energy
Parallelism / placement	Strong automatic planners	Rare	Semantic or goodput	Some	Energy / carbon / failure objective
Carbon	Deployed and modeled schedulers	Yes	SLO, rarely model quality	Yes	Joint LLM configuration
Reliability	Production recovery at large scale	No direct optimization	Convergence protected	Yes	Lost-work energy in objective

Quantitative Evidence Summary and Study-Quality Assessment

To complement the qualitative synthesis, the 55 reviewed entries were quantified across primary mechanism, evidence class, direct energy measurement, measurement boundary, training regime, and quality condition, and each entry was scored on an eight-domain study-quality rubric (peer-review status, workload directness, energy-measurement rigor, boundary completeness, comparator adequacy, quality control, scale and replication, and artifact availability; zero to two points each) and rated for risk of energy-evidence over-interpretation (high 0–6, moderate 7–11, low 12–16). Table 10 reports the distribution; the per-entry registry, the rubric, and the scores are provided in Supplementary File S1. Only 15 of 55 entries provide direct measured energy or carbon

evidence, and only nine report a multi-component, cluster, or lifecycle boundary, whereas 33 are enabling or secondary. Although 36 entries report or preserve model quality, few report energy at a declared or matched target quality, which is the central evaluation gap this survey identifies. No entry is rated high risk, but 41 are moderate, which reflects sound mechanism-level engineering evidence alongside limited direct, quality-normalized, whole-system training-energy measurement.

Table 10. Quantitative summary of the 55 reviewed entries by primary mechanism, evidence class, direct energy measurement, measurement boundary, training regime, quality condition, and study-quality risk. Per-entry classifications are provided in Supplementary File S1.

Dimension	Category	Count
Evidence class	Direct measured	15
	Modeled or simulated	7
	Enabling or secondary	33
Direct energy measurement	Yes	15
	No	40
Measurement boundary	Multi-component/cluster/lifecycle	9
	Partial device/server	12
	Not reported/not an energy study	34
Training regime	LLM or large-model pretraining	23
	LLM fine-tuning	1
	Generic DNN/ML training and scheduling	20
	Inference or hardware characterization	2
	Survey, context, or tooling	9
Risk of over-interpretation	Low (12–16)	14
	Moderate (7–11)	41
	High (0–6)	0
Primary mechanism	Measurement and carbon accounting	9
	Model, data, and numerical efficiency	7
	Memory and communication	8
	Parallelism, placement, and planning	9
	Power and energy control	8
	Carbon-aware scheduling and lifecycle	6
	Reliability and operations	2
	Survey, context, or hardware	6
Quality condition	Model quality reported or preserved	36
	Quality not evaluated (tool, scheduler, survey)	19

13. Research Gaps and Survey-Derived Agenda

The following gaps emerge from the cross-paper synthesis rather than from any single study. The reviewed literature contains mature individual mechanisms, but five survey-level gaps remain. First, studies use inconsistent measurement boundaries and evidence types. Second, many comparisons are not normalized to the same model quality. Third, distributed overheads, failures, and transition costs are often omitted. Fourth, mechanism interactions are rarely evaluated under one objective. Fifth, results are difficult to reproduce because telemetry, carbon signals, configuration manifests, and uncertainty are reported inconsistently. These gaps motivate the recommended evaluation framework and inform the longer-term self-designing training direction, but they do not constitute empirical validation of the conceptual architecture presented later in the manuscript.

13.1. Measurement and Attribution Gap

The first gap is transparent attribution across accelerator, host, network, storage, cooling, and lifecycle boundaries. Existing studies frequently measure only a subset of these components, and modeled values are not always clearly distinguished from direct observations.

13.2. Cross-Study Comparability Gap

Cross-study comparison remains difficult because workloads, model sizes, hardware, quality targets, and accounting boundaries differ. Future surveys and experiments should use common reporting fields and avoid normalized rankings when the underlying conditions are not comparable.

13.3. Mechanism-Integration Gap

Individual mechanisms are well studied, but their interactions are less understood. Precision, recomputation, sharding, communication, placement, and power can improve one metric while worsening another. Future systems should integrate these mechanisms incrementally and report the net effect.

13.4. Reliability and Carbon-Accounting Gap

Failures, recovery, and carbon signals are often evaluated separately from training energy and final quality. Future work should report lost-work energy, checkpoint and recovery overhead, carbon-signal provenance, and the constraints that limit temporal or spatial workload shifting.

13.5. Evaluation and Reproducibility Gap

Future work needs equal-quality comparisons, explicit energy boundaries, transition overhead, fault and recovery accounting where relevant, carbon-signal provenance, statistical uncertainty, and reproducible manifests. These requirements form the main practical output of this survey.

14. Outlook: Toward Energy-Aware Self-Designing Training

The evidence synthesized above points to one long-term research direction, offered as a survey-derived outlook rather than a validated contribution of this paper: an energy-aware, self-designing distributed trainer. Figure 8 sketches a conceptual reference architecture for such a system; it is neither implemented nor experimentally validated here.

At a high level, such an architecture would comprise four layers, each mapping to mechanisms already established in the reviewed literature: a *sensing* layer combining per-rank accelerator and host telemetry with typed carbon, price, and power-budget signals; a *modeling* layer supplying calibrated latency, energy, memory, communication, quality, and failure predictors; a *planning* layer selecting a Pareto-efficient configuration of model, precision, memory, parallelism, communication, placement, and power state under hard quality, deadline, and policy constraints and a *runtime and safety* layer applying only supported transitions with stable checkpoints, bounded adaptation, rollback, and provenance logging. The central research opportunity is their coordinated integration under measured whole-system energy and an explicit target-quality constraint; sensible development would be incremental and empirically validated rather than monolithic. This architecture is a hypothesis for future work, not a result of this survey.

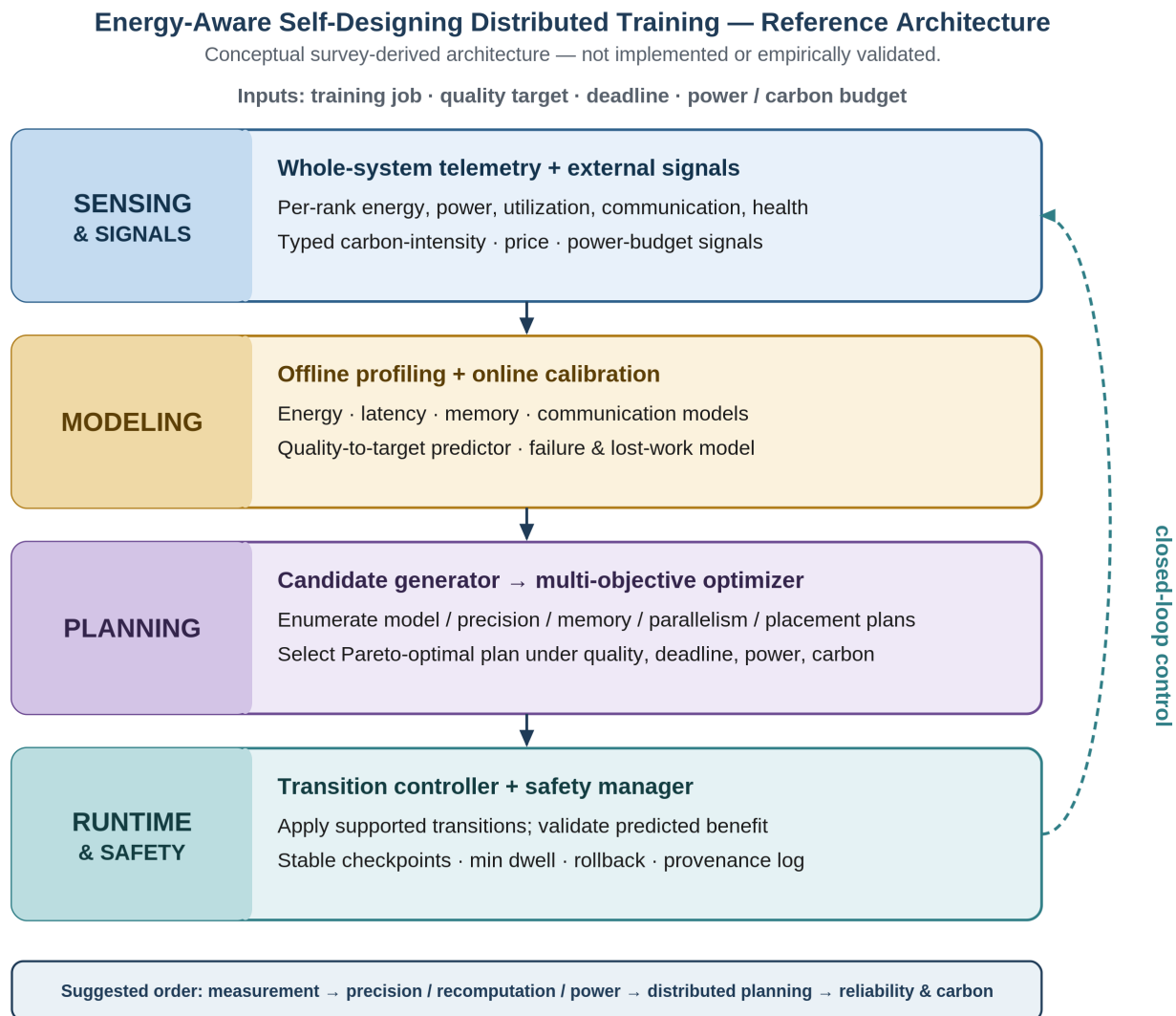


Figure 8. Conceptual survey-derived reference architecture for energy-aware self-designing distributed training. The design is not implemented or empirically validated.

Beyond the electronic training stack surveyed here, emerging hardware paradigms offer complementary, longer-horizon efficiency levers. Artificial intelligence is increasingly used to design metamaterials and metasurfaces, while, conversely, metamaterial and photonic structures are used to build energy-efficient neural hardware [64]. Optical neural networks have demonstrated on-hardware training of large photonic models at competitive accuracy [65,66], and metasurface-based optical encoders have reported order-of-magnitude reductions in system-level energy per inference [67]. These approaches alter energy at the device-physics level rather than at the software and systems level that governs current electronic LLM pretraining; they are, therefore, noted here as an adjacent hardware direction whose energy benefits for large-scale language-model training remain to be demonstrated under the whole-system, quality-normalized boundaries used in this survey, and are not generalized to present-day electronic training.

15. Recommended Evaluation and Reporting Framework

Based on the evidence synthesized throughout this review, the survey recommends a staged evaluation and reporting framework for future energy-efficient LLM-training studies. The preferred primary metric is whole-system energy to a declared target quality; carbon to target quality should be reported when a clearly typed and timestamped carbon-

intensity signal is used. Studies that cannot measure the whole system should report the widest feasible boundary and explicitly list omitted or modeled components.

Table 11 lists the recommended evaluation and reporting requirements.

Table 11. Recommended evaluation and reporting requirements.

Dimension	Minimum Requirement	Primary Comparator	Required Disclosure
Evaluation scales	Small repeatable model, 1–3B distributed model, and largest feasible model	Best baseline per regime	Model, data, token budget, target quality
Energy	GPU + host + best-effort network/storage/cooling	Equal-quality static and adaptive baselines	Counters, meters, modeled components, uncertainty
Carbon	Timestamped typed signal	Energy-only and carbon-aware baselines	Average/marginal/curtailment source and timezone
Reliability	Repeatable node/rank/network/checkpoint faults	Failure-blind planner	Lost work, recovery energy, completion probability
Control	Phase changes and noisy events	Static plan and unguarded controller	Transitions, overhead, rollback, violations
Statistics	Independent runs, disclosed random seeds, and blocked analysis	Paired where possible	Effect size, confidence interval, exclusions
Artifacts	Code, environments, manifests, telemetry configuration, raw traces, and analysis	Independent replay	Hashes, versions, licensing constraints

The strongest baseline set should include a tuned static plan, a throughput-oriented planner, a Kareus-like energy controller, Pollux-like co-adaptation, cluster-level energy-budget scheduling, and verified carbon-aware schedulers from the canonical corpus. Any newer competitor should be added only after its primary paper and evaluation conditions are verified. Core ablations should remove whole-system telemetry, quality prediction, failure modeling, carbon, runtime adaptation, power control, parallelism selection, communication control, safety, and online calibration.

16. Threats to Validity

The reviewed corpus is broad but not exhaustive. Rapidly changing publications, proprietary training systems, and inaccessible production measurements can alter the novelty conclusion. Coverage is also uneven across mechanism areas: some areas, notably reliability (two entries) and carbon-aware scheduling (six entries), are represented by only a few direct-evidence studies, so conclusions specific to those areas are correspondingly tentative and should be read as indicative rather than definitive. The documented competitor search should be repeated immediately before final publication.

Cross-paper numerical comparison is limited by model, dataset, hardware, software, topology, quality target, and measurement boundary. The survey, therefore, avoids pooled effect sizes and retains native metrics. Survey papers are used for taxonomy rather than primary numerical proof.

The future architecture is derived from the survey evidence but is not an empirical result. Its feasibility, predictive accuracy, control stability, transition overhead, and scalability remain to be demonstrated through separate implementation research. The survey's contribution is the mechanism taxonomy, evidence discipline, cross-paper synthesis, and recom-

mended evaluation framework. Consequently, the architecture should be interpreted as a hypothesis for future empirical research rather than evidence established by this survey.

Carbon accounting depends on average or marginal signals, regional mapping, forecast uncertainty, PUE, and lifecycle assumptions. Results should be interpreted within the declared accounting method rather than as universal environmental outcomes.

17. Conclusions

This survey demonstrates that energy-efficient large-model training is a multi-resource, multi-timescale systems problem. The literature already contains mature techniques for model and numerical efficiency, memory and communication reduction, automatic parallelism, power control, carbon scheduling, and reliability. Their reported outcomes cannot be collapsed into a single efficiency score, and enabling speed or memory results cannot be relabeled as energy savings.

The most important unresolved challenge is integrated, quality-normalized evaluation: selecting and adapting model and systems configurations using measured whole-system energy while accounting for deadlines, failures, transition overhead, and carbon signals. The conceptual reference architecture is one possible research direction rather than a validated solution. Future progress will depend less on discovering isolated optimization mechanisms than on integrating established mechanisms within transparent, reproducible, and quality-aware evaluation frameworks. Progress should be judged by energy and carbon to target quality, explicit measurement boundaries, uncertainty, recovery cost, and reproducibility.

The stable evidence registry (55 reviewed entries; every arXiv-only preprint from the original submission was withdrawn or replaced by a peer-reviewed source in revision), the eight-domain study-quality assessment, the documented search records, and the citation crosswalk are provided as Supplementary File S1.

Supplementary Materials: The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/electronics15194473/s1>, Supplementary File S1: Evidence Registry, Extraction Matrix, Claim Log, Search & Study-Quality Audit.

Author Contributions: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, visualization, and project administration, P.R.N.; writing—review and editing, P.R.N. and E.S.; supervision, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new datasets were generated in this study, which is a review of published literature; all reviewed works are cited in the manuscript. The evidence registry and extraction matrix (55 reviewed entries; all arXiv-only preprints were withdrawn or replaced by peer-reviewed sources in revision), the claim-level log, the eight-domain study-quality assessment, the documented literature-search protocol and screening ledger, and the citation crosswalk are provided in Supplementary File S1.

Acknowledgments: During the preparation of this manuscript, the authors used generative AI tools (OpenAI ChatGPT and Anthropic Claude) to assist with literature search, reference verification, drafting, and formatting. All AI-assisted content was subsequently reviewed, verified against primary sources, and edited by the authors, who take full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LLM	Large language model
GPU	Graphics processing unit
DVFS	Dynamic voltage and frequency scaling
FP8	8-bit floating point
MoE	Mixture of experts
PUE	Power usage effectiveness
FLOPs	Floating-point operations
ZeRO	Zero Redundancy Optimizer
FSDP	Fully Sharded Data Parallel

References

- Sevilla, J.; Heim, L.; Ho, A.; Besiroglu, T.; Hobbhahn, M.; Villalobos, P. Compute Trends Across Three Eras of Machine Learning. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022. [\[CrossRef\]](#)
- Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy, 28 July–2 August 2019; pp. 3645–3650.
- Kachris, C. A Survey on Hardware Accelerators for Large Language Models. *Appl. Sci.* **2025**, *15*, 586. [\[CrossRef\]](#)
- Duan, J.; Zhang, S.; Wang, Z.; Jiang, L.; Qu, W.; Hu, Q.; Wang, G.; Weng, Q.; Yan, H.; Zhang, X.; et al. Efficient Training of Large Language Models on Distributed Infrastructures: A Survey. *Vicinagearth* **2026**, *3*, 9. [\[CrossRef\]](#)
- Kim, G.I.; Hwang, S.; Jang, B. Efficient Compressing and Tuning Methods for Large Language Models: A Systematic Literature Review. *ACM Comput. Surv.* **2025**, *57*, 253. [\[CrossRef\]](#)
- Wan, Z.; Wang, X.; Liu, C.; Alam, S.; Zheng, Y.; Liu, J.; Qu, Z.; Yan, S.; Zhu, Y.; Zhang, Q.; et al. Efficient Large Language Models: A Survey. *arXiv* **2024**, arXiv:2312.03863.
- Xu, M.; Cai, D.; Yin, W.; Wang, S.; Jin, X.; Liu, X. Resource-Efficient Algorithms and Systems of Foundation Models: A Survey. *ACM Comput. Surv.* **2025**, *57*, 110. [\[CrossRef\]](#)
- Narayanan, D.; Harlap, A.; Phanishayee, A.; Seshadri, V.; Devanur, N.R.; Ganger, G.R.; Gibbons, P.B.; Zaharia, M. PipeDream: Generalized Pipeline Parallelism for DNN Training. In Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), Huntsville, ON, Canada, 27–30 October 2019.
- Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; Casas, D.d.; Hendricks, L.A.; Welbl, J.; Clark, A.; et al. Training Compute-Optimal Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022.
- Rosenfeld, J.S.; Rosenfeld, A.; Belinkov, Y.; Shavit, N. A Constructive Prediction of the Generalization Error Across Scales. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 26 April–1 May 2020.
- Narayanan, D.; Shoenybi, M.; Casper, J.; LeGresley, P.; Patwary, M.; Korthikanti, V.; Vainbrand, D.; Kashinkunti, P.; Bernauer, J.; Catanzaro, B.; et al. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), St. Louis, MO, USA, 14–19 November 2021. [\[CrossRef\]](#)
- Rajbhandari, S.; Rasley, J.; Ruwase, O.; He, Y. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Virtual, 9–19 November 2020.
- Huang, Y.; Cheng, Y.; Bapna, A.; Firat, O.; Chen, D.; Chen, M.; Lee, H.; Ngiam, J.; Le, Q.V.; Wu, Y.; et al. GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019.
- Patterson, D.; Gonzalez, J.; Hölzle, U.; Le, Q.; Liang, C.; Munguia, L.M.; Rothchild, D.; So, D.R.; Texier, M.; Dean, J. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *Computer* **2022**, *55*, 18–28. [\[CrossRef\]](#)
- Henderson, P.; Hu, J.; Romoff, J.; Brunskill, E.; Jurafsky, D.; Pineau, J. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *J. Mach. Learn. Res.* **2020**, *21*, 248.
- Wu, C.J.; Raghavendra, R.; Gupta, U.; Acun, B.; Ardalani, N.; Maeng, K.; Chang, G.; Aga, F.; Huang, J.; Bai, C.; et al. Sustainable AI: Environmental Implications, Challenges and Opportunities. In Proceedings of the Machine Learning and Systems (MLSys), Santa Clara, CA, USA, 11–14 April 2022.
- Chung, J.W.; Gu, Y.; Jang, I.; Meng, L.; Bansal, N.; Chowdhury, M. Reducing Energy Bloat in Large Model Training. In Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), Austin, TX, USA, 4–6 November 2024.

18. Xu, K.; Sun, D.; Tian, H.; Zhang, J.; Chen, K. GREEN: Carbon-Efficient Resource Scheduling for Machine Learning Clusters. In Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI), Philadelphia, PA, USA, 28–30 April 2025.
19. Espenshade, C.; Peng, R.; Hong, E.; Calman, M.; Zhu, Y.; Parida, P.; Lee, E.K.; Kim, M.A. 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 (EuroMLSys), Athens, Greece, 22 April 2024. [\[CrossRef\]](#)
20. Zhu, S.; Wang, D. Energy-Efficient LLM Training in GPU Datacenters with Immersion Cooling Systems. In Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems (e-Energy), Rotterdam, The Netherlands, 17–20 June 2025. [\[CrossRef\]](#)
21. Zhang, Z.; Strati, F.; Klimovic, A. Characterizing Energy and Performance for Distributed Training of Large Language Models. In Proceedings of the European Workshop on Machine Learning and Systems (EuroMLSys), Edinburgh, UK, 27 April 2026. [\[CrossRef\]](#)
22. Iftikhar, S.; Khan, H.; Davy, S. SustainFed-LLM: Renewable Energy Aware Client Selection for Energy Efficient Federated Training of Large Language Models. In Proceedings of the 28th European Conference on Artificial Intelligence (ECAI), Bologna, Italy, 25–30 October 2025.
23. Wu, R.; Chung, J.W.; Chowdhury, M. Kareus: Joint Reduction of Dynamic and Static Energy in Large Model Training. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), Seattle, WA, USA, 13–15 July 2026.
24. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *Commun. ACM* **2020**, *63*, 54–63. [\[CrossRef\]](#)
25. Tang, Z.; Wang, Y.; Wang, Q.; Chu, X. The Impact of GPU DVFS on the Energy and Performance of Deep Learning: An Empirical Study. In Proceedings of the Tenth ACM International Conference on Future Energy Systems (e-Energy), Phoenix, AZ, USA, 25–28 June 2019; pp. 315–325.
26. You, J.; Chung, J.W.; Chowdhury, M. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, USA, 17–19 April 2023.
27. Faiz, A.; Kaneda, S.; Wang, R.; Osi, R.; Sharma, P.; Chen, F.; Jiang, L. LLMCarbon: Modeling the End-to-End Carbon Footprint of Large Language Models. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
28. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLoRA: Efficient Finetuning of Quantized LLMs. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 10–16 December 2023.
29. Frantar, E.; Alistarh, D. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. In Proceedings of the International Conference on Machine Learning (ICML), Honolulu, HI, USA, 23–29 July 2023.
30. Korthikanti, V.A.; Casper, J.; Lym, S.; McAfee, L.; Andersch, M.; Shoeybi, M.; Catanzaro, B. Reducing Activation Recomputation in Large Transformer Models. In Proceedings of the Machine Learning and Systems (MLSys), Miami, FL, USA, 4–8 June 2023.
31. Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W.J. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
32. Wang, G.; Qin, H.; Jacobs, S.A.; Holmes, C.; Rajbhandari, S.; Ruwase, O.; Yan, F.; Yang, L.; He, Y. ZeRO++: Extremely Efficient Collective Communication for Large Model Training. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
33. Douillard, A.; Feng, Q.; Rusu, A.A.; Chhaparia, R.; Donchev, Y.; Kuncoro, A.; Ranzato, M.; Szlam, A.; Shen, J. DiLoCo: Distributed Low-Communication Training of Language Models. In Proceedings of the 2nd Workshop on Advancing Neural Network Training (WANT) at the International Conference on Machine Learning (ICML), Vienna, Austria, 27 July 2024.
34. Radovanovic, A.; Koningstein, R.; Schneider, I.; Chen, B.; Duarte, A.; Roy, B.; Xiao, D.; Haridasan, M.; Hung, P.; Care, N.; et al. Carbon-Aware Computing for Datacenters. *IEEE Trans. Power Syst.* **2023**, *38*, 1270–1280. [\[CrossRef\]](#)
35. Sukprasert, T.; Souza, A.; Bashir, N.; Irwin, D.; Shenoy, P. On the Limitations of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud. In Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys), Athens, Greece, 22–25 April 2024.
36. Acun, B.; Lee, B.; Kazhamiaka, F.; Maeng, K.; Gupta, U.; Chakkaravarthy, M.; Brooks, D.; Wu, C.J. Carbon Explorer: A Holistic Framework for Designing Carbon Aware Datacenters. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Vancouver, BC, Canada, 25–29 March 2023.
37. Bashir, N.; Gohil, V.; Belavadi, A.; Shahrad, M.; Irwin, D.; Olivetti, E.; Delimitrou, C. The Sunk Carbon Fallacy: Rethinking Carbon Footprint Metrics for Effective Carbon-Aware Scheduling. In Proceedings of the ACM Symposium on Cloud Computing (SoCC), Redmond, WA, USA, 20–22 November 2024. [\[CrossRef\]](#)
38. Luccioni, A.S.; Viguiet, S.; Ligozat, A.L. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *J. Mach. Learn. Res.* **2023**, *24*, 253.

39. mlco2. *CodeCarbon: Track and Reduce CO₂ Emissions from Computing*, Version 3.2.2; Mila (Quebec AI Institute): Montréal, QC, Canada, 2024. Available online: <https://github.com/mlco2/codecarbon> (accessed on 25 August 2026).
40. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, Canada, 24–28 June 2017.
41. Anthony, L.F.W.; Kanding, B.; Selvan, R. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. In Proceedings of the ICML Workshop on Challenges in Deploying and Monitoring Machine Learning Systems, Virtual, 17–18 July 2020.
42. Hu, Q.; Ye, Z.; Wang, Z.; Wang, G.; Zhang, M.; Chen, Q.; Sun, P.; Lin, D.; Wang, X.; Luo, Y.; et al. Characterization of Large Language Model Development in the Datacenter. In Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI), Santa Clara, CA, USA, 16–18 April 2024.
43. Du, N.; Huang, Y.; Dai, A.M.; Tong, S.; Lepikhin, D.; Xu, Y.; Krikun, M.; Zhou, Y.; Yu, A.W.; Firat, O.; et al. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In Proceedings of the International Conference on Machine Learning (ICML), Baltimore, MD, USA, 17–23 July 2022.
44. Fedus, W.; Zoph, B.; Shazeer, N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *J. Mach. Learn. Res.* **2022**, *23*, 120.
45. Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; et al. Mixed Precision Training. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
46. Narayan, S.; Gupta, A.; Paul, M.; Blalock, D. μ nit Scaling: Simple and Scalable FP8 LLM Training. In Proceedings of the International Conference on Machine Learning (ICML), Vancouver, BC, Canada, 13–19 July 2025.
47. Gruslys, A.; Munos, R.; Danihelka, I.; Lanctot, M.; Graves, A. Memory-Efficient Backpropagation Through Time. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Barcelona, Spain, 5–10 December 2016; Volume 29.
48. Zhao, Y.; Gu, A.; Varma, R.; Luo, L.; Huang, C.C.; Xu, M.; Wright, L.; Shojanazeri, H.; Ott, M.; Shleifer, S.; et al. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. *Proc. VLDB Endow.* **2023**, *16*, 3848–3860. [[CrossRef](#)]
49. Dao, T.; Fu, D.Y.; Ermon, S.; Rudra, A.; Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022.
50. Dao, T. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
51. Jia, Z.; Zaharia, M.; Aiken, A. Beyond Data and Model Parallelism for Deep Neural Networks. In Proceedings of the Machine Learning and Systems (MLSys), Stanford, CA, USA, 31 March–2 April 2019.
52. Zheng, L.; Li, Z.; Zhang, H.; Zhuang, Y.; Chen, Z.; Huang, Y.; Wang, Y.; Xu, Y.; Zhuo, D.; Xing, E.P.; et al. Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), Carlsbad, CA, USA, 11–13 July 2022.
53. Miao, X.; Wang, Y.; Jiang, Y.; Shi, C.; Nie, X.; Zhang, H.; Cui, B. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism. *Proc. VLDB Endow.* **2023**, *16*, 470–479.
54. Jia, X.; Jiang, L.; Wang, A.; Xiao, W.; Shi, Z.; Zhang, J.; Li, X.; Chen, L.; Li, Y.; Zheng, Z.; et al. Whale: Efficient Giant Model Training over Heterogeneous GPUs. In Proceedings of the USENIX Annual Technical Conference (ATC), Carlsbad, CA, USA, 11–13 July 2022.
55. Jiang, Z.; Lin, H.; Zhong, Y.; Huang, Q.; Chen, Y.; Zhang, Z.; Peng, Y.; Li, X.; Xie, C.; Nong, S.; et al. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. In Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI), Santa Clara, CA, USA, 16–18 April 2024.
56. Qiao, A.; Choe, S.K.; Subramanya, S.J.; Neiswanger, W.; Ho, Q.; Zhang, H.; Ganger, G.R.; Xing, E.P. Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), Virtual, 14–16 July 2021.
57. Yuan, B.; He, Y.; Davis, J.Q.; Zhang, T.; Dao, T.; Chen, B.; Liang, P.; Ré, C.; Zhang, C. Decentralized Training of Foundation Models in Heterogeneous Environments. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022.
58. McDonald, J.; Li, B.; Frey, N.; Tiwari, D.; Gadepally, V.; Samsi, S. Great Power, Great Responsibility: Recommendations for Reducing Energy for Training Language Models. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, USA, 10–15 July 2022.
59. Choi, S.; Koo, I.; Ahn, J.; Jeon, M.; Kwon, Y. EnvPipe: Performance-Preserving DNN Training Framework for Saving Energy. In Proceedings of the USENIX Annual Technical Conference (ATC), Boston, MA, USA, 10–12 July 2023.

60. Wang, F.; Zhang, W.; Lai, S.; Hao, M.; Wang, Z. Dynamic GPU Energy Optimization for Machine Learning Training Workloads. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 2943–2954. [[CrossRef](#)]
61. Wang, Y.; Hao, M.; He, H.; Zhang, W.; Tang, Q.; Sun, X.; Wang, Z. DRLCap: Runtime GPU Frequency Capping with Deep Reinforcement Learning. *IEEE Trans. Sustain. Comput.* **2024**, *9*, 712–726. [[CrossRef](#)]
62. Hanafy, W.A.; Liang, Q.; Bashir, N.; Irwin, D.; Shenoy, P. CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency. *Proc. ACM Meas. Anal. Comput. Syst.* **2023**, *7*, 57. [[CrossRef](#)]
63. Wiesner, P.; Behnke, I.; Scheinert, D.; Gontarska, K.; Thamsen, L. Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud. In Proceedings of the 22nd International Middleware Conference, Virtual, 6–10 December 2021.
64. Qian, C.; Kaminer, I.; Chen, H. A Guidance to Intelligent Metamaterials and Metamaterials Intelligence. *Nat. Commun.* **2025**, *16*, 1154. . Correction in *Nat. Commun.* **2025**, *16*, 11201. [[CrossRef](#)] [[PubMed](#)]
65. Xue, Z.; Zhou, T.; Xu, Z.; Yu, S.; Dai, Q.; Fang, L. Fully Forward Mode Training for Optical Neural Networks. *Nature* **2024**, *632*, 280–286. [[CrossRef](#)] [[PubMed](#)]
66. Fu, T.; Zhang, J.; Sun, R.; Huang, Y.; Xu, W.; Yang, S.; Zhu, Z.; Chen, H. Optical Neural Networks: Progress and Challenges. *Light. Sci. Appl.* **2024**, *13*, 263. [[CrossRef](#)] [[PubMed](#)]
67. Choi, M.; Xiang, J.; Wirth-Singh, A.; Baek, S.; Shlizerman, E.; Majumdar, A. Transferable Polychromatic Optical Encoder for Neural Networks. *Nat. Commun.* **2025**, *16*, 5623. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.