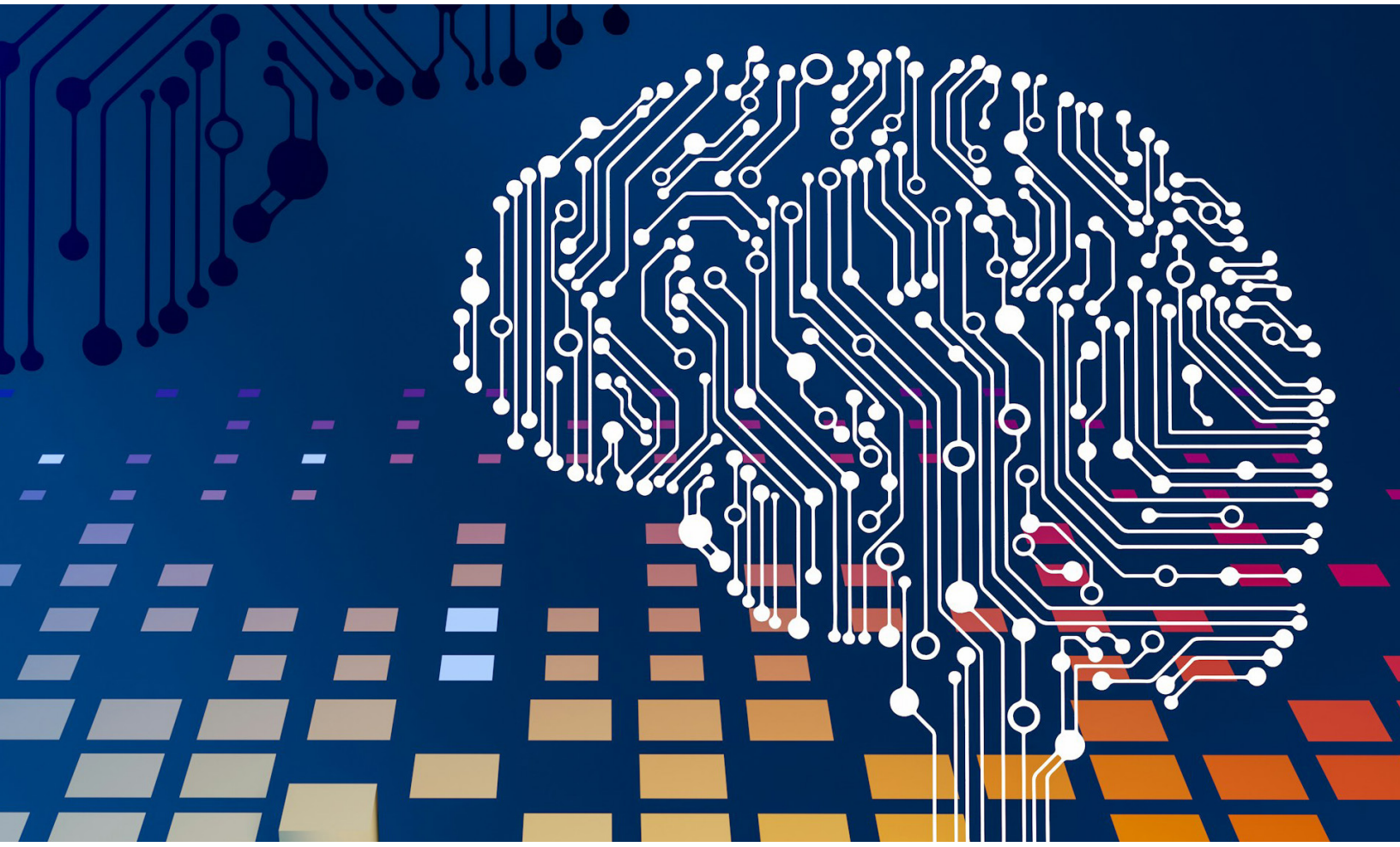
Right-Sizing Edge AI Compute:
A Five-Step Engineering Selection Framework



Artificial intelligence (AI) is changing the expectations for embedded and edge systems. Applications that once collected data for review are now tasked with inspecting images, detecting objects, predicting failures, combining sensor inputs, and supporting automated decisions at the point where data is created.

This shift makes hardware selection more complex. Choosing an AI-ready platform is not simply a matter of finding the most powerful processor or the graphics processing unit (GPU) with highest advertised TOPS (Trillions of Operations Per Second). The right compute architecture depends on the workload, the data inputs, the response time, the operating environment, and the scaling needs of the system over time.

When planning for or designing new hardware, the most effective place to start is not the processor. It is the application and workload.

Start With the Application, Not the Processor

Modern edge systems often combine control, sensor acquisition, communications, storage, networking, and AI inference. Selecting compute, therefore, requires more than comparing CPU benchmarks, GPU specifications, or accelerator TOPS.

The right approach is the one that can sustain the complete application within its timing, data-movement, power, thermal, environmental, software, and lifecycle constraints. This five-step framework helps engineering teams narrow the viable approaches before detailed processor comparison and platform benchmarking begin.

Five Steps to Narrow the Architecture

1. Identify the Dominant Workload

Define the system's dominant workload, then identify the workloads that must coexist with it and the resources they share. A vision system may also require control logic, camera synchronization, storage, networking, and supervisory functions. The architecture must support the combined workload, not just the AI function.

Separate deployed inference from model development. If training or substantial fine-tuning must occur locally, treat it as a distinct high-performance requirement rather than sizing it as a typical inference workload.

2. Characterize the Computational Profile

Timing and determinism: Define average and worst-case latency and identify functions with hard deadlines. Real-time behavior depends on the complete hardware/software path, not processor class alone.

Parallelism and inference: Identify operations that benefit from parallel execution. Quantify model type, precision, inference rate, concurrency, and acceptable response time.

I/O, data movement, and memory: Calculate aggregate camera, sensor, network, storage, and expansion bandwidth. Confirm that interfaces, PCIe lanes, DMA paths, and memory bandwidth can feed the processing resources at the required rate.

Sensor synchronization and fusion: For multi-sensor systems, define timing alignment, preprocessing, and movement of data between acquisition, CPU, accelerator, and output stages.

3. Define Deployment Constraints

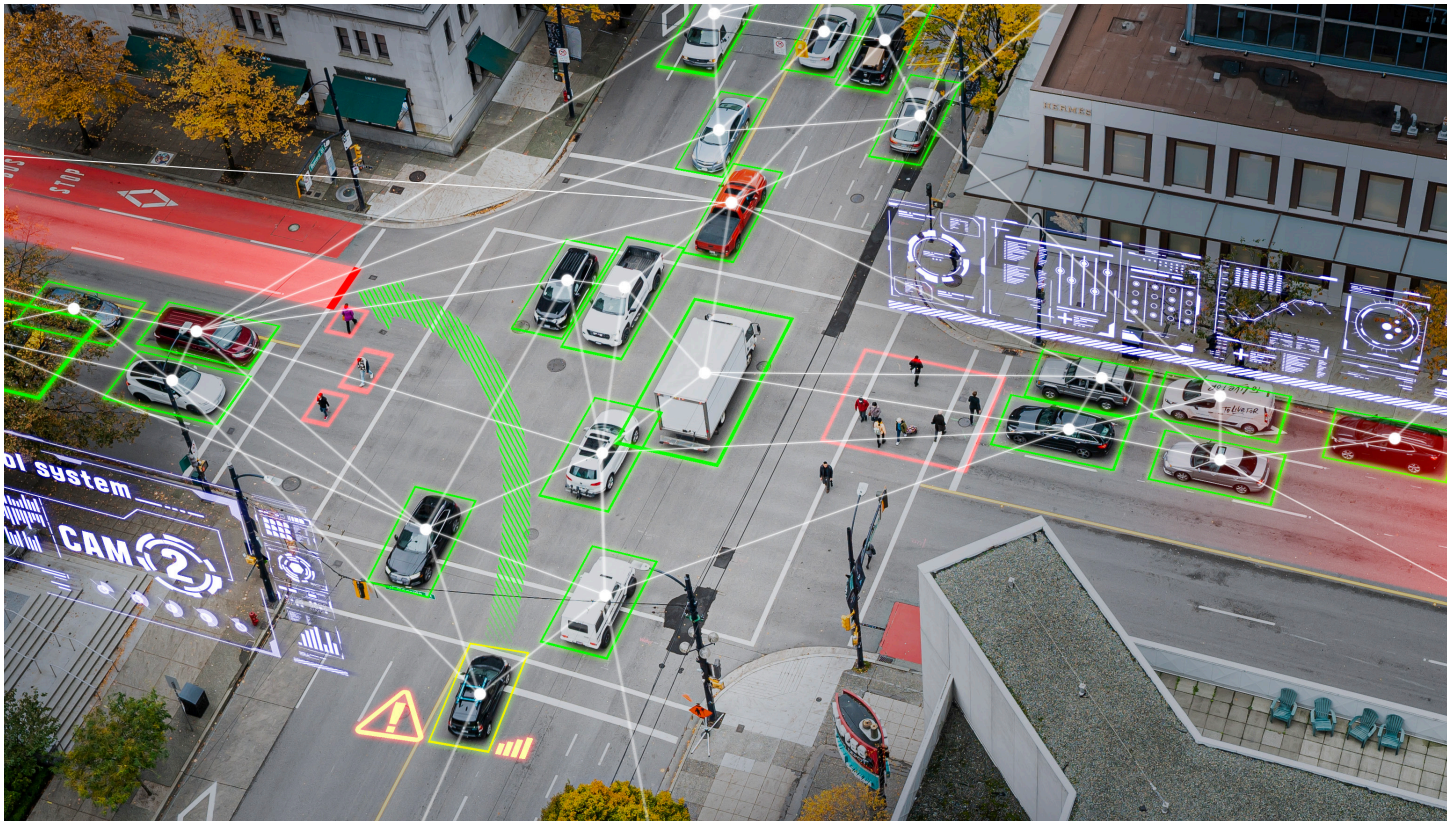
Set the continuous power budget, cooling approach, worst-case ambient temperature, enclosure limits, and environmental requirements. Evaluate sustained performance in the intended deployment condition. Unused compute capability can still create power, thermal, mechanical, and cost penalties.

4. Establish Lifecycle and Software Requirements

Define product life, driver support, operating system and framework dependencies, security maintenance, certification considerations, and migration strategy. In long-lived systems, software continuity and module availability can matter as much as initial compute performance.

5. Map Requirements to a Compute Approach and System Architecture

Only after the application is characterized should the team compare compute approaches. CPU, GPU, and dedicated accelerators describe how work is processed. COM Express, COM-HPC, SMARC, and other modular approaches describe how compute is integrated, customized, expanded, and migrated over time. These decisions are related, but they are not interchangeable.



Compute / System Approach	Applications	Risks
CPU	Control, communications, data acquisition, protocol handling, HMI, or gateway functions dominate.	Adding acceleration the workload cannot use; overlooking I/O, real-time behavior, or lifecycle.
CPU + Dedicated AI Accelerator	The host already fits the application and a defined inference workload needs efficient acceleration.	Model/operator support, conversion workflow, host-to-accelerator data movement, utilization, and future model changes.
Integrated GPU / AI platform	Vision, detection, sensor fusion, robotics, or multimodal inference need substantial parallel compute in a compact platform.	Sizing from TOPS alone; sustained thermals, memory bandwidth, camera ingest, framework dependencies, and lifecycle.
Modular COM Architecture	Application-specific I/O, scalable compute, PCIe expansion, processor migration, or lifecycle planning are major requirements.	Selecting the module standard before carrier I/O, thermal, expansion, and migration requirements are defined.
Rugged Integrated System	Environmental, connector, power-input, sealing, shock, vibration, or temperature requirements materially shape the compute design.	Treating ruggedization as an enclosure step instead of a system-level design constraint.

Two Architecture Examples

NVIDIA Jetson: An integrated heterogeneous platform that combines CPU, GPU, memory, high-speed interfaces, and an edge AI software ecosystem. It is a strong candidate when vision, sensor fusion, parallel processing, or inference are central requirements. System fit still depends on the complete data path, carrier I/O, power mode, thermal design, and lifecycle.

COM-HPC: A modular computer-on-module standard for high-performance embedded systems. Its value is not that it is inherently an AI architecture, but that it supports high-speed I/O, expansion, scalable processing, application-specific carrier design, and processor migration. It can be paired with CPU- or accelerator-centric solutions.

Validate the Complete System

- Benchmark the actual model, framework, precision, input resolution, and concurrency on representative hardware.
- Test sustained performance at the intended power mode and worst-case ambient condition.
- Validate the full data path from acquisition through preprocessing, inference, post-processing, storage, networking, and control output.
- Confirm the software stack early: BSP, drivers, frameworks, deployment tooling, security maintenance, and update strategy.
- Retest the architecture against realistic growth in model size, sensor count, camera resolution, I/O, and processing rate.

Engineering Selection Checklist

1. What workload dominates, and what other workloads must coexist with it?
2. What average, worst-case, and deterministic timing requirements apply?
3. What model, precision, inference rate, parallelism, and memory footprint are required?
4. How much data enters and leaves the system, and where could I/O or memory movement become the bottleneck?
5. What power, cooling, temperature, environmental, and physical constraints define deployment?
6. What OS, drivers, frameworks, toolchains, and security-maintenance requirements must be supported?
7. How long must the platform remain available, and what is the migration or upgrade strategy?
8. Has the architecture been tested with the real workload and full I/O profile rather than synthetic benchmarks alone?

Right-Size the System, Not Just the Compute

Edge AI is fundamentally a system-level engineering challenge. The processor or accelerator is only one part of an architecture that must move data, execute inference, manage control functions, communicate with other systems, operate within thermal and power limits, and remain supportable throughout its intended lifecycle.

The most effective approach is to begin with the application. Characterize the workload, data path, deployment environment, software requirements, and lifecycle expectations before selecting a compute architecture. Then validate the complete system using the actual models, interfaces, workloads, and operating conditions the application will encounter.

The objective is not to maximize compute. It is to right-size compute: provide enough validated performance margin to meet today's requirements while maintaining a practical path for tomorrow's workload growth.

Start With the Workload

Before selecting a processor, accelerator, or compute module, evaluate the complete application against the five-step framework. Identify the dominant workload, quantify the data path and timing requirements, define deployment constraints, establish lifecycle and software requirements, and then map those requirements to the appropriate compute and system architecture.

Sealevel Systems can help engineering teams evaluate the I/O, compute, thermal, mechanical, and lifecycle requirements that shape an edge computing architecture. Start with your application requirements and build the compute architecture around them.