COLOMBO SCIENTIFIC PUBLISHING

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE & INFORMATION SYSTEM
E-ISSN: 2536-7919
P-ISSN: 2536-7900

DOI: - https://doi.org/10.55640/ijcsis/Volume11Issue02-06 **PAGE NO: 43-51**

Evolution of Architectural Approaches to Energy-Efficient Computing in Edge Artificial Intelligence Systems

Anton Budkevich

Chief Executive Officer, ShineDevelop LLC Miami, USA

ARTICLE INFO

Article history:

Submission: December 18, 2025
Accepted: January 17, 2026
Published: February 21, 2026
VOLUME: Vol.11 Issue 02 2026

Keywords:
edge artificial intelligence, energy efficiency, computing architecture, compact neural networks

ABSTRACT

This article examines the evolution of architectural approaches to ensuring energy-efficient computing in edge artificial intelligence systems operating under stringent resource constraints. The aim of the study is to analytically trace the transition from model-centric optimizations to multi-layer energy budget management that encompasses inference metrics, memory organization, data movement schemes, and connectivity with remote infrastructure. The relevance of the work is driven by the rapid growth in the number of edge devices, tightening requirements for latency, privacy, and autonomy, and the emergence of on-device generative models. The novelty of the article lies in a comprehensive reconstruction of the trajectory of architectural development, from compact neural networks, compression, and adaptive execution to hardware–software co-design, hybrid edge–cloud schemes, and the integration of event-driven and neuromorphic principles. It is shown that the key bottleneck is not arithmetic but memory and data transfers, while promising directions include energy-aware compilation and automated design under a given energy budget. The article will be of interest to researchers and engineers involved in the development of edge AI systems and energy-efficient computing architectures.

Introduction

Peripheral execution of artificial intelligence algorithms has become a natural response to a situation in which data are generated continuously and locally, and the value of a solution is often determined by reaction time. When processing is offloaded to remote data centers, transmission delays and network unpredictability are added to the computation, and in control, monitoring, and human–machine interaction tasks, this turns into an architectural risk. An additional driver is that moving computation closer to the data source simplifies compliance with privacy requirements: sensitive signals do not need to leave the device or local loop, thereby shrinking the leakage surface and reducing the need for continuous data exchange. In the same context, autonomy ceases to be an optional add-on and becomes a fundamental characteristic: the device must maintain operability and decision quality even under limited or absent connectivity, which is often highlighted in edge artificial intelligence surveys as one of the key drivers of development [1].

However, gains in latency and privacy are quickly constrained by the edge platform's resource ceiling. On a small device, energy budget, bandwidth, and memory capacity are constrained, while compute units are often specialized and non-universal; therefore, simply porting the model often results in degraded battery life and overheating, or an inability to place parameters and intermediate representations. This indicates that an entire direction targets microcontroller classes with nominal power consumption below 1 mW, but architectural decisions must then be adapted to strict limits on memory and the cost of each data access [2]. In a broader research context, it is emphasized that increasing network complexity and parameter counts directly raise compute- and memory-related requirements, making energy efficiency a key bottleneck for practical edge deployment [3].

This very contradiction, the need for local, fast, and privacy-preserving intelligence under simultaneous energy and hardware stringency, defines the subsequent logic of the article: architectures begin to evolve from model shrinking toward a deeper reconsideration of how computation and data movement are organized and how functionality is distributed between the software stack and hardware. This transition explains why the following sections consistently introduce compact networks, methods of compression and adaptive computation, joint hardware–software co-design and hybrid schemes, and then an interest in event-driven principles and neuromorphic architectures as a way to reshape the computation model itself under an energy budget [1].

Materials and Methodology

The study rests on a corpus of six specialized sources, including two survey studies on Edge AI and TinyML and four works focusing on contemporary practices of edge inference and the design of energy-oriented accelerators. The materials were selected so as to cover the entire path from motivation to mechanism: from formulating local inference as a response to latency, network uncertainty, and privacy constraints [1], to its engineering bottleneck, where limits on energy, memory, and bandwidth turn model porting into a problem of architectural redistribution of computations and data [2, 3]. The theoretical framework was formed through comparing perspectives on the key drivers of edge AI, encompassing reactivity and autonomy as systemic necessities, the energy barrier as the main constraint on practical deployment under growing parameter counts and network complexity [1, 3], and an additional focus on small machine intelligence as a distinct line of development in which extreme consumption and memory regimes become decisive for architectural choices [2, 6].

Methodologically, the work is constructed as a synthesis of an analytical review and comparative architectural analysis, in which sources are read not by topics but through causal links among computation, memory, and the real energy balance of a device. As a methodological core, the interpretation of energy efficiency in edge inference is used as a multicomponent quantity that cannot be reduced to the number of operations and requires accounting for memory, data transfers, and operating modes, an emphasis that is consistently maintained in surveys of edge inference and resource-constrained environments [4, 6]. To reconstruct the evolution of approaches, classes of architectural strategies and their dominant energy-saving mechanisms were compared: from methods designed for the strict limits of microcontrollers to broader schemes of inference optimization and hardware–software co-design, where the key factor is not only reducing computation but also managing dataflows and aligning the computational graph with the accelerator [2, 5]. In addition, recent consolidating work on the advancements and challenges of Edge Intelligence was taken into account to link local technical solutions with systemic deployment constraints (model complexity growth, robustness, and autonomy requirements) and thereby keep the discussion within an architectural rather than a purely local-optimization logic [3–4].

Results and Discussion

In edge artificial intelligence systems, energy efficiency is most reasonably interpreted as the ability to achieve a specified inference quality with minimal energy expenditure in a real deployment scenario, that is, with regard not only to arithmetic operations but also to memory, data exchange, and device operating modes. In practical terms, this definition is often expressed through performance per watt; however, for comparing models and platforms, a set of applied metrics is more convenient: energy per inference (joules per inference), energy per elementary operation (joules per operation), energy–delay product (as a measure of the trade-off between savings and performance), accuracy per joule (normalizing quality by cost), and latency as such as a reaction-time constraint.

This set must be supplemented by memory footprint metrics, parameter volume, peak working-memory demand, and access intensity, because memory often determines not only the energy profile but also the achievable speed on edge hardware. The logic of such a set aligns with the fact that modern surveys of edge inference directly consider the coupling of accuracy, latency, and energy costs as the central axis of architectural design [4].

In the energy balance of edge inference, raw computations rarely dominate: a significant share is contributed by data transfers across memory hierarchies and the movement of activations and weights between storage and compute units, and in networked scenarios, also by transmission costs, which depend on channel and protocol. A methodologically important implication for metrics is that the joules-per-inference estimate must include memory and communication costs. Otherwise, architectural comparisons become illusory, especially for models with irregular access or high sparsity. The background is no less significant: idle modes and prolonged standby sensing, in which total energy is determined not by peaks but by the time integral. Consequently, in accelerator-architecture research, the emphasis increasingly shifts to organizing dataflows and data reuse, since altering the pattern of data movement can yield gains unattainable through arithmetic optimization alone [5].

Historically, edge artificial intelligence began with relatively simple machine-learning methods on microcontrollers, where the decisive factor was fitting within stringent memory and power budgets. Yet, the exponential increase in the requirements of perception (images, audio, complex sensor streams) made deep neural networks inevitable. The use of cloud computing, which entails sending raw data to data centers, reveals its bottlenecks (latency, connectivity, bandwidth, and privacy) and redefines on-device smart inference as both an architectural requirement and a compelling use case, rather than a luxury. As a result, an agenda of energy-oriented architectures emerged, in which minimizing energy and latency is treated as a coequal objective alongside accuracy, and small machine intelligence became a distinct direction, capturing the shift from cloud dependence to autonomous on-device inference [6]. The foundations of energy efficiency in edge AI are presented in **Figure 1.**

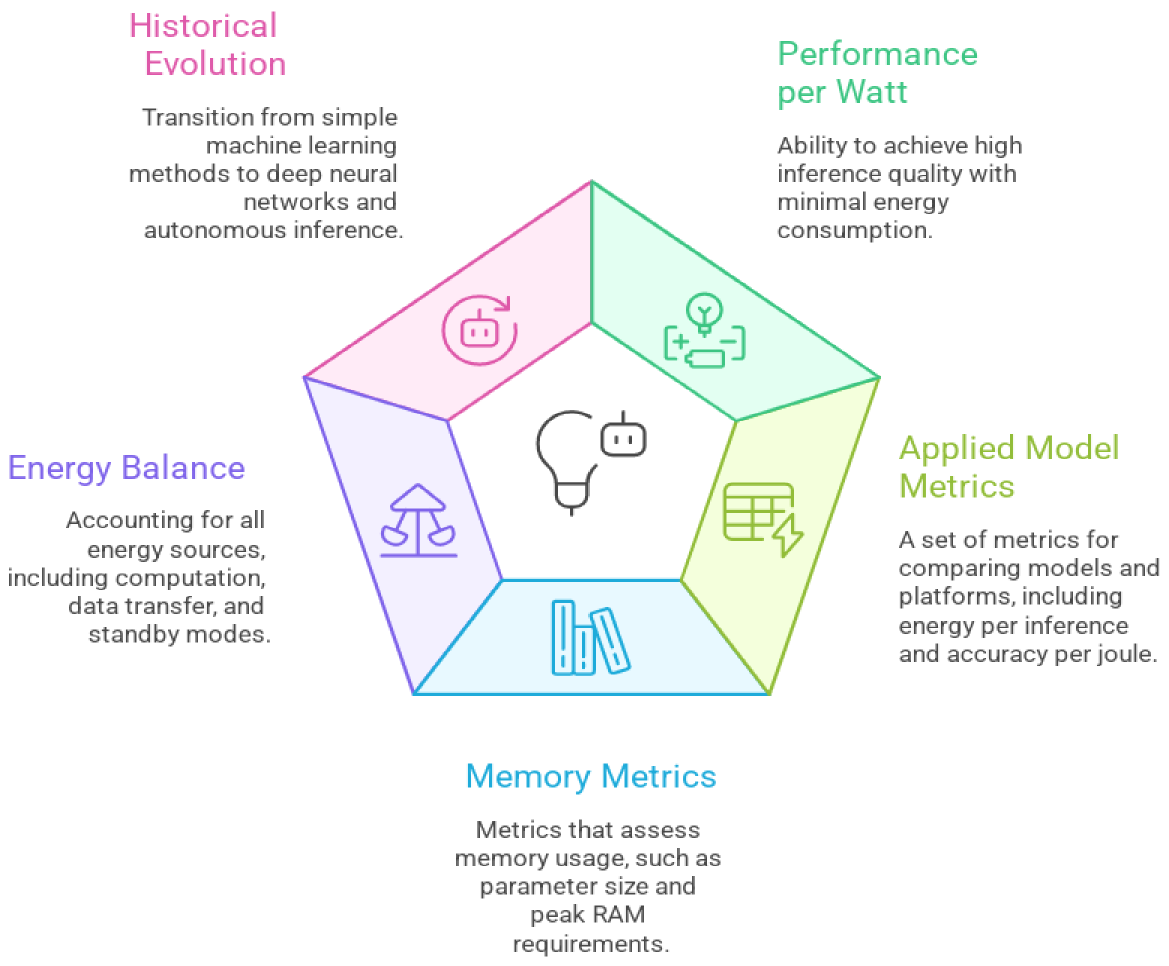
Historical Evolution
Transition from simple machine learning methods to deep neural networks and autonomous inference.
Performance per Watt
Ability to achieve high inference quality with minimal energy consumption.
Energy Balance
Accounting for all energy sources, including computation, data transfer, and standby modes.
Applied Model Metrics
A set of metrics for comparing models and platforms, including energy per inference and accuracy per joule.
Memory Metrics
Metrics that assess memory usage, such as parameter size and peak RAM requirements.

Fig. 1. Fundamentals of Energy Efficiency in Edge AI

Compact neural network architectures emerged as a response to the tension between the need for sophisticated perception and the strict resource constraints at the edge. Their basic idea is straightforward: preserve the model's useful expressiveness while simultaneously reducing the number of parameters and computations so that the cost of a single inference becomes commensurate with the device's energy budget. An important nuance is that smaller does not automatically mean worse: with an appropriate computational structure, a network can concentrate effort precisely where it contributes most to quality.

The first layer of techniques involves rethinking convolutional transformations as the dominant mechanism of feature extraction. Instead of heavy operations, constructs are used that separate processing across channels and spatial dimensions, reduce connectivity, and thereby decrease cost. This is not mere cosmetic pruning but a change in the geometry of computation: the network begins to move less data and access memory less frequently, which is often more important than a pure reduction in the number of multiplications.

The second layer comprises lightweight blocks, in which heavy components are replaced by chains of simpler operations with intermediate normalization and nonlinearities. Such blocks better match constrained compute units and fit into the cache more easily, thereby reducing data-movement penalties. At the same time, modularity is preserved: models can be assembled for different budgets, and scaling is supported without complete network redesign.

A third direction within compact architectures is reducing network depth. Network width is also reduced as a controlled trade-off. Decreasing the width reduces the memory channel and bandwidth load. Decreasing the depth decreases the number of steps and impacts latency. It can be more important at the edges than tuning within the model, as the question becomes whether the model will fit in memory or whether the device can be driven to that operating point.

Typical application scenarios for compact networks involve constant readiness and predictable latency. These include recognition of simple events in an audio stream, local frame filtering, key-object detection, sensor-based state estimation, and situations where the device must react without network access. In such tasks, a compact architecture functions not as a cut-down version but as a deliberately engineered instrument that provides sufficient quality at an acceptable cost.

However, architectural compactness does not span the entire task space, particularly when a model has already been chosen or cannot be radically altered due to compatibility or quality requirements. In this context, architectures based on compression and adaptation emerge, reducing computational and memory cost while preserving the original structure or modifying it minimally. This is important in practice because it enables migrating models originally designed for more powerful conditions to edge platforms.

Compression here includes lowering the numeric precision of weights and activations, removing low-salience connections and layers, training a compact student to reproduce the behavior of a larger teacher, and transitioning to sparse representations. The intent is to make the model lighter in terms of storage and computation without sacrificing its generalization ability. At the same time, different compression methods interact with hardware in different ways: techniques that effectively reduce model size may impair access regularity and thus raise memory-related energy costs.

Computation adaptation adds another degree of freedom: the model does not always execute everything, but performs just enough to reach a confident decision. This is implemented via early exiting, dynamic depth and width variation, and conditional activation of blocks based on the input. As a result, energy consumption becomes dependent on signal complexity, and in real streams, where most situations are typical, the average energy can drop substantially.

The limitations of this approach stem from two sources. The first is accuracy: any compression or conditionality increases the risk of degradation in rare, complex cases where the system must not fail. The second is deployment complexity: sparsity and conditional branching complicate compilation, profiling, and execution-time predictability, potentially degrading latency stability and increasing overhead.

To keep energy systematically under control, model-level optimization alone is usually insufficient, because a significant share of energy is spent on data movement. An edge device operates within a memory hierarchy in which accesses to different levels have radically different costs, and poor data organization can nullify the gains from compression. Consequently, the next stage in evolution is hardware–software co-design, where the model is treated as part of a stack rather than an isolated object.

Co-design can be conveniently conceptualized as a continuous chain of transformations from the computational graph structure to the behavior of the actual accelerator. At the model level, operations and block forms are chosen to match available compute primitives. At the compiler level, the graph is rewritten to reduce intermediate buffers and promote data reuse. At runtime, the system handles scheduling, memory allocation, and dataflow coordination. Frequencies, voltages, and sleep modes are configured at the accelerator and power-management levels; together, these define the actual cost of inference.

Common optimizations used in co-design include fusing operations into a single pass, appropriately ordering tensors in memory to avoid excessive permutations, batching or streaming operations based on latency tolerances, and dynamically adjusting frequency and power consumption based on performance needs. However, these optimization methods are rarely used in isolation. For example, operation fusion may reduce memory accesses at the cost of register pressure, streaming may reduce peak memory usage at the cost of synchronization overhead, and frequency scaling may reduce energy consumption at the cost of increased latency. Architecture in this sense becomes the art of assembling a compromise.

In real systems, an additional layer is added to this picture: hybrid architectures in which computation is distributed between the edge device and the remote infrastructure. The principle of partitioning is organized around the question of which operations must be executed immediately and locally, and which can be deferred and offloaded. The local part typically provides responsiveness, privacy, and robustness to connectivity loss, whereas the remote part provides high accuracy, complex models, and heavy computation.

Partitioning can occur at different levels: the device can compute features and send a more compact representation, or it can execute early layers and transmit intermediate activations. Partitioning may act as a trigger that activates remote analysis only in suspicious events, or as a filter that discards most routine situations. The core idea is to reduce the volume of transmitted data and network calls while maintaining high quality where it truly matters.

The trade-offs of hybrid schemes are fundamental and multidimensional: energy savings on the device can lead to increased traffic; reduced latency of the local response may require sacrificing a complex remote component; strengthening privacy may require retaining more computation on the device. Reliability also comes into play: the network may be unavailable, and the architecture must then support a degraded yet safe operating mode. In this sense, a hybrid system is a mechanism for managing quality under connectivity constraints.

Against this background, neuromorphic computing appears as an attempt to alter the paradigm itself and bring computation closer to the regime in which sensory streams operate. The event-driven principle asserts that computation should occur not on a fixed schedule but in response to significant changes. In periods of stability, the device does almost nothing, and energy is spent not on repeatedly performing identical operations but only on processing new content. Spiking encoding formalizes this principle: information is represented as sparse events in time rather than continuous value arrays.

Hardware implementations of the event-driven approach rely on asynchrony and locality: numerous simple elements process events in parallel and independently, and computation propagates through the network only where activity exists. Architecturally, memory and computation are tightly coupled: state storage and updates occur in close proximity, rather than through constant accesses to distant memory. This reduces data movement and improves the energy profile for long-term autonomous operation.

The application of neuromorphic systems is especially natural in robotics and sensing devices, where both reaction speed and frugality are important. Event-based cameras and similar sensors generate streams of changes rather than frames, immediately reducing redundancy and easing the compute load. In such systems, architecture benefits twice: less data at the input and activation of computation only in regions with dynamics.

The advantages of the neuromorphic approach can be summarized in four properties: asynchrony, locality, noise robustness, and an inherent capacity to handle temporal dependencies. Asynchrony removes the need for global clocking of computation, and locality lowers memory cost. Noise robustness arises because random fluctuations may not generate meaningful events, and temporal dependencies are accounted for by the representational format itself. From an energy perspective, this implies that the computational cost scales with the information content rather than its length.

Training spiking networks is more difficult, and tools for developing, debugging, and diagnosing them are less mature. There is lower cross-platform support. Finally, there are interoperability issues with existing software stacks. Adding a neuromorphic component to an ecosystem based on classical networks means building bridges and converters, as well as new tests.

To connect the architectures described with practice, models of energy consumption are needed that translate design decisions into measurable consequences. Analytical models usually describe energy as the sum of contributions from computation, memory, and data exchange, linking energy to frequency, voltage, operation counts, and data-transfer volumes. Their value lies in enabling comparisons at an early stage, when devices do not yet exist, or experimental tuning is prohibitively expensive.

Empirical models rely on measurements and profiling. They also rely on observing system behavior under real operating regimes. Such models capture overheads, driver effects, scheduler peculiarities, caching effects, and actual sleep modes that analytical approaches can overlook. At the edge, empiricism is particularly important because of long idle periods: if a device listens most of the time and rarely performs inference, total energy is determined not only by the cost of a single inference but also by how frugally the always-on mode is organized. Table 1 links architectural classes to the energy-balance component they primarily target and to a typical side effect.

Table 1. Correspondence Between Architectural Approach, Dominant Energy Source, and Typical Constraint

Approach	Main source of energy savings	Typical limitation
Compact networks	Fewer computations and data transfers	Quality ceiling on complex data
Compression and sparsity	Less memory use and arithmetic	Deployment complexity and irregular memory access
Adaptive execution	Fewer steps on average	Unpredictable latency and risk of errors on rare cases
Hardware–software co-design	Less data movement	Dependence on the stack and platform
Edge–cloud hybrid	Lower local costs	Network dependence and privacy concerns
Neuromorphic systems	Event-driven computation	Immature tools and training methods

Generative models and large language models at the edge create a distinct tension line, because their cost manifests not in a single inference act but in sequential generation, where each subsequent step depends on the previous ones. For such models, memory and state accumulated over time are critical, and this is precisely what makes migration to the edge difficult, even with moderate parameter sizes. An additional complication is that latency is perceived by the user as a stream, which means the system must maintain a steady output rate rather than compute quickly once.

Architectural strategies for generative models derive from the principles described above. Compression reduces memory use and accelerates individual operations, and partitioned computation enables offloading the computationally intensive parts to remote infrastructure. At the same time, local private processing is preserved, and local assistants are organized so that most requests are resolved on-device, with only rare complex cases delegated. Schemes in which the edge component is responsible for preliminary understanding, filtering, and safe execution, and the remote component is responsible for high-accuracy generation when connectivity is available, are of particular importance.

The scenarios where generative models would be deployed on edge devices are mostly those that require autonomy and privacy and can afford local suggestions, on-device explanations for local data, offline local control assistance, and closed loops for sensitive queries. They almost inevitably require a degradation pattern. In the absence of connectivity, the system must also operate at reduced capacity but expand to full capacity when connectivity is present. This directly links generative models to hybrid schemes and to the idea of an adaptive budget.

All the directions described inevitably encounter architectural trade-offs that admit no universal resolution. Accuracy competes with energy; latency competes with energy savings in sleep regimes; generality competes with fine-grained optimization for a particular accelerator; independence from the network competes with access to more powerful compute and larger models. Crucially, these axes do not form a single line: improving one dimension can simultaneously degrade two others, turning design into a multicriteria problem. Strategies for generative models at the edge are presented in Figure 2.

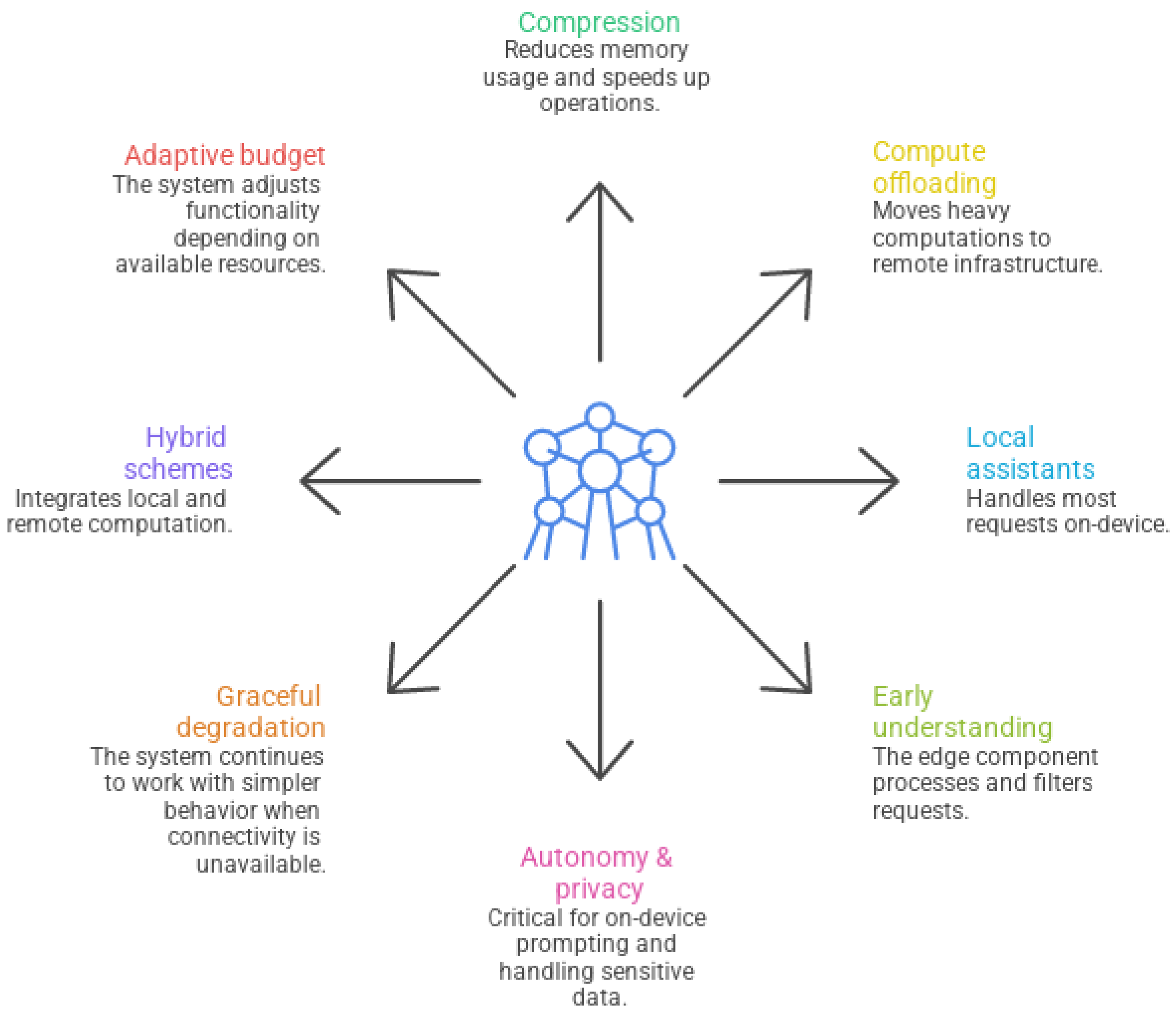
Compression
Reduces memory usage and speeds up operations.
Adaptive budget
The system adjusts functionality depending on available resources.
Compute offloading
Moves heavy computations to remote infrastructure.
Hybrid schemes
Integrates local and remote computation.
Local assistants
Handles most requests on-device.
Graceful degradation
The system continues to work with simpler behavior when connectivity is unavailable.
Early understanding
The edge component processes and filters requests.
Autonomy & privacy
Critical for on-device prompting and handling sensitive data.

Fig. 2. Strategies for Generative Models at the Edge

Development prospects follow naturally from this map: further progress will be determined by how deeply models, compilers, runtimes, and hardware modes can be interlinked so that energy is managed as deliberately as accuracy. Energy-oriented compilation ceases to be an auxiliary optimization and becomes the central mechanism for transferring architectural decisions into a real device, where the costs of memory, movement, and waiting differ by orders of magnitude.

The next expected step is the automation of design under an energy budget, where the search for model structure and configuration will take device constraints into account as strictly as quality constraints. What matters here is the transition from isolated successful architectures to systematic selection that yields reproducible results across platforms and operating regimes, including prolonged standby and rare bursts of activity.

Convergence between neuromorphic principles and traditional architectures is likely to unfold not through complete replacement but through incremental borrowing of event-driven ideas in specific parts of the stack. This may appear as event-based input representations, hybrid blocks activated only upon changes, and new operating modes for sensors and compute units in which the system wakes up on informative signals. Such a path lowers deployment barriers by enabling the gradual integration of event-driven mechanisms without disrupting the entire ecosystem.

Conclusion

The presented work captures a fundamental shift in the understanding of edge artificial intelligence: local inference is no longer seen as a particular engineering choice but as an architectural response to three interrelated determinants: latency and network unpredictability, privacy requirements, and the need for autonomy. Yet the same locality immediately reveals the edge platform's resource ceiling: constraints on energy, memory, and bandwidth shift the problem from the plane of model porting to the plane of managing how computation and data movement are organized. In this sense, energy efficiency is treated in the article both substantively and operationally: as achieving a given quality with minimal energy in real operating regimes, with metrics inevitably extending beyond arithmetic to include memory, communication, latency, and even prolonged standby. This formulation places at the center of design the trade-off among accuracy, latency, and energy, in which memory and the movement of activations/weights often become coequal, and sometimes dominant, sources of cost, rendering joules-per-inference estimates methodologically vulnerable if communication is ignored.

The logic of architectural evolution is reconstructed in the text as a progressive deepening of intervention. The trajectory runs from compact networks to controlled reductions in depth and width, then to compression, sparsity, and adaptive execution, where energy depends on input complexity, but the risk of degradation in rare cases increases, and latency predictability becomes more difficult. The emphasis then naturally shifts to hardware–software co-design, since it is precisely the organization of dataflows and memory reuse that can deliver gains unattainable by arithmetic alone, and subsequently to edge–cloud hybrid schemes as mechanisms for managing quality under connectivity and privacy constraints. The most radical line comprises event-driven and neuromorphic principles that propose scaling computational cost with information content rather than observation length, but currently faces immature training methods and tooling. In the final outlook, this map of tensions crystallizes around generative models at the edge, where the cost manifests in sequential generation and state accumulation; hence, energy-oriented compilation and automated design under energy budgets, as well as soft integration of event-driven mechanisms into the traditional stack via hybrid modes that wake on informative signals, become especially valuable.

References

1. R. Singh and S. S. Gill, "Edge AI: A survey," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, Mar. 2023, doi: https://doi.org/10.1016/j.iotcps.2023.02.004.

2. W. Raza, A. Osman, F. Ferrini, and F. D. Natale, "Energy-Efficient Inference on the Edge Exploiting TinyML Capabilities for UAVs," *Drones*, vol. 5, no. 4, p. 127, Oct. 2021, doi:

https://doi.org/10.3390/drones5040127.

3. T. Wang, J. Guo, B. Zhang, G. Yang, and D. Li, "Deploying AI on Edge: Advancement and Challenges in Edge Intelligence," *Mathematics*, vol. 13, no. 11, p. 1878, Jun. 2025, doi: https://doi.org/10.3390/math13111878.

4. D. Ngo, H.-C. Park, and B. Kang, "Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments," *Electronics*, vol. 14, no. 12, p. 2495, Jun. 2025, doi: https://doi.org/10.3390/electronics14122495.

5. Raha, D. A. Mathaikutty, S. Kundu, and S. K. Ghosh, "FlexNPU: a dataflow-aware flexible deep learning accelerator for energy-efficient edge devices," *Frontiers in High Performance Computing*, vol. 3, Jun. 2025, doi: https://doi.org/10.3389/fhpcp.2025.1570210.

6. S. Heydari and Q. H. Mahmoud, "Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions," *Sensors*, vol. 25, no. 10, p. 3191, May 2025, doi: https://doi.org/10.3390/s25103191.