Review

Not peer-reviewed version

Embodied Foundation Models at the Edge: A Survey of Deployment Constraints and Mitigation Strategies

Utkarsh Grover , Ravi Ranjan , Mingyang Mao , Trung Tien Dong , Satvik Praveen , Zhenqi Wu , Morris Chang , Tinoosh Mohsenin , Yi Sheng , Agoritsa Polyzou , Eiman Kanjo , Xiaomin Lin *

Posted Date: 17 March 2026

doi: 10.20944/preprints202603.1293.v1

Keywords: embodied foundation models; edge AI; embodied AI; edge deployment; vision language action models; diffusion policies; real time control; system co design; memory bandwidth; compute latency; scheduling; safety critical systems

 doi:10.20944/preprints202603.1293.v1

Review

Embodied Foundation Models at the Edge: A Survey of Deployment Constraints and Mitigation Strategies

Utkarsh Grover [1], Ravi Ranjan [2], Mingyang Mao [3], Trung Tien Dong [3], Satvik Praveen [3], Zhenqi Wu [3], J. Morris Chang [4], Tinoosh Mohsenin [3], Yi Sheng [5], Agoritsa Polyzou [6], Eiman Kanjo [7] and Xiaomin Lin [3,*]

[1] University of South Florida, Tampa, FL 33620, USA
[2] Florida International University, Miami, FL 33199, USA
[3] University of South Florida, Tampa, FL 33620, USA
[4] Johns Hopkins University, Baltimore, MD 21218, USA
[5] Florida International University, Miami, FL 33199, USA
[6] Nottingham Trent University, Nottingham, NG1 4FQ
[7] Imperial College London, London, UK
* Correspondence: xiaominlin01@gmail.com

Abstract

Deploying foundation models in embodied edge systems is fundamentally a systems problem, not just a problem of model compression. Real-time control must operate within strict size, weight, and power constraints, where memory traffic, compute latency, timing variability, and safety margins interact directly. The Deployment Gauntlet organizes these constraints into eight coupled barriers that determine whether embodied foundation models can run reliably in practice. Across representative edge workloads, autoregressive Vision-Language-Action policies are constrained primarily by memory bandwidth, whereas diffusion-based controllers are limited more by compute latency and sustained execution cost. Reliable deployment therefore depends on system-level co-design across memory, scheduling, communication, and model architecture, including decompositions that separate fast control from slower semantic reasoning.

Keywords: embodied foundation models; edge AI; embodied AI; edge deployment; vision language action models; diffusion policies; real time control; system co design; memory bandwidth; compute latency; scheduling; safety critical systems

1. Introduction

The migration of Foundation Models (FMs) [1] from hyperscale data centers to resource-constrained edge platforms fundamentally alters how intelligence is executed, not merely where inference occurs [2,3]. Cloud-based models are designed around assumptions of elastic power availability, abundant cooling, and relaxed latency constraints; embodied systems invalidate these assumptions by requiring execution within strict Size, Weight, and Power (SWaP) envelopes imposed by autonomous mobile robots (AMRs), aerial platforms, and wearables [4]. As a result, deploying foundation models on the edge is not a compression problem alone, but a systems orchestration challenge [5], in which high bandwidth, memory-bound inference must operate reliably under thermally unstable conditions, battery-limited power budgets, and hard real-time constraints.

This execution regime exposes Moravec's Paradox as a governing systems constraint: while high-level reasoning has become increasingly tractable, robust sensorimotor integration remains computationally expensive and architecturally brittle [6]. Real-world autonomy therefore demands omni-modal perception that fuses the correlated physics of light (vision), sound (audio), and geometry (LiDAR) under strict temporal alignment to maintain control fidelity [7]. When multi-rate, asynchronous sensor streams are coupled with large, memory-intensive foundation models, the interaction

 doi:10.20944/preprints202603.1293.v1

stresses the limits of embedded silicon, producing failure modes rooted in timing, bandwidth, and synchronization that do not arise in disembodied or cloud-based intelligence.

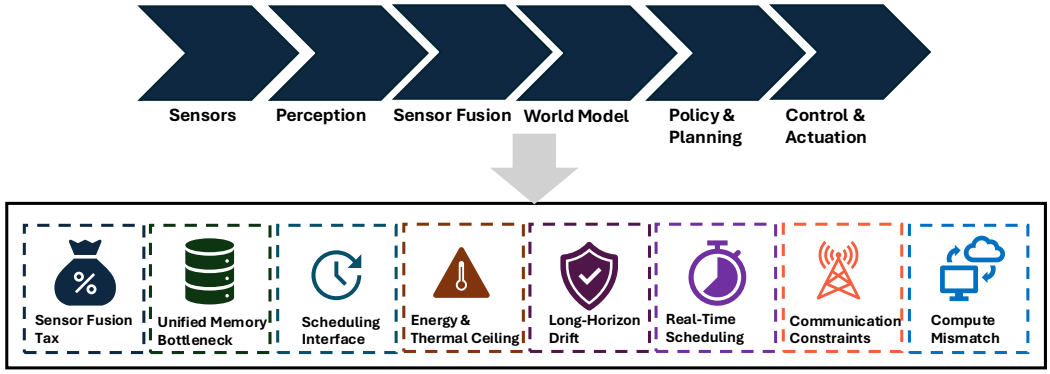

Figure 1. The Deployment Gauntlet. A unified view of the eight major system barriers that limit the deployment of foundation models from the cloud to edge embodied AI platforms.

1.1. The Omni-Modal Imperative

Closed loop autonomy requires world representations that can be updated and acted upon within tight latency bounds. Vision Language Action architectures such as RT 2 and OpenVLA seek to meet this requirement by collapsing perception, semantic grounding, and policy generation into a single computational graph [8,9]. This unification simplifies training and interfaces, but it also couples sensing, reasoning, and control into a monolithic inference path with strict execution demands. A representative 7B parameter VLA requires sustained high bandwidth weight streaming, dense attention computation, and fine grained synchronization across heterogeneous sensor inputs [10]. These demands exceed the latency, bandwidth, and scheduling guarantees typically available on edge platforms.

On embedded accelerators such as the NVIDIA Jetson AGX Orin and Qualcomm RB5, this mismatch appears in modality specific ways. Sparse LiDAR workloads stall dense tensor cores and often require inefficient CPU mediated scatter gather operations [11,12]. Transformer based audio models can exceed the SRAM capacity of low power DSPs, forcing expensive off chip memory access [13,14], while high resolution visual encoders generate bursty traffic that saturates LPDDR channels and triggers thermal throttling [15,16]. More broadly, the *Omni Model Imperative* that favors large modality agnostic foundation models in server class settings [17,18] becomes a liability at the sensor edge, where parameter scale, deep attention stacks, and autoregressive decoding amplify memory pressure, latency variance, and scheduling instability [19]. These interlocking constraints define the *Deployment Gauntlet*, a set of system barriers that prevent the theoretical capabilities of foundation models from translating into reliable real time embodied behavior.

1.2. The Deployment Gauntlet

Existing approaches to deploying foundation models on constrained hardware primarily focus on local optimizations such as quantization and pruning to reduce inference cost [20,21]. Although effective for static workloads, these methods do not resolve the dominant failures of embodied deployment, where sensing, computation, memory, timing, and control interact under closed loop execution. In such systems, a violation in any one dimension can cascade across the stack. To capture this challenge, we define the ***Deployment Gauntlet***, a systems taxonomy of the conditions that foundation models must jointly satisfy to operate reliably in real time embodied environments.

Figure 1 illustrates the eight barriers that make up the Deployment Gauntlet. These barriers are not heuristic categories, but arise from a structured decomposition of the embodied intelligence pipeline, from multimodal sensing and data movement through heterogeneous computation and shared memory to time critical control, long horizon operation, safety enforcement, and edge to cloud coordination. Each represents a distinct and irreducible class of failure, and because embodied systems

 doi:10.20944/preprints202603.1293.v1

are tightly coupled in time and energy, stress in one dimension often destabilizes the others. Together, they provide a minimal and nonredundant account of the system constraints that govern real time embodied AI.

The Deployment Gauntlet comprises the following eight coupled barriers, ordered along the sensing execution control continuum:

1. *The Sensor Fusion Tax:* Latency and jitter introduced by temporally aligning asynchronous sensor streams, including middleware-induced serialization and buffering overheads (e.g., ROS 2) [22, 23].
2. *Heterogeneous Compute Mismatch:* Inefficiencies that arise when sparse or irregular workloads are mapped onto accelerators optimized for dense, static matrix operations [24].
3. *Unified Memory Bottleneck:* Contention between high-bandwidth model weight streaming and continuous sensor ingestion over shared memory channels [25,26].
4. *Energy & Thermal Ceiling:* Performance collapse caused by thermal throttling and accelerated battery depletion during sustained operation [27].
5. *Long-Horizon Drift:* Progressive degradation of state estimation, calibration, and clock synchronization over extended deployments [28].
6. *Safety & Verification Gap:* Irreversible failure modes induced by hallucinated states or out-of-distribution behavior within safety-critical control loops [29,30].
7. *Real-Time Scheduling Interference:* Control instability driven by OS jitter, non-deterministic kernel execution, and contention in shared accelerators [31].
8. *Communication Constraints:* Bandwidth and latency limits that restrict edge-to-cloud offloading and multi-agent coordination [32].

1.3. Contributions

This survey frames the deployment of edge foundation models as a fundamental problem of *embodied AI execution* rather than isolated model compression. By evaluating prior work through the lens of closed-loop systems, we make explicit how physical constraints timing, energy, memory, and control govern the realizability of edge intelligence. To this end, our core contributions are:

First, in Section 2, we formalize a **Landscape of Foundation Models**. We analyze how distinct workloads (vision, LiDAR, audio) interact with embedded hardware along orthogonal axes like memory bandwidth, sparsity, and thermal stability. This reveals modality specific incompatibilities routinely obscured by static benchmarks.

Second, in Section 3, we introduce the ***Deployment Gauntlet***, a systems level framework derived from the end-to-end physical intelligence loop. The Gauntlet identifies a non-redundant set of interdependent barriers that arise during real-time execution, unifying recurring failure modes such as thermal throttling, synchronization jitter, and long-horizon drift that sever benchmark-level performance from time-sensitive physical operation.

Finally, in Section **??**, we **synthesize existing mitigation strategies**, including pipelined sensing, hardware-aware distillation, and multi-rate inference scheduling. Demonstrating that incremental optimization cannot resolve the structural constraints of embodied execution, we chart an architectural roadmap arguing that bifurcated System 1/System 2 designs and neuromorphic sensing are necessary evolutions beyond conventional von Neumann limits. Figure 3 gives an overview of these challenges and solutions.

 doi:10.20944/preprints202603.1293.v1

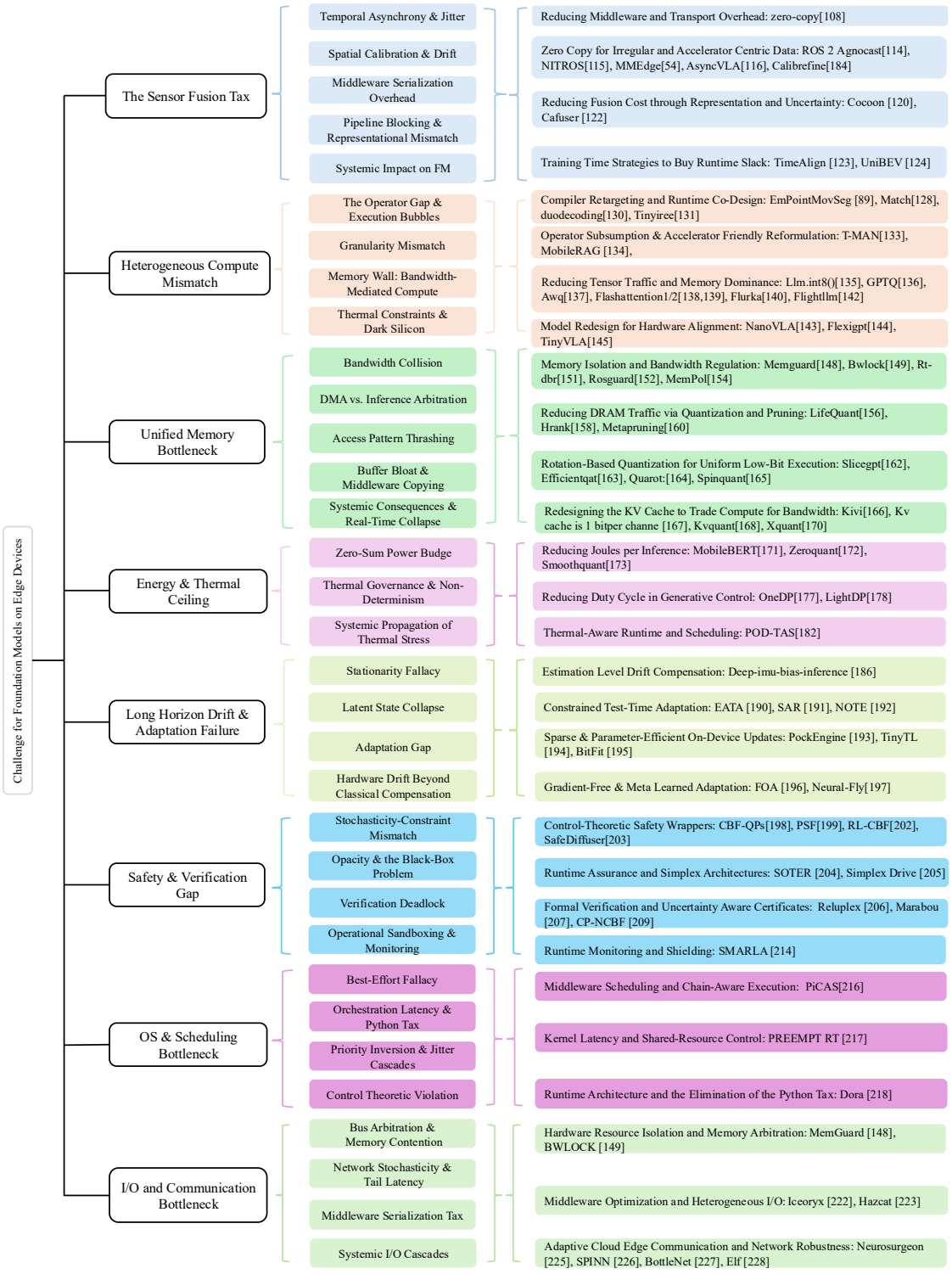

Figure 2. The taxonomy for the gauntlets and solutions in Foundation Models under Embodied Constraints.

2. Landscape of Foundation Models on the Edge

The deployment of foundation scale models on the edge dictates a structural shift from monolithic end-to-end execution to a two-tier architecture: server resident cognitive backbones paired with compressed, real time edge executors. Because closed loop control and irreversible physical interaction preclude cloud only inference, strict execution budgets enforce this paradigm. In consumer systems, Apple's iOS 18 "Intelligence" couples a server resident model with a ∼3-billion-parameter on-device counterpart, relying on LoRA adapters and mixed 2-bit/4-bit quantization (averaging roughly 3.7 bits per weight) to maintain interactive latency [33]. Similarly, Google's *Gemini Robotics On-Device* executes

 doi:10.20944/preprints202603.1293.v1

vision-language-action (VLA) policies entirely on-board for bimanual manipulation without network dependence [34].

Robotic systems impose even stricter execution regimes. While foundational policies like PaLM-SayCan and RT-2 successfully generate task plans and semantic grounding [8,35], their real-time execution exposes the limits of monolithic inference. Deployed systems require extensive distillation and hardware specialization to meet latency and power constraints, as demonstrated by the diffusion transformer based Large Behavior Models on Boston Dynamics' Atlas [36] and the deployable on-board inference engines of NVIDIA's R^2D^2 sim-to-real workflow [37].

Across these domains, a consistent principle governs embodied AI execution: predictable latency and safe interaction cannot be guaranteed by unconstrained generative inference. Practical embodied intelligence demands rigorous co-design across model architecture, training strategy, and embedded hardware [33]. Motivated by these constraints, we taxonomize edge-relevant workloads according to their dominant resource demands, memory bandwidth, compute intensity, and I/O pressure to evaluate their architectural compatibility with Size, Weight, and Power (SWaP)-constrained platforms.

2.1. Taxonomy of Edge Relevant Workloads

Embodied AI execution at the edge demands a workload level understanding that extends beyond aggregate metrics such as parameter count or FLOPs [24]. We therefore classify foundation model workloads along three orthogonal axes that directly determine real time feasibility in closed loop, embodied systems. The first axis specifies the *Primary Role* of the model within the control loop, distinguishing perceptual models that estimate state from policy models that generate actions [38,39]. The second axis captures the *I/O Modality*, spanning dense, high throughput inputs such as RGB video to sparse, irregular signals such as LiDAR point clouds [40]. The third axis identifies the *Dominant Bottleneck* imposed during execution memory capacity and bandwidth, sustained compute demand, or I/O throughput [41,42].

These axes are intentionally minimal and non redundant: each isolates a distinct source of execution pressure that cannot be inferred from model scale alone. Together, they explain why architectures with similar parameter counts or nominal compute budgets exhibit qualitatively different behavior when deployed under Size, Weight, and Power (SWaP) constraints [19].

2.2. Vision-Language-Action (VLA) Policies

Vision-Language-Action (VLA) [43,44] policies constitute a uniquely demanding workload in embodied AI by explicitly entangling high-level semantic reasoning with time-critical control. Architectures such as RT-2 [8] and OpenVLA [9] jointly tokenize observations and actions, producing discrete control sequences such as end-effector poses or skill tokens directly from RGB inputs and natural language commands. While this architectural unification enables broad task generalization, it collapses deliberative inference and real-time actuation into a single execution path, creating severe resource contention and undermining determinism at the edge. For instance, OpenVLA (7B) [9] demands a $\sim$14 GB FP16 memory footprint, saturating the DRAM capacity of unified-memory systems-on-chip (SoCs) like the Jetson Orin NX and leaving insufficient headroom for operating system services or sensor buffering.

In edge deployments, VLA execution is constrained primarily by memory bandwidth rather than peak compute throughput. Autoregressive decoding repeatedly accesses massive weight tensors for each generated action token, driving sustained DRAM traffic. On embedded platforms, this traffic competes directly with high-rate camera and LiDAR DMA streams, oversubscribing the memory controller and triggering latency spikes that violate real-time control deadlines [42]. This execution-time variance renders monolithic VLA inference fundamentally incompatible with deterministic, closed-loop operation.

Consequently, edge-oriented architectures enforce a strict separation between semantic reasoning and low-level control. Systems such as NanoVLA [45] route high-level task planning to a heavyweight vision-language model while delegating high-frequency ($\sim$50 Hz) control to a lightweight visual

 doi:10.20944/preprints202603.1293.v1

policy. This bifurcation amortizes transformer inference, caps sustained memory pressure, and yields order-of-magnitude speedups (e.g., 52× on representative edge hardware) without sacrificing task expressiveness. Ultimately, under physical AI constraints, VLA deployability is governed by two structural mandates: aggressive quantization and distillation to throttle memory traffic, and the architectural isolation of slow, deliberative reasoning (System 2) from time-critical control loops (System 1) [35].

2.3. Diffusion-Based Policies

Diffusion-based policies constitute a compute-intensive execution regime in embodied AI by generating continuous control trajectories through iterative denoising [46]. Rather than emitting discrete action tokens, these models synthesize smooth, multi-step motion sequences conditioned on observations, rendering them highly effective for contact-rich manipulation. While architectures like Octo [47] exemplify this paradigm's capacity for generalist, multi-modal behavior, their iterative structure imposes a severe constraint: producing a single action trajectory typically requires 10 to 100 denoising steps. This establishes a latency floor fundamentally misaligned with the isochronous control frequencies required by real-time systems.

Consequently, under edge constraints, diffusion policies are inherently compute-bound and energy-intensive. Each denoising step executes dense matrix multiplications that saturate accelerator cores for extended durations. Unlike memory-bound VLA policies, diffusion workloads exert sustained compute and power pressure, accelerating battery depletion and triggering thermal throttling on mobile platforms. Deployed naively, multi-step diffusion controllers achieve control rates of merely 1–2 Hz grossly insufficient for stable, closed-loop manipulation.

Edge-oriented architectures therefore seek to collapse or amortize this iterative inference process. OneDP [48] distills multi-step diffusion into a single-step generator, increasing control frequency from ∼1.5 Hz to over 60 Hz on representative edge hardware. Similarly, LightDP [49] constrains compute demand through structured pruning and architectural simplification to remain within strict mobile thermal envelopes. These approaches, however, expose a fundamental trade-off: reducing denoising steps improves responsiveness and energy efficiency strictly at the cost of generative fidelity.

Within embodied systems, the diffusion step count emerges as a first-class execution parameter directly governing control smoothness, latency, and energy consumption. While step reduction enables real-time operation, it does not alter the underlying compute-bound physics of the workload. As a result, diffusion-based policies are best suited for low-frequency planning, trajectory refinement, or short-horizon generation or explicitly integrated into hierarchical architectures that decouple high-rate control from slower, generative inference.

2.4. Vision Encoders and Multimodal LMMs

Vision encoders and multimodal large language models (LMMs) impose a distinct execution regime in Embodied AI , dominated by the cost of ingesting high dimensional sensory streams at video rates. While these models provide the perceptual substrate for planners and controllers, their primary constraint at the edge is not steady state inference but repeated visual *prefill*. Large-scale encoders such as CLIP-ViT-L deliver strong semantic grounding [13], yet their quadratic attention complexity renders naive deployment at frame rates on the order of 30 FPS computationally and thermally infeasible on resource constrained platforms. As a result, perception workloads are governed by the prefill stage required to process high resolution visual tokens.

Edge oriented perception pipelines therefore restructure visual ingestion to amortize or collapse prefill cost. LLaVA-Mini [50] compresses visual inputs into a single token prior to language model ingestion, sharply reducing prefill latency while preserving multimodal reasoning capability. MiniCPM-V [51] demonstrates that architectural hyper parameters explicitly tuned for mobile deployment coupled with targeted data curation can approach GPT-4V-level performance on edge devices without increasing model scale. Complementary analyses such as ViT-Edge [52] systematize encoder

 doi:10.20944/preprints202603.1293.v1

side optimizations, including low rank factorization and structured compression, that reduce compute and memory footprint while maintaining representational fidelity.

From an execution perspective, vision encoders are inherently *prefill bound* and thermally constrained. Continuous high resolution video streams induce bursty compute demand during frame ingestion, producing transient thermal spikes that dominate the power envelope of fanless edge modules [42]. When visual tokens cannot be reused or incrementally updated across frames, the encoder becomes the principal source of thermal throttling, limiting sustained operation even when downstream reasoning and control are lightweight. Consequently, practical Embodied AI systems rely on token reuse, temporal caching, and frame rate decimation to bound prefill cost and maintain stable real time perception.

2.5. Efficient Segmentation Algorithm

Promptable segmentation models, such as the Segment Anything Model (SAM) [53], impose a uniquely burst-driven execution regime in embodied AI. While these architectures deliver strong generalization for object delineation and affordance discovery, their reliance on heavyweight vision transformers precludes continuous, closed-loop execution. Even with edge-oriented variants like MobileSAM [54] and EdgeSAM [55] which distill the ViT encoder into lightweight CNN-ViT hybrids to enable on-device inference segmentation remains strictly an interaction-triggered operation. Each request forces a full encoder forward pass, generating a per-invocation compute burst that violates thermal and latency budgets if sustained at video rates.

Consequently, segmentation is fundamentally incompatible with per-frame execution inside high-frequency robotic control loops. Instead, embodied systems must invoke these models opportunistically, typically during scene initialization, object discovery, or grasp proposal decoupling rich perception from continuous actuation. This execution profile reflects a strict structural constraint at the edge: deep semantic scene understanding must be applied selectively to preserve the deterministic timing and real-time responsiveness of the underlying platform.

2.6. 3D & LiDAR Encoders

3D and LiDAR encoders define a sparsity-bound execution regime in Embodied AI, where irregular point distributions and dynamic spatial density dominate runtime behavior. Unlike dense vision pipelines, these workloads are governed by non contiguous memory access and data dependent control flow, which violate the throughput and locality assumptions underlying modern accelerators [11]. PointPillars remains a widely adopted baseline by projecting sparse point clouds into pseudo-images, enabling partial reuse of 2D CNN backbones and existing hardware support [56].

This projection mitigates, but does not eliminate, the fundamental cost of sparsity. DensePillarNet [57] restructures the backbone to favor dense layer execution, reducing end to end latency by up to 30% on constrained devices. However, voxelization and feature aggregation remain unavoidable sources of irregular memory access. In multi-modal fusion pipelines such as BEVFusion [58], large intermediate feature maps must be scattered and gathered across CPU and GPU boundaries, introducing synchronization overhead and latency variance that erode deterministic real time execution.

From an execution standpoint, 3D perception workloads are intrinsically *sparsity bound*. Voxelization and point aggregation thrash caches, stall SIMD units, and prevent efficient utilization of dense matrix accelerators [12]. Fixed function inference engines, such as NVIDIA's Deep Learning Accelerator (DLA), lack native support for sparse indexing and dynamic data structures, forcing LiDAR workloads onto general purpose GPUs. As a result, even moderately sized 3D perception stacks can monopolize shared compute and memory bandwidth, directly interfering with time critical control loops.

Consequently, practical Embodied AI systems cannot treat 3D perception as a continuously executing pipeline. Sparse encoders are routinely downsampled, temporally decimated, or invoked at lower frequencies than vision, reflecting a structural limitation imposed by sparsity and memory irregularity rather than deficiencies in model architecture.

 doi:10.20944/preprints202603.1293.v1

2.7. Audio & Speech Foundation Models

Audio and speech foundation models characterize the always-on regime of Embodied AI, where continuous situational awareness must be maintained under strict power, memory, and latency constraints. Unlike vision or LiDAR pipelines, audio systems are expected to operate persistently to support wake-word detection, speech understanding, and anomaly monitoring, thereby imposing stringent requirements on timing jitter, memory residency, and energy efficiency [59].

Transformer-based ASR models, such as Whisper and its compressed variants, are increasingly deployed on-device through optimized runtimes including whisper.cpp [60], with recent work also considering incremental and personalized adaptation under domain shift [61]. Although their overall FLOP counts may be lower than those of vision backbones, their execution is dominated by long temporal sequences and repeated attention over audio frames.

From a systems perspective, these workloads are inherently *sequence bound*. Transformer-based ASR must preserve and process extended temporal context, often exceeding the SRAM capacity of low-power DSPs used in conventional audio pipelines [62]. Once activations spill beyond on-chip memory, execution shifts to the CPU or NPU, incurring off-chip memory accesses that substantially increase power consumption and violate milliwatt-scale always-on budgets.

Consequently, full-capacity speech models remain impractical for continuous edge deployment in embodied systems. Their use is therefore typically constrained to streaming, chunking, early-exit strategies, or event-triggered activation. More broadly, audio foundation models illustrate that workloads with modest nominal compute demand can still become dominant bottlenecks when persistent execution is required under tight energy, memory, and timing limits.

2.8. Multimodal Fusion Stacks

Multimodal fusion stacks define the execution regime of Embodied AI by combining heterogeneous perceptual streams into a temporally consistent world state for reasoning and control [63]. Rather than introducing a distinct computation pattern, fusion couples the execution properties of all modalities, making it the point where timing, memory, and scheduling constraints converge.

Recent systems such as MMEdge [64] replace sequential fusion pipelines with pipelined sensing that overlaps sensor ingestion, encoding, and fusion. Although this improves average throughput, it does not guarantee deterministic execution. Once modalities are coupled, overall behavior is governed less by component-level efficiency than by system-level scheduling.

From a systems perspective, multimodal fusion is fundamentally *scheduling bound*. Its performance depends on middleware latency, executor design, and operating-system preemption [23]. On unified-memory SoCs, fusion also becomes the point of contention among high-rate sensor DMA, model weight streaming, and intermediate activation buffers sharing LPDDR bandwidth [65]. Small disturbances, including delayed callbacks, temporary memory pressure, or executor backlog, can therefore propagate across modalities and produce temporal misalignment in downstream control.

These interactions create failure modes not present in unimodal pipelines. Even when individual models satisfy latency targets independently, their combined execution through shared middleware and memory can violate end-to-end real-time constraints. Multimodal fusion thus emerges as a final systems bottleneck in Embodied AI, where limitations arise primarily from synchronization, scheduling, and resource contention. Reliable deployment consequently requires co-design across perception, middleware, and operating-system scheduling rather than isolated model or accelerator optimization.

3. The Deployment Gauntlet

Section 2 showed that embodied foundation model workloads differ sharply in their dominant execution bottlenecks. In deployment, however, these workloads rarely operate in isolation. Memory intensive VLA policies, compute-heavy diffusion controllers, burst-driven visual encoders, and synchronization sensitive fusion pipelines are typically co-located on the same embedded system-on-chip

 doi:10.20944/preprints202603.1293.v1

(SoC), where they compete for shared memory bandwidth, thermal headroom, power budget, and scheduling priority [65]. As a result, end-to-end failures often emerge not from any single model component, but from their interaction under shared physical constraints.

This interaction is the basis of the *Deployment Gauntlet*. We use the term to denote a systems taxonomy of the recurring barriers that embodied foundation models must satisfy to operate reliably on edge platforms. These barriers appear across platforms, modalities, and application domains [38] because they arise from a common deployment setting: closed-loop execution under tight constraints on memory, compute, I/O, timing, power, and safety. When these pressures compound, systems exhibit latency variance, deadline misses, synchronization failures, and thermal throttling, even when individual components remain tractable in isolation. Figure 3 summarizes the Deployment Gauntlet as a systems taxonomy of the eight coupled barriers that shape reliable embodied foundation model execution at the edge.

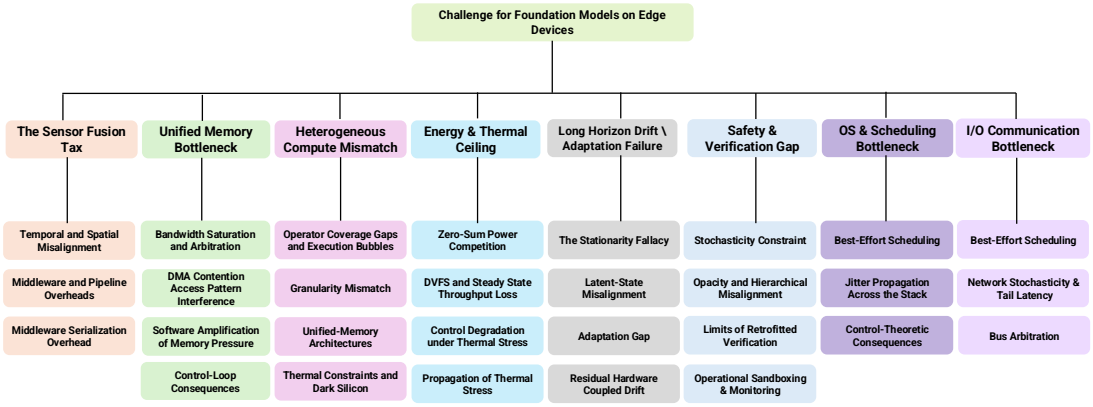

Figure 3. The taxonomy for the gauntlets and solutions in Foundation Models under Embodied Constraints.

The Deployment Gauntlet is therefore not a list of isolated implementation defects, nor is it exhausted by local optimizations such as pruning, quantization, or accelerator specialization [24]. Instead, it organizes the recurring systems constraints that emerge when embodied foundation models must sustain real-time perception, inference, and control on shared edge hardware. The remainder of this section develops the Gauntlet through eight coupled barriers, each corresponding to a distinct but strongly interacting source of deployment failure. Table 1 (Barriers I-IV) and Table 2 (Barriers V-VIII), that emerge repeatedly across platforms, modalities, and application domains [38]

3.1. The Sensor Fusion Tax

The **Sensor Fusion Tax** is the latency, jitter, and memory overhead incurred when heterogeneous, asynchronous sensor streams are transformed into time-aligned, control-ready representations. In embodied systems, this overhead can consume a substantial fraction of the control-cycle budget. It does not originate from a single bottleneck. Instead, it accumulates across temporal synchronization, spatial calibration, middleware orchestration, and representational translation. Foundation models intensify this burden because they add sustained memory traffic, increase sensitivity to timing error, and enlarge the execution graph that must remain stable under closed-loop operation.

3.1.1. Temporal and Spatial Misalignment

Robotic sensing is inherently asynchronous: cameras, LiDAR, and IMUs operate on different clock domains and exhibit variable transport delay. Enforcing temporal consistency therefore requires buffering and synchronization mechanisms such as ROS 2's *ApproximateTime* filter. These improve alignment, but they also increase end-to-end latency and reduce control responsiveness. Li et al. [66] show that synchronization can idle downstream computation while waiting for delayed packets, reducing effective throughput. In high-speed manipulation, even millisecond-scale jitter can produce substantial divergence between perceived and true system state [67]. Spatial fusion introduces a

 doi:10.20944/preprints202603.1293.v1

second source of overhead. Calibration degrades under vibration, thermal expansion, and mechanical drift, and corrective methods such as LCCNet [68] and CalibRefine [69] add nontrivial computation to the control loop. The resulting trade-off is straightforward: better alignment often consumes more of the available control-cycle budget.

3.1.2. Middleware and Pipeline Overheads

Middleware compounds these costs. ROS 2 enables modular message passing, but serialization and deserialization remain expensive even in intra-process settings, with reported latency penalties of up to 50% relative to shared-memory designs [23]. For high-bandwidth modalities such as LiDAR and 4K video, repeated copying consumes memory bandwidth that would otherwise be available to inference. Non-deterministic scheduling further creates queue buildup and jitter spikes on the order of 10–40 ms [70]. Conventional fusion pipelines impose an additional constraint: they often block downstream inference until a complete sensor sweep has arrived, effectively tying control frequency to the slowest sensing modality. Pipelined systems such as MMEdge [64] reduce average latency by overlapping sensing and inference, but they do not remove the underlying synchronization and scheduling burden.

3.1.3. Representational Mismatch and Foundation Model Sensitivity

Representational mismatch introduces a third source of overhead. Event cameras, for example, produce sparse, microsecond-resolution streams that do not map naturally to frame-based foundation models. Converting these streams into dense tensors through reconstruction or voxelization can offset part of the energy advantage of neuromorphic sensing and increase CPU-side preprocessing overhead when dedicated support is unavailable [71,72]. Foundation models then amplify the fusion burden in three ways. First, sustained parameter movement can saturate LPDDR bandwidth on unified-memory systems and reduce headroom for sensor buffering and transport [2]. Second, end-to-end VLA policies can be highly sensitive to temporal misalignment, so synchronization error propagates directly into action quality [73]. Third, practical control rate is often bounded not by nominal inference throughput, but by the slowest stage in the sensing and synchronization path [74].

3.1.4. Implications

The Sensor Fusion Tax is a recurring systems constraint in embodied deployment, not a narrow implementation artifact. Once perception is embedded in a real-time control stack, synchronization, calibration, and data movement become first-order determinants of latency stability, memory pressure, and closed-loop reliability.

 doi:10.20944/preprints202603.1293.v1

Table 1. Deployment Gauntlet barriers 1–4: recurring system-level challenges for embodied foundation models at the edge.

Barrier	Representative Sub-problem	Systems Effect
Sensor Fusion Tax	Temporal and Spatial Misalignment	Asynchronous sensor clocks, transport delay, and calibration drift force buffering, synchronization, and corrective alignment, trading control responsiveness for temporal and spatial consistency [66–69].
	Middleware and Pipeline Overheads	Serialization, copying, buffering, and sweep-level pipeline blocking consume bandwidth and introduce latency jitter that destabilizes high-frequency control execution even when average throughput is acceptable [23,64,70].
	Representational Mismatch and Foundation-Model Sensitivity	Modalities such as event streams require costly reconstruction or voxelization before they can be processed by frame-based models, while foundation models further amplify fusion overhead through higher memory traffic and greater sensitivity to timing error [2,71–74].
Heterogeneous Compute Mismatch	Operator Coverage Gaps and Execution Bubbles	Limited operator support on edge accelerators forces graph partitioning, CPU fallback, and host–device synchronization, creating execution bubbles and large latency spikes that reduce end-to-end throughput [45,75].
	Granularity Mismatch and Kernel Launch Overhead	Fine-grained transformer kernels incur substantial launch overhead on embedded GPUs, accounting for 30–60% of latency and driving utilization below 20% during autoregressive decoding [76].
	The Memory Wall under Unified-Memory Architectures	Shared LPDDR bandwidth contention among CPUs, GPUs, NPUs, and sensors makes execution memory-bound and can offset nominal hardware-acceleration gains through arbitration and tensor movement overhead [9,77].
	Thermal Constraints and Dark Silicon	Sustained multimodal workloads trigger DVFS throttling, reduce steady-state throughput, and make nominal peak compute difficult to maintain during prolonged execution [78,79].
Unified Memory Bottleneck	Bandwidth Saturation and Arbitration	Concurrent FM parameter streaming and high-rate sensor ingestion saturate shared LPDDR bandwidth, so accelerators stall because memory delivery, rather than arithmetic throughput, becomes the limiting factor [76,80,81].
	DMA Contention and Access Pattern Interference	Sensor DMA traffic competes directly with accelerator memory access, while mismatched access patterns reduce cache efficiency and increase latency variance across perception and control loops [65,77].
	Software Amplification of Memory Pressure	Middleware buffering, serialization, and large FM weight footprints amplify memory pressure, triggering reallocations, evictions, and other memory-management events that reduce throughput and destabilize execution [9].
	Control-Loop Consequences	Memory stalls delay accelerators, block control threads, increase thermal load, and degrade real-time behavior before nominal compute limits are reached [20,70,78].
Energy & Thermal Ceiling	Zero-Sum Power Competition	Foundation-model inference competes directly with propulsion, actuation, and sensing for a fixed onboard energy budget, reducing endurance under sustained execution [76,82,83].
	DVFS and Steady-State Throughput Loss	High-duty-cycle workloads exceed passive cooling capacity and trigger DVFS transitions that reduce sustained throughput once the platform reaches thermal equilibrium [84].
	Control Degradation under Thermal Stress	Reduced steady-state inference rate increases perception staleness and lowers closed-loop responsiveness by slowing state updates and control decisions [84].
	Propagation of Thermal Stress	Thermal coupling across the SoC can throttle CPUs responsible for sensing, middleware, and control orchestration even when the model workload itself is unchanged [85].
	Control-Loop Jitter and Modal Trade-offs	Thermal limits can introduce actuation jitter and force the system to stagger, reduce, or selectively disable sensing modalities to remain within sustained power and thermal budgets [45,85–87].

3.2. Heterogeneous Compute Mismatch

The **Heterogeneous Compute Mismatch** arises when embodied foundation model workloads are mapped onto edge platforms whose CPUs, GPUs, and NPUs are optimized for different classes of operators and execution regimes. Modern systems such as Jetson AGX Orin and Apple M-series devices rely on this heterogeneity to balance performance and energy efficiency under tight thermal limits [88]. Embodied FM workloads, however, combine dense tensor kernels, irregular preprocessing, host-side coordination, and latency-sensitive control in the same execution graph. The result is a recurring systems barrier: hardware heterogeneity improves nominal efficiency, but it undermines predictable low-latency execution. In practice, latency becomes sensitive to operator placement, runtime fragmentation, and scheduling effects rather than to peak compute alone [24].

3.2.1. Operator Coverage Gaps and Execution Bubbles

Foundation-model inference depends on a broad operator set, including attention, normalization, tensor reshaping, and irregular preprocessing, whereas edge NPUs accelerate a much narrower subset of dense kernels such as INT8 matrix multiplication. Unsupported operators force graph partitioning and CPU fallback, introducing host–device synchronization and interrupting otherwise continuous execution. These transitions create *execution bubbles* in which accelerators idle while control returns to the host. Sparse perception stages such as LiDAR voxelization are especially prone to these fallbacks and can introduce large latency spikes [75]. NanoVLA shows that removing even small CPU-resident operators can improve throughput by up to 1.7×, highlighting that fragmented execution, not peak compute alone, often determines end-to-end performance [45].

 doi:10.20944/preprints202603.1293.v1

3.2.2. Granularity Mismatch and Kernel Launch Overhead

Foundation-model inference also mismatches the execution granularity of mobile GPUs. Transformer decoding decomposes into many fine-grained kernels, but embedded GPUs cannot amortize launch overhead as effectively as larger server-class devices. On Jetson Orin, CPU-side launch overhead can exceed kernel execution time and account for 30–60% of end-to-end latency during autoregressive decoding [76]. Under these conditions, runtime overhead becomes a major component of inference cost. For VLA policies, sequential token generation can drive utilization below 20%, making multi-billion-parameter models difficult to reconcile with high-frequency control.

3.2.3. The Memory Wall under Unified-Memory Architectures

Unified-memory architectures amplify the mismatch further. On embedded SoCs, CPUs, GPUs, NPUs, and sensor DMA engines share the same LPDDR channels, so FM inference competes directly with high-rate perception traffic for bandwidth [77]. Diffusion-based policies intensify this pressure by revisiting large tensors across repeated denoising steps and sustaining DRAM traffic over long intervals. Cross-device tensor migration introduces an additional penalty. Moving high-resolution features between heterogeneous units, such as the CPU and NPU, serializes execution and can add 4–15 ms of latency, often offsetting the nominal benefit of hardware acceleration [9]. In this regime, end-to-end performance is constrained less by arithmetic throughput than by memory movement and bandwidth arbitration.

3.2.4. Thermal Constraints and Dark Silicon

Thermal limits turn these inefficiencies into sustained deployment failures. Multimodal embodied pipelines often activate CPUs, GPUs, and NPUs concurrently, pushing passive or compact cooling systems beyond their practical envelope. Dynamic voltage and frequency scaling (DVFS) responds by reducing clock rates, and mixed-workload saturation has been reported to reduce steady-state inference throughput by as much as 60% relative to cold-start performance [78]. Foundation-model workloads are especially exposed because they sustain high utilization over time rather than in short bursts. As a result, nominal peak performance becomes difficult to maintain during prolonged operation, and systems that initially satisfy real-time targets may later fall below them as thermal throttling accumulates [79].

3.2.5. Implications

Heterogeneous Compute Mismatch is therefore not just an efficiency loss; it is a stability problem for closed-loop embodied execution. Once inference spans heterogeneous units under shared thermal and memory limits, operator placement, launch overhead, and runtime fragmentation become first-order determinants of whether the system can sustain safe real-time behavior.

3.3. The Unified Memory Bottleneck

The **Unified Memory Bottleneck** arises when embodied workloads force inference, sensing, and runtime services to contend for the same LPDDR bandwidth and memory service time. Unlike server platforms with dedicated GPU VRAM, embedded SoCs place CPUs, GPUs, NPUs, and sensor peripherals on a shared memory fabric. Under these conditions, performance is constrained less by nominal capacity than by arbitration and data movement under concurrent load [80]. In embodied systems, that contention directly degrades latency stability and closed-loop responsiveness.

3.3.1. Bandwidth Saturation and Arbitration

Foundation-model inference places sustained pressure on memory bandwidth. Autoregressive decoding and diffusion-based policies repeatedly stream large parameter tensors from DRAM [81]. Data-center accelerators can mask much of this cost with bandwidth on the order of terabytes per second, whereas edge SoCs typically operate in the 25–204 GB/s range. In embodied deployment, that same bandwidth must also serve RGB cameras, LiDAR, event streams, and other high-rate inputs. As

 doi:10.20944/preprints202603.1293.v1

a result, accelerators often fail to reach nominal arithmetic throughput because execution is limited by memory supply rather than compute availability [76]. As arbitration pressure increases, latency variance rises and real-time margins shrink.

3.3.2. DMA Contention and Access Pattern Interference

DMA traffic adds a second source of interference. On platforms such as Jetson AGX Orin, image signal processors, USB controllers, and other peripheral interfaces compete with GPU and NPU kernels for the same memory cycles [65]. Because DMA traffic is bursty and often hardware-prioritized, inference can stall even when average bandwidth appears sufficient. Access-pattern mismatch further reduces effective throughput. Linear sensor streams interfere with the poor-locality access patterns of attention layers and sparse LiDAR pipelines, reducing cache efficiency and degrading bandwidth utilization [77]. Under unified memory, this interference disrupts both sensor buffering and model weight reuse.

3.3.3. Software Amplification of Memory Pressure

Software abstractions magnify the hardware bottleneck. Middleware frameworks such as ROS 2 often serialize, duplicate, or buffer high-bandwidth sensor messages across node boundaries. When large models already consume 6–8 GB for weights alone, these additional image and LiDAR buffers leave little headroom for stable execution. Small shifts in allocation demand can then trigger buffer reallocation, tensor rematerialization, page eviction, and other costly memory-management events. The result is latency jitter, throughput loss, and, in stressed regimes, node instability or watchdog resets [9]. The bottleneck therefore scales with model size not only because of hardware limits, but also because software layers amplify memory pressure.

3.3.4. Control-Loop Consequences

The effects of memory contention extend well beyond inference throughput. When DRAM service becomes unreliable, NPUs stall waiting for on-chip buffers to refill, and CPU threads responsible for synchronization and control queue behind memory-bound tasks. In robotic systems operating at 50–100 Hz, these delays appear as actuator jitter, delayed servo response, and degraded sensor fusion [70]. Sustained DRAM traffic also raises power draw and accelerates thermal saturation, triggering DVFS before compute units are fully utilized [78]. Empirical studies show that workloads such as diffusion policies can drop from roughly 30 FPS in isolation to 6–12 FPS under realistic sensor load because of memory contention alone. In embodied settings, this is not just a throughput penalty; it directly affects the freshness of observations presented to the control loop [20].

3.3.5. Implications

The Unified Memory Bottleneck is therefore a first-order systems barrier in embodied edge deployment. Once sensing, inference, and control share the same memory fabric, bandwidth arbitration, DMA behavior, and software buffering become direct determinants of whether the platform can sustain safe real-time operation.

 doi:10.20944/preprints202603.1293.v1

Table 2. Deployment Gauntlet barriers 5–8: recurring system-level challenges for embodied foundation models at the edge.

Barrier	Representative Subproblem	Systems Effect
Long-Horizon Execution Drift and Adaptation Failure	The Stationarity Fallacy	Embodied deployment violates the stationarity assumptions of offline training: sensor noise, actuator behavior, and calibration drift shift over time, causing prediction error to accumulate as observations move away from well-supported regions of the training distribution [89].
	Latent-State Misalignment	Physical drift can disrupt the alignment between sensory embeddings and action-relevant latent structure, causing internally coherent predictions to diverge from the true physical state of the platform [90].
	The Adaptation Gap	Edge platforms often cannot support timely online adaptation under SWaP constraints, while cloud-mediated adaptation introduces latency and connectivity dependence, leaving long-duration deployments governed by increasingly outdated model assumptions [91,92].
	Residual Hardware and Environment-Coupled Drift	Sensor bias instability, thermal noise, mechanical wear, and terrain-dependent slip introduce residual time-varying error that classical compensation can attenuate but not fully remove [93–96].
Safety & Verification Gap	Stochastic Objectives and Constraint Violations	Likelihood-based objectives such as token prediction and diffusion denoising do not inherently encode hard physical constraints, so actions may be statistically plausible while remaining dynamically unsafe under real-world perturbations [97].
	Opacity and Hierarchical Misalignment	High-dimensional latent representations are difficult to interpret formally, and mismatched interfaces between learned planners and constrained controllers can yield semantically valid but dynamically infeasible actions [35,38,98].
	Limits of Retrofitted Verification	Barrier certificates, reachability analysis, and fallback logic provide only partial coverage when state estimates are shifted, uncertainty is poorly calibrated, or policy dimensionality exceeds the scope of practical formal analysis.
	Operational Sandboxing and Structural Implications	Reduced velocities, geofencing, teleoperation, and runtime safety monitors lower operational risk, but they remain reactive and do not remove the underlying mismatch between generative model behavior and physically constrained control [99,100].
OS & Scheduling Bottleneck	Best-Effort Scheduling and Runtime Nondeterminism	Throughput-oriented schedulers and runtime-level serialization introduce jitter, descheduling, and callback delay that are difficult to reconcile with bounded-latency control execution [101,102].
	Jitter Propagation Across the Stack	Descheduling, priority inversion, and shared-resource contention propagate across sensing, inference, and actuation, producing bursty sensing, irregular frame timing, and cross-layer jitter cascades [102].
	Control-Theoretic Consequences	Best-effort timing violates assumptions such as bounded execution and regular sampling, introducing heavy-tailed delay that can manifest as oscillation, missed grasps, or delayed corrective action [103].
I/O & Communication Bottleneck	Bus Arbitration, Network Stochasticity, and Middleware Overhead	Shared DMA engines, LPDDR arbitration, network delay, and middleware serialization jointly disrupt bounded-delay multimodal transport, degrading throughput and timing regularity under load [104–106].
	Cross-Layer Timing Cascades	Delayed data movement can stall inference, misalign fusion, reorder messages, and propagate timing inconsistency across perception, planning, and control rather than remaining local to the transport layer.

3.4. The Energy and Thermal Ceiling

The **Energy and Thermal Ceiling** arises because mobile platforms must execute foundation models within fixed power, cooling, and endurance budgets. Unlike cloud infrastructure, mobile and aerial robots operate under strict Size, Weight, Power, and Cost (SWaP-C) constraints and have little thermal headroom. Under these conditions, sustained inference competes directly with propulsion, sensing, and actuation for both energy and heat dissipation capacity [82]. As a result, embodied performance is often bounded by steady-state power and thermal limits before nominal model capacity is reached.

3.4.1. Zero-Sum Power Competition

On embedded platforms, computation and physical actuation draw from the same onboard energy reservoir. In aerial systems, allocating an additional 10–15 W to accelerator-driven transformer inference can reduce flight endurance by several minutes [76]. The trade-off is particularly severe for foundation models because their cost is driven not only by arithmetic but also by DRAM access, which remains among the most energy-intensive operations in modern hardware [83]. Sustained memory traffic therefore raises both power draw and heat generation, forcing a direct trade-off between model throughput and operational range.

3.4.2. DVFS and Steady-State Throughput Loss

As the platform approaches its thermal envelope, Dynamic Voltage and Frequency Scaling (DVFS) lowers clock rates to maintain safe operation. The result is a drop in sustained throughput that can differ markedly from cold-start performance. A vision encoder that begins near 30 FPS, for example, may settle at a much lower rate after thermal throttling. Benchmark studies indicate that high-duty-

 doi:10.20944/preprints202603.1293.v1

cycle foundation-model workloads often enter this regime within minutes of continuous execution [84]. For embodied deployment, the relevant metric is therefore not startup throughput, but the control rate the platform can maintain after reaching thermal equilibrium.

3.4.3. Control Degradation under Thermal Stress

This steady-state slowdown propagates directly into the control stack. A policy that initially runs near 20 Hz may stabilize at a much lower frequency after throttling, increasing perception staleness and reducing closed-loop responsiveness. In embodied systems, timing margins are often narrow, so reduced inference rate leads directly to slower state updates and delayed control decisions. Thermal stress therefore alters not only throughput, but the temporal behavior of the system as a whole.

3.4.4. Propagation of Thermal Stress

Thermal effects are also coupled across the SoC. CPUs, GPUs, and NPUs typically share a die-level or package-level thermal envelope, so sustained GPU or NPU utilization can trigger CPU throttling even when the model workload itself is unchanged. This is consequential because the CPU continues to handle sensor drivers, middleware callbacks, and control orchestration. When CPU frequency drops, those services slow as well, increasing end-to-end latency outside the model proper [85].

3.4.5. Control-Loop Jitter and Modal Trade-offs

Once throttling spreads through the system, control-loop timing degrades. A nominal 100 Hz loop can develop substantial actuation jitter when CPU-bound planning and coordination stages stretch from single-digit milliseconds to tens of milliseconds [85]. Thermal limits also constrain multimodal execution. Running high-resolution vision, LiDAR fusion, and continuous audio concurrently can exceed the thermal design power of compact edge SoCs, forcing the system to reduce rates, stagger execution, or disable modalities selectively. In this setting, compact architectures such as NanoVLA [45], TinyVLA [86], and OneDP [87] are not just efficiency improvements; they are practical mechanisms for keeping multimodal inference within sustained thermal and power budgets.

3.4.6. Implications

The Energy and Thermal Ceiling is a sustained-execution barrier, not a short-burst benchmarking artifact. For embodied deployment, the relevant question is not whether a model can run briefly at peak speed, but whether the platform can maintain the required sensing, inference, and control rate over time without exhausting energy reserves or destabilizing timing.

3.5. Long-Horizon Execution Drift and Adaptation Failure

The **Long-Horizon Execution Drift and Adaptation Failure** barrier arises because embodied foundation models are deployed with mostly fixed representations in physical systems whose sensing, actuation, and environment change over time. Unlike cloud inference, embodied execution is inherently non-stationary: sensor characteristics shift, actuator behavior changes, and contact dynamics evolve across deployment horizons. When frozen models continue to assume a stable observation–action mapping under these conditions, prediction error accumulates and control quality degrades [107]. The core difficulty is not drift alone, but drift combined with limited capacity for online correction on edge platforms.

3.5.1. The Stationarity Fallacy

Embodied deployment violates the stationarity assumptions embedded in offline training. Sensor noise changes with temperature, actuator output degrades with battery discharge, and calibration shifts under vibration and wear. These effects induce persistent covariate shift in $P(O_t)$. Classical robotics pipelines address such variation through online state estimation and model-based correction, but frozen foundation models typically lack an equally direct mechanism for updating their internal representation of routine physical change. As drift accumulates, observations move away from well-

 doi:10.20944/preprints202603.1293.v1

supported regions of the training distribution, and prediction error compounds over time [89]. This yields a gradual degradation mode that is often underrepresented in short-horizon evaluation.

3.5.2. Latent-State Misalignment

Long-horizon drift is especially problematic for policies that rely on latent state integration, including Vision-Language-Action (VLA) and diffusion-based controllers. These systems assume that sensory embeddings remain aligned with the action-relevant structure learned during training. Physical perturbations can disrupt that alignment, mapping observations into latent regions that are less informative for control. The model may then continue to produce internally coherent predictions while diverging from the true physical state of the platform [90]. Recovery becomes difficult because the same drift that corrupted the latent state also weakens the corrective value of subsequent observations.

3.5.3. The Adaptation Gap

The barrier is compounded by an **adaptation gap** at the edge. Online gradient-based adaptation is often incompatible with the compute, timing, and energy budgets of SWaP-constrained platforms, while cloud-mediated adaptation introduces latency and connectivity dependence. As a result, long-duration deployments may be governed by increasingly outdated model assumptions. Test-time adaptation methods are promising [91,92], but current approaches often remain too costly or too slow for real-time edge execution. In practice, model error and physical drift can therefore reinforce one another over long horizons.

3.5.4. Residual Hardware and Environment-Coupled Drift

Not all long-horizon drift can be removed through classical compensation. Sensor and actuator behavior contain residual, time-varying effects that violate idealized estimator assumptions and remain only partially observable during deployment. At the sensor level, IMUs exhibit bias instability under temperature and vibration, establishing a persistent noise floor [93]. Vision and LiDAR sensors show related non-stationarity through thermally induced noise, gain variation, and dark-current shifts [94]. At the actuation level, mechanical wear changes friction and compliance, while terrain-dependent slip breaks rigid-body assumptions and accumulates odometric error [95,96]. Classical filters can attenuate part of this drift, but residual error remains and continues to perturb the observation action mapping presented to the model.

3.5.5. Implications

Long-horizon execution drift is therefore not only a perception or calibration issue. It is a systems-level barrier that emerges when fixed model representations are coupled to evolving physical platforms without a practical mechanism for continual correction. Over long missions, robust deployment depends not just on initial model accuracy, but on whether the system can preserve alignment among sensing, latent state, and control as conditions change.

3.6. The Safety and Verification Gap

The **Safety and Verification Gap** arises because foundation-model objectives, representations, and inference behavior do not provide the same safety assurances as explicitly constrained control pipelines. Classical control stacks are often built around known dynamics, feasibility constraints, and analyzable stability conditions; embodied foundation models instead inherit the stochasticity and opacity of large neural architectures. In these systems, strong predictive or generative performance does not imply constraint satisfaction or safe closed-loop behavior. The gap therefore reflects a structural mismatch between how these models are trained and what physical autonomy requires.

3.6.1. Stochastic Objectives and Constraint Violations

Foundation models are typically optimized for predictive likelihood, denoising quality, or sequence modeling objectives rather than explicit physical risk. Vision-Language-Action token prediction,

 doi:10.20944/preprints202603.1293.v1

diffusion denoising, and latent interpolation do not inherently encode hard constraints such as joint limits, contact stability, or collision margins. As a result, generated actions may be statistically plausible while still being dynamically unsafe. Modest perturbations, including sensor noise, lighting variation, or adversarial artifacts, can induce action errors and constraint violations in embodied settings [97]. The core limitation is straightforward: likelihood-based objectives do not, by themselves, enforce physical invariants at inference time.

3.6.2. Opacity and Hierarchical Misalignment

The verification problem is compounded by model opacity. Classical controllers expose interpretable state variables, local sensitivities, and feasibility structure; foundation models instead operate through high-dimensional latent representations whose relation to physical state is often indirect [38]. As a result, failure analysis depends heavily on empirical testing rather than formal inspection [98]. Opacity also complicates hierarchical integration. High-level semantic planners may lack explicit contracts with lower-level kinematic or feedback controllers, so a planner can produce linguistically valid but dynamically infeasible actions that downstream systems must reject or repair [35]. The barrier is therefore not only model uncertainty, but also interface mismatch between learned planning and constrained control.

3.6.3. Limits of Retrofitted Verification

Retrofitting formal guarantees onto foundation-model policies remains difficult. Control Barrier Functions can enforce local safety conditions, but those guarantees are only as reliable as the state estimate and model assumptions on which they depend. Under hallucinated or shifted observations, those assumptions may no longer hold. Formal reachability analysis faces a different limitation: it does not scale easily to high-dimensional, stochastic, multimodal transformer-based policies. Weak uncertainty calibration further narrows the available safety margin. When a model cannot reliably express epistemic uncertainty, downstream systems have limited ability to reject low-confidence actions or trigger fallback behavior before error accumulates. In practice, retrofitted verification remains useful, but it covers only part of the safety problem and degrades under the same distribution shifts that challenge the policy itself.

3.6.4. Operational Sandboxing and Structural Implications

Current embodied deployments often manage this gap through operational containment: reduced velocities, geofencing, supervised execution, and human teleoperation oversight. Runtime safety monitors that predict and interrupt imminent violations provide an important additional layer of protection [99]. However, these mechanisms remain largely reactive and depend on the same sensing and representation stack as the policy they supervise. They reduce operational risk, but they do not remove the underlying mismatch between generative model behavior and physically constrained control. Progress will require tighter integration of physical invariants, constraint satisfaction, and uncertainty estimation into the learning and control architecture itself [100].

3.6.5. Implications

The Safety and Verification Gap is therefore a systems-level barrier, not merely a control-theoretic one. In embodied deployment, the central question is not whether a model can generate a plausible action, but whether the full stack can establish that the action is feasible, safe, and reliable under uncertainty before it reaches the plant.

3.7. The OS and Scheduling Bottleneck

The **OS and Scheduling Bottleneck** arises because embodied foundation-model pipelines often execute on software stacks optimized for flexibility and throughput rather than bounded latency. Classical robotic control systems commonly rely on RTOS or bare-metal execution to enforce explicit timing guarantees, whereas FM-based embodied stacks are often built on Linux kernels, containerized

 doi:10.20944/preprints202603.1293.v1

environments, and Python-centered orchestration. These layers simplify integration and support heterogeneous workloads, but they also introduce timing variability across kernel scheduling, middleware execution, and user-space coordination. In closed-loop embodied systems, that variability weakens the timing guarantees on which safe control depends.

3.7.1. Best-Effort Scheduling and Runtime Nondeterminism

A primary source of instability is Linux's Completely Fair Scheduler (CFS), which is designed to maximize aggregate throughput rather than worst-case latency. Under this policy, safety-critical control loops compete with logging, networking, thermal management, and other auxiliary processes for CPU time. At control rates of roughly 100–500 Hz, even modest descheduling becomes consequential. On Jetson-class platforms, empirical studies report jitter spikes of 5–20 ms caused by kernel wake-ups, interrupt bursts, and I/O-driven preemption [101]. Once scheduling delay approaches or exceeds the control period, the problem is no longer throughput; it is a direct loss of timing integrity in the control loop.

Python-based orchestration compounds the same issue. The Global Interpreter Lock serializes concurrent execution, and garbage collection can introduce pauses that are both unbounded and opaque to the scheduler. As a result, common coordination mechanisms in perception and control stacks, including asynchronous coroutines and ROS 2 executors, are difficult to reconcile with predictable real-time execution. Callback delays of only 5–10 ms can disrupt timestamp alignment, propagate stale observations into Vision-Language-Action policies, and increase execution drift and collision risk [102]. These effects are not isolated anomalies; they follow from the interaction between best-effort scheduling and runtime-level serialization.

3.7.2. Jitter Propagation Across the Stack

Multimodal workloads amplify this problem through priority inversion and shared-resource contention. Background LiDAR decoding threads may preempt safety monitors, asynchronous data loaders may delay control execution, and ROS 2 executors may defer perception updates while servicing unrelated timers [102]. The immediate effect is bursty sensing, irregular frame timing, and transient sensor dropouts.

The more serious issue is that latency does not remain local to the component in which it originates. CPU descheduling can delay GPU kernel dispatch, and DMA activity from one sensor can block service to another. A single LiDAR burst may therefore delay camera buffering and trigger a *jitter cascade* across perception, planning, and actuation. Under these conditions, learned policies and predictive filters must operate on temporally inconsistent inputs rather than merely noisy ones.

3.7.3. Control-Theoretic Consequences

OS-level nondeterminism undermines several assumptions used in classical control analysis. Stability arguments typically depend on bounded execution times, Zero-Order Hold discretization, and noise models that do not exhibit heavy-tailed latency behavior [103]. Best-effort scheduling violates those assumptions by introducing irregular sampling and actuation delay. In practice, the resulting failures may appear as oscillations, missed grasps, or delayed corrective action and are easily misattributed to perception or policy quality, even when the underlying cause is temporal interference in the execution stack.

3.7.4. Implications

The OS and Scheduling Bottleneck is therefore a systems barrier, not merely an implementation detail. Gains in model efficiency or accelerator utilization do not by themselves restore bounded-latency execution on best-effort software stacks. Embodied FM systems that require dependable physical interaction need real-time-aware runtimes, priority-sensitive scheduling, and explicit timing contracts across sensing, inference, and control.

 doi:10.20944/preprints202603.1293.v1

3.8. The I/O and Communication Bottleneck

The **I/O and Communication Bottleneck** arises because embodied foundation-model pipelines depend on multimodal data arriving with bounded delay, consistent ordering, and usable cross-modal synchronization, yet the underlying I/O stack cannot reliably preserve all three under load. Closed-loop perception and control often operate on sub-10 ms timing budgets, but those budgets are eroded by arbitration in the I/O hierarchy, network variability, and middleware overhead. As a result, failures often originate not in model inference alone, but in the data path that feeds and coordinates it.

3.8.1. Bus Arbitration, Network Stochasticity, and Middleware Overhead

The first source of instability appears in the on-chip interconnect and memory subsystem. High-resolution RGB, high-rate IMU, and dense LiDAR streams compete for shared DMA engines and LPDDR bandwidth, which must also support foundation-model inference. This creates direct coupling between sensing and inference: when I/O pressure rises, arbitration delays data movement for both. On Orin-class platforms, such interference has been reported to reduce Vision-Language-Action throughput by 15–30% under background I/O load, primarily because of arbitration stalls rather than compute saturation [104].

In cloud-assisted and split-compute deployments, the same problem extends beyond the device. Packet loss, retransmissions, fading, and queueing introduce heavy-tailed delay, and round-trip latencies of 30–80 ms on Wi-Fi 6 can exceed the stability margins of high-frequency control. Related communication limits also affect federated and fleet learning, where aggressive compression and update throttling are often required to keep communication overhead manageable [105]. The main issue is therefore not bandwidth alone, but the inability to maintain regular timing across sensing, inference, and remote coordination.

Middleware adds a further source of timing distortion. ROS 2 and DDS introduce serialization, copying, marshalling, and discovery overheads that consume CPU time and memory bandwidth for large tensors [106]. More importantly, DDS QoS policies reshape delivery timing through buffering, queueing, and retransmission, which can disrupt cross-modal simultaneity. Without zero-copy transport and explicit end-to-end timing contracts, middleware can invalidate the temporal assumptions required for reliable fusion and state estimation.

3.8.2. Cross-Layer Timing Cascades

These timing disturbances do not remain local to the transport layer. Memory contention can delay DMA, delayed DMA can stall inference, stalled inference can misalign fusion, and middleware buffering can reorder or defer messages. Thermal throttling may further slow packet processing and callback execution. The result is degraded temporal consistency across perception, planning, and control.

In embodied systems, that inconsistency has direct operational consequences. Obstacle updates become stale, fused state estimates lose coherence, and control actions are issued against an environment that has already changed. These failures are not purely algorithmic; they emerge from the interaction between model execution and the I/O path that supplies its inputs.

3.8.3. Implications

The I/O and Communication Bottleneck is therefore a systems barrier rooted in data movement, ordering, and timing, not just in raw compute latency. Mitigating it requires data-centric redesign, including prioritization of safety-critical traffic, near-sensor preprocessing and compression, and zero-copy communication paths for large tensor flows [108]. Without such changes, improvements in model efficiency alone will not eliminate timing failures in embodied foundation-model deployment.

 doi:10.20944/preprints202603.1293.v1

4. Mitigation Families for the Deployment Gauntlet

The literature does not respond to the Deployment Gauntlet through eight isolated solution streams. The same mitigation often relieves multiple barriers at once by reducing synchronization cost, lowering memory traffic, or improving runtime predictability. For that reason, the solution space is better understood as a small set of recurring mitigation families rather than as a one-to-one mapping from barrier to fix. We group the literature into four such families: transport-aware sensing and synchronization relaxation, hardware-aligned model and compiler co-design, memory-traffic reduction and schedulable memory use, and energy- and thermal-aware execution. Figure 4 summarizes the Mitigation strategies for deployment gauntlet.

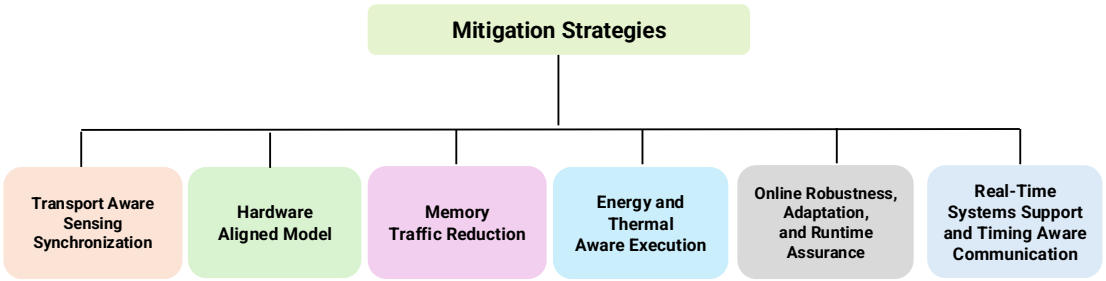

Figure 4. The taxonomy for Mitigation Strategies for Deployment Gauntlet on edge

4.1. Transport-Aware Sensing and Synchronization Relaxation

The first family acts on the data path before model inference begins. Its goal is to reduce transport overhead, relax unnecessary synchronization, and preserve temporal structure without forcing the full stack to wait on the slowest modality. Early work pursued this through zero-copy middleware transport. Iceoryx introduced shared-memory communication for fixed-size messages, while ROS 2 formalized loaned messages for middleware-managed buffer reuse [109,109,110]. Profiling studies show that default ROS 2 configurations can incur substantial latency relative to optimized shared-memory pipelines, with behavior strongly shaped by DDS and executor design [23]. Executor-level refinements were therefore introduced to reduce jitter and improve predictability under real-time load [70,102,111].

Recent work extends this idea beyond fixed-size message transport. Agnocast generalizes zero-copy communication to variable-sized autonomy messages while preserving ROS 2 compatibility [109]. NITROS maintains accelerator-friendly formats through type adaptation and reduces unnecessary copies for high-resolution sensor streams [112]. At the fusion level, MMEdge reduces idle waiting through pipelined sensing and cross-modal speculative skipping [64], while AsyncVLA relaxes rigid action timing by generating control tokens asynchronously [113]. Calibration maintenance belongs in the same family: CalibRefine lowers the downstream cost of spatial misalignment through targetless online refinement [69].

A related line of work reduces the need for strict synchronization altogether. Lightweight fusion modules, cooperative edge-oriented VLM pipelines, and hardware-aware multimodal co-design all aim to reduce how much data must be aligned exactly at runtime [114,115]. Uncertainty-aware fusion methods dynamically gate modality contributions instead of assuming equal reliability and perfect simultaneity [116–118]. Condition-aware pipelines similarly adapt to environmental regime rather than enforcing expensive redundancy [119]. Training can also absorb some of the burden. TimeAlign, UniBEV, and sensor-dropout strategies explicitly expose models to temporal misalignment or partial sensing, trading modest training complexity for substantially greater runtime slack [120–123]. Taken together, these methods treat sensing and synchronization as a transport-and-timing problem, not only as a perception problem.

 doi:10.20944/preprints202603.1293.v1

4.2. Hardware-Aligned Model and Compiler Co-Design

The second family addresses the mismatch between transformer-style workloads and heterogeneous edge accelerators. The central idea is to expose hardware constraints at compile time or redesign operators and models so that execution remains on the accelerator as much as possible [124]. Compiler-driven approaches are the most direct expression of this idea. MATCH extends TVM-style compilation with explicit SoC hardware models and BYOC integration to optimize placement and scheduling across heterogeneous engines [125–127]. TinyIREE similarly unifies compilation and deployment for embedded targets and reduces execution fragmentation across devices [128,129]. The shared lesson is that heterogeneity must be visible at compile time if host fallback and cross-engine synchronization are to be minimized.

When native hardware support is missing, operator reformulation becomes the next lever. T-MAN replaces unsupported low-bit primitives on NPUs with lookup-table abstractions that preserve end-to-end accelerator execution [130,131]. Quantization methods such as LLM.int8, GPTQ, and AWQ reduce tensor movement while preserving fidelity, improving accelerator residency and reducing host-device traffic [132–134]. Memory-aware attention redesigns such as FlashAttention reduce high-bandwidth memory access through IO-aware tiling [135–137], while FlightLLM targets host-device synchronization and kernel-launch overhead during autoregressive decoding [76,138].

The same design logic now appears at the architectural level. NanoVLA and TinyVLA redesign VLA pipelines for edge hardware by reducing redundant computation, sustaining accelerator residency, and separating heavyweight semantic reasoning from lightweight control [45,86,139]. SARA-RT replaces quadratic attention with linear-complexity alternatives to recover real-time feasibility in transformer-based robotic control [140]. Across these approaches, robust edge deployment emerges less from post-training cleanup than from joint redesign of compiler, operator, and model.

4.3. Memory-Traffic Reduction and Schedulable Memory Use

The third family targets shared memory directly. On edge SoCs, the main problem is often not arithmetic throughput but whether LPDDR service remains predictable when foundation models, sensing, and control all compete for the same memory fabric. One response is isolation and regulation. Early work showed that shared-cache and DRAM interference can inflate tail latency even when average throughput remains acceptable [65,141]. MemGuard-style reservation and policing mechanisms allocate DRAM budgets and throttle interfering workloads [142,143]. Later work extends these ideas to heterogeneous CPU–GPU systems through reserved bandwidth for real-time GPU tasks, accelerator-aware isolation, per-bank regulation, LLC control, and microsecond-scale policing [144–148]. In this line of work, memory is treated as a schedulable shared resource rather than a passive substrate.

A complementary strategy reduces traffic volume. Low-bit quantization lowers bandwidth demand while preserving accuracy; LifeQuant and ProxQ focus specifically on maintaining quantized performance in streaming or non-stationary settings [149–151]. Structured pruning reduces both parameter count and activation traffic through hardware-aware channel and block removal [152–154]. Rotation-based methods suppress activation outliers that would otherwise require precision escalation, and QuaRot extends low-bit execution across weights, activations, and KV caches [155–158].

The KV cache has become a particularly important target because it dominates long-context memory footprint. KIVI and related methods compress the cache through ultra-low-bit or coupled quantization [159,160], while adaptive-precision schemes vary cache precision across layers or tokens to allocate bits where they are most useful [161,162]. XQuant instead stores quantized layer inputs and rematerializes keys and values on demand, trading some on-chip computation for lower DRAM pressure [163]. Across these approaches, the picture is consistent: once sensing and inference share unified memory, real-time viability depends on both lowering traffic volume and making the remaining traffic schedulable.

 doi:10.20944/preprints202603.1293.v1

4.4. Energy and Thermal-Aware Execution

The fourth family addresses the fact that embodied deployment is governed by steady-state energy and thermal limits rather than by short-burst benchmark throughput. MAVBench makes this explicit by showing that autonomy workloads directly reduce vehicle endurance in micro aerial systems [164]. At a lower level, Horowitz and later efficient-accelerator studies show that memory access often dominates arithmetic energy, which helps explain why memory-heavy foundation models generate disproportionate thermal load even when compute is optimized [83,165,166]. The relevant design target is therefore not peak throughput, but *joules per decision* and sustainable duty cycle under continuous execution [167,168].

One response lowers the cost of each inference step. Distillation and compact architectures such as MobileBERT reduce deployment cost while preserving useful capacity [169]. Post-training quantization methods including ZeroQuant and SmoothQuant further reduce compute and memory-transfer energy by lowering precision without large accuracy loss [170,171]. These methods do not remove the thermal ceiling, but they shift the steady-state operating point.

For generative control, runtime duration matters as much as instantaneous cost. Diffusion-based policies are particularly demanding because repeated denoising accumulates heat over many steps [46]. OneDP reduces that burden by collapsing multi-step diffusion into a single-step generator [48], while LightDP prunes and restructures diffusion modules for sustained on-device execution [49]. In this regime, fewer generation steps directly improve thermal sustainability.

Thermal-aware runtime management provides the final lever. POD TAS uses predictive thermal-aware scheduling based on reduced-order thermal models to reduce peak temperature and temperature variance [172,173]. The broader lesson across this family is that thermal feasibility is a runtime property: compression lowers cost per operation, generative redesign reduces sustained duty cycle, and predictive scheduling shapes temperature growth over time. Under continuous edge deployment, sustainable embodied performance is determined by thermal equilibrium rather than by peak capability.

4.5. Online Robustness, Adaptation, and Runtime Assurance

Mitigations for long-horizon drift and safety failures share a common constraint: they must improve robustness without exceeding edge budgets for compute, timing, and energy. As a result, the literature does not pursue unrestricted online relearning of the full model. It instead relies on bounded corrective mechanisms that contain model mismatch before it propagates into unsafe closed-loop behavior. This family includes estimator-level drift compensation, parameter-efficient or gradient-free adaptation, control-theoretic safety filters, runtime assurance architectures, and reactive monitoring.

One strategy localizes correction outside the full perception or control backbone. In visual-inertial navigation, external neural modules estimate bias dynamics and inject corrections into factor-graph or invariant filtering pipelines [174,175]. In soft robotics, where sensing drift is intrinsic to the hardware, continual learning with rehearsal [176] and RNN-based observers [177] track nonlinear drift while keeping compensation decoupled from task-level policy representations. These methods are effective precisely because they restrict adaptation to a narrow interface rather than perturbing the full model.

A second strategy constrains adaptation itself. Test-time adaptation methods suppress low-information gradients and stabilize streaming updates [178–180], but on edge hardware such updates remain subordinate to latency-critical inference. Parameter-efficient methods narrow the update space further through sparse backpropagation [181], lightweight bias modules [182], and bias-only transformer adaptation with well under 0.1% of parameters [183]. When backpropagation is too costly or poorly supported by the hardware stack, gradient-free methods estimate updates from parameter perturbations [184], while meta-learning shifts most of the adaptation burden offline so that deployment requires only low-dimensional coefficient updates, as in Neural Fly [185]. These approaches make adaptation feasible, but only within a deliberately limited correction range.

The same logic appears on the safety side. Control Barrier Function wrappers and related quadratic-program filters impose forward-invariant constraints around learned actions [186], while

 doi:10.20944/preprints202603.1293.v1

Predictive Safety Filters and MPC-based supervisory schemes accept or replace actions according to an explicit feasibility model [187,188]. Reachability-based intervention and RL-CBF variants similarly restrict unsafe exploration through safe sets or uncertainty-aware safety envelopes [189,190], and SafeDiffuser extends this principle to diffusion-based planners by enforcing constraint-aware denoising [191]. These methods improve safety by constraining or correcting learned behavior externally rather than by making the generative model intrinsically safe.

Runtime assurance architectures generalize that externalization. Simplex-style designs, SOTER, and related supervisory frameworks treat the learned policy as an advanced but uncertified controller that may be overridden by a verified baseline when safety is threatened [192,193]. Formal verification methods such as Reluplex, Marabou, and neural reachability analysis provide stronger guarantees for small or structured networks [194–196], but they do not scale naturally to high-dimensional multimodal foundation models under deployment shift. Probabilistic safety certificates, conformal wrappers, and runtime shielding partially address that limitation by combining risk-bounded guarantees, runtime enforcement, and black-box monitoring [197–201]. In practice, these methods reduce operational risk, but they still depend on bounded correction, fallback, and supervision.

The central lesson is consistent across this family. Long-horizon drift and safety failures are mitigated through structured correction, constrained adaptation, and runtime oversight rather than through open-ended online relearning. These mechanisms improve robustness under realistic edge constraints, but they also make clear that dependable embodied deployment still depends on keeping correction bounded, explicit, and computationally tractable.

4.6. Real-Time Systems Support and Timing-Aware Communication

Mitigations for the OS, scheduling, I/O, and communication barriers address a different systems problem: preserving timing integrity across the execution stack. Their shared objective is to bound interference, reduce unnecessary data movement, and prevent timing disturbances from cascading across perception, planning, and control. Unlike model-centric optimizations, these methods operate on middleware, runtime, kernel, and transport layers.

At the middleware layer, chain-aware execution and callback isolation are central. Casini et al. [202] show that default ROS 2 executors produce pessimistic response-time bounds because of priority inversion and callback interference. Later work mitigates this by separating safety-critical callbacks from background activity [98] and by scheduling end-to-end execution chains rather than isolated callbacks. PiCAS is a representative example: it uses priority-driven, chain-aware scheduling to reduce perception-to-control latency substantially on embedded platforms [203]. These methods improve timing predictability, but they remain dependent on lower-level scheduling and resource isolation.

Kernel-level real-time support addresses a second part of the same problem. PREEMPT_RT substantially reduces CPU wake-up latency and improves interrupt handling under load [204], which is valuable for classical control and for the CPU-resident orchestration surrounding learned policies. Heterogeneous embodied systems, however, also depend on GPU kernels, DMA engines, and shared memory fabrics that the CPU scheduler governs only indirectly. OS-level real-time support therefore improves timing behavior but does not by itself guarantee end-to-end stability.

Runtime restructuring provides a more aggressive response. Systems such as Dora remove critical execution from Python and place it in compiled dataflow runtimes with zero-copy shared memory and explicit execution graphs, reducing latency relative to conventional ROS 2 Python pipelines [205]. The broader lesson is that runtime architecture matters: dataflow-style execution, reduced interpreter overhead, and explicit dependency graphs are often more compatible with predictable timing than general-purpose task orchestration.

Transport and communication optimizations tackle the same issue from the data path. On unified-memory platforms, bandwidth regulation mechanisms such as MemGuard, BWLOCK, and later dynamic or DSU-level policies protect safety-critical traffic by throttling interferers and reserving DRAM service under contention [142–144,206]. Zero-copy shared-memory transport such as Iceoryx

 doi:10.20944/preprints202603.1293.v1

reduces serialization and large-payload copying overhead [207], while Hazcat and FPGA-assisted DDS reduce host-side transport cost by unifying memory handling or offloading communication across heterogeneous devices [208,209]. These methods do not remove timing variability, but they reduce how much of it is injected into the critical path.

When execution spans cloud and edge, communication-aware partitioning becomes necessary. Dynamic split-compute frameworks such as Neurosurgeon and SPINN adapt the partition point to current channel conditions [210,211]. BottleNet, Elf, and related selective-transmission methods further reduce bandwidth demand through representation compression or sparse token transfer [212–214]. These approaches improve survivability under communication variability, but they do so by trading model fidelity, freshness, or cross-modal coherence for lower transport load.

Across this family, the conclusion is straightforward: timing reliability in embodied FM deployment is not recovered by faster models alone. It depends on coordinated support across middleware scheduling, kernel preemption, runtime design, bandwidth isolation, and communication-aware execution. The objective is not perfect determinism under all conditions, but an execution stack in which timing disruptions are bounded, localized, and less likely to cascade into unsafe control behavior. Table 3 summarizes the mitigation families that address the long-horizon robustness, safety-assurance, timing, and communication barriers in embodied foundation model deployment.

 doi:10.20944/preprints202603.1293.v1

Table 3. Mitigation families for the Deployment Gauntlet: representative strategies for improving embodied foundation model deployment at the edge.

Mitigation Family	Core Strategy	Deployment Benefit	Representative Techniques / Evidence
Transport-Aware Sensing and Synchronization Relaxation	Reduce transport overhead, relax rigid synchronization, and preserve usable temporal structure before inference begins.	Lower tail latency and jitter, reduce copy overhead, and improve multimodal alignment without forcing the stack to wait on the slowest modality.	Zero-copy middleware transport and loaned buffers [109,109,110], executor and synchronization refinements [23,70,102,111], variable-sized zero-copy and accelerator-native transport [109,112], pipelined and asynchronous sensing/action generation [64,113], calibration maintenance [69], and training-time robustness to misalignment or missing modalities [120–123].
Hardware-Aligned Model and Compiler Co-Design	Expose hardware constraints at compile time and redesign operators or models to reduce fallback, fragmentation, and unsupported execution paths.	Higher accelerator residency, fewer host fallbacks, lower synchronization overhead, and better real-time feasibility on heterogeneous edge hardware.	SoC-aware compilation and heterogeneous placement [124–129], operator reformulation for end-to-end accelerator execution [130,131], traffic-reducing quantization and IO-aware attention [132–137], host-device synchronization reduction [76,138], and hardware-aligned model redesign [45,86,139,140].
Memory-Traffic Reduction and Schedulable Memory Use	Treat shared memory as a schedulable resource and reduce traffic volume so sensing, inference, and control can coexist without excessive jitter.	More predictable LPDDR service, lower tail latency, fewer stalls, and improved sustained performance under unified-memory contention.	DRAM isolation and bandwidth policing [65,141–148], streaming-stable quantization and pruning [149–154], outlier suppression and uniform low-bit execution [155–158], and KV-cache redesign that trades compute for lower LPDDR pressure [159–163].
Energy- and Thermal-Aware Execution	Optimize for sustainable duty cycle and steady-state thermal feasibility rather than short-burst benchmark throughput.	Lower joules per decision, better endurance, reduced throttling, and more stable long-horizon real-time behavior.	Endurance-aware deployment evidence [164], energy-per-inference reduction through distillation and quantization [83,165,166,169–171], duty-cycle reduction for generative control [46,48,49], and predictive thermal-aware runtime management [167,168,172,173].
Online Robustness, Adaptation, and Runtime Assurance	Bound model mismatch through localized correction, constrained adaptation, external safety filters, and runtime supervision rather than unrestricted online relearning.	Improves long-horizon robustness and reduces unsafe behavior under realistic edge constraints while keeping correction computationally tractable.	Estimator-level drift compensation through neural bias prediction and lightweight observers [174–177], constrained test-time adaptation and streaming-stable updates [178–180], sparse and parameter-efficient adaptation [181–183], gradient-free and meta-learned adaptation [184,185], control-theoretic safety wrappers and predictive safety filters [186,187,189–191], runtime assurance architectures [188,192,193], formal verification and uncertainty-aware certificates [194–198], and runtime shielding and black-box monitoring [199–201].
Real-Time Systems Support and Timing-Aware Communication	Bound interference across middleware, runtime, kernel, memory, and transport layers so timing disturbances do not cascade across perception, planning, and control.	Improves timing predictability, reduces transport-induced jitter, and limits the spread of scheduling and communication variability through the execution stack.	Middleware callback isolation and chain-aware scheduling [98,202,203], kernel-level real-time support [204], compiled dataflow runtimes and zero-copy execution graphs [205], DRAM bandwidth regulation and arbitration control [142–144,206], zero-copy and heterogeneous I/O transport optimization [207–209], and communication-aware cloud–edge partitioning with compression or selective transmission [210–214].

5. Future Directions: Beyond the Deployment Gauntlet

The Deployment Gauntlet suggests that incremental gains in compression, quantization, or model scaling often improve *average* performance more than *worst-case* deployment behavior under SWaP-constrained, closed-loop execution. Many of the most consequential failures-including memory contention, missed deadlines, thermal excursions, and safety-envelope violations-arise from cloud-inference assumptions such as best-effort scheduling, batch-oriented memory traffic, and weak timing contracts. Figure 5 summarizes a set of representative architectural responses to these constraints.

Representative architectural directions:

- **Event-driven sensing:** Replace dense frame pipelines with event-structured inputs to reduce redundant compute and bandwidth, improve temporal alignment, and tighten end-to-end latency budgets [215,216].
- **Memory-centric and near-sensor compute:** Reorganize dataflow to reduce tensor movement through near-sensor preprocessing and PIM-style primitives, mitigating the unified-memory contention that often dominates edge latency and energy [217] [218].

 doi:10.20944/preprints202603.1293.v1

- **World-model-centric autonomy:** Use predictive state as a synchronization substrate under delay, occlusion, and sensor disagreement, enabling temporally grounded planning rather than purely reactive control [219,220].
- **Bifurcated cognition (System 1 / System 2):** Separate high-rate control authority from lower-rate semantic reasoning so that deliberative stalls do not propagate directly into actuation hazards [35, 38].
- **Real-time safety enforcement:** Combine lightweight monitoring with bounded-overhead runtime constraint enforcement, such as CBF-style reflex layers, that remain effective under missed deadlines and resource pressure [38,221].
- **Collaborative embodiment and fleet learning:** Treat robustness partly as a fleet-level property by amortizing rare events and distribution shift across deployments through federated, split, or distilled update mechanisms under limited connectivity [222].

5.1. Evaluation Trends: From Accuracy to Operational Validity

These architectural directions imply a corresponding shift in evaluation. Accuracy and average latency remain necessary, but they are weak predictors of real deployment outcomes when the execution stack is constrained by timing, memory, and thermal pressure. Future benchmarks should therefore report at least three additional properties: (i) **temporal coherence**, including state freshness and synchronization under load; (ii) **bounded failure and recoverability**, including graceful degradation within safety envelopes; and (iii) **resource stability**, including sensitivity to bandwidth pressure, I/O contention, thermal throttling, and scheduling jitter. Stress-test protocols that inject jitter, bus pressure, packet loss, and sensor dropout would make evaluation more operationally meaningful by measuring whether systems remain coherent, bounded, and safe under realistic constraints.

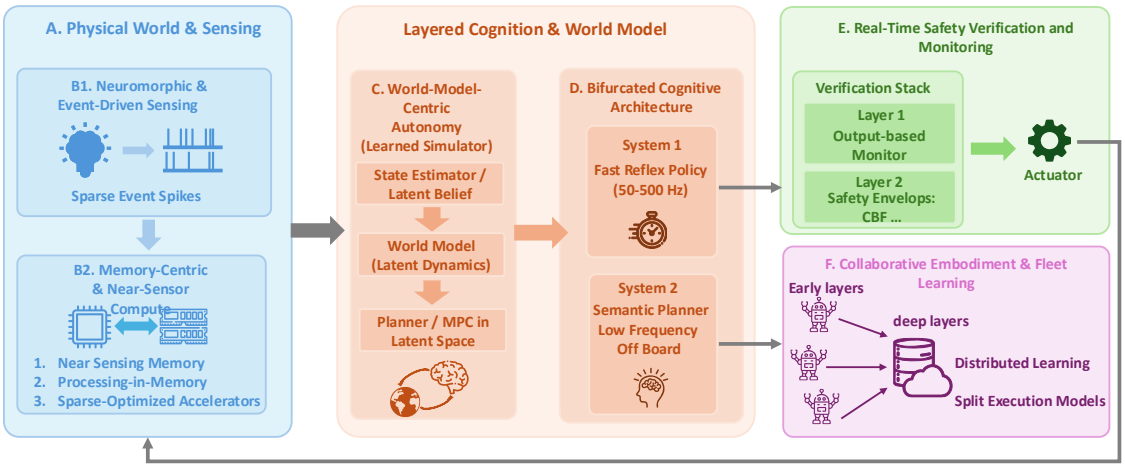

Figure 5. Architecture trends for future Embodied Foundation Models: Data volume is minimized via Neuromorphic and event sensing (A), and near sensor compute mechanism (B). A central predictive world model stabilizes perception (C) while cognition is separated into fast reflexes and slow reasoning (D). The rigorous safety policy(E) enforces constraints before actuation. And the Fleet learning (F) extends the feasible complexity.

6. Conclusion

Deploying foundation models on edge embodied platforms is a systems problem, not simply a problem of compression or accelerator efficiency. Reliable embodied performance depends on the interaction of sensing, computation, memory, timing, power, and safety under closed-loop execution. We organized these recurring constraints through the Deployment Gauntlet, a systems taxonomy that explains why gains in average-case throughput do not necessarily translate into dependable deployment. Across the literature, quantization, pruning, and related optimizations improve deploya-

 doi:10.20944/preprints202603.1293.v1

bility, but they do not, on current evidence, resolve the worst-case failures associated with memory contention, timing variability, thermal instability, and limited safety assurance.

A central implication of this survey is that embodied deployment increasingly favors architectural decomposition over monolithic end-to-end execution. Separating high-rate control from lower-rate semantic reasoning is one promising response to the tension between control-loop determinism and deliberative inference. More broadly, reducing data movement is often as important as reducing arithmetic cost. This makes transport-aware sensing, memory-centric execution, and near-sensor or in-memory processing especially relevant for future embodied systems operating under strict SWaP constraints.

The broader lesson is that robust embodied intelligence will depend less on transferring cloud-oriented model assumptions to the edge and more on explicit co-design across sensing, memory hierarchy, runtime scheduling, communication, and model architecture. The semantic capabilities of foundation models remain valuable, but realizing them in physical systems requires stacks designed around edge constraints rather than adapted to them only after training. In that sense, the future of embodied foundation models lies not simply in smaller models on smaller devices, but in systems whose learning objectives, execution substrate, and safety requirements are aligned from the outset.

References

1. Qazi, M.A.; Iosifidis, A.; Zhang, Q. PRISM: Distributed Inference for Foundation Models at Edge. *arXiv preprint arXiv:2507.12145* **2025**.
2. Bommasani, R.; Hudson, D.A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M.S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* **2021**.
3. Li, Y.; Wen, H.; Wang, W.; Li, X.; Yuan, Y.; Liu, G.; Liu, J.; Xu, W.; Wang, X.; Sun, Y.; et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459* **2024**.
4. Duan, J.; Yu, S.; Tan, H.L.; Zhu, H.; Tan, C. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence* **2022**, *6*, 230–244.
5. Reddi, V.J.; Cheng, C.; Kanter, D.; Mattson, P.; Schmuelling, G.; Wu, C.J.; Anderson, B.; Breughe, M.; Charlebois, M.; Chou, W.; et al. Mlperf inference benchmark. In Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020, pp. 446–459.
6. Moravec, H. *Mind children: The future of robot and human intelligence*; Harvard University Press, 1988.
7. Girdhar, R.; El-Nouby, A.; Liu, Z.; Singh, M.; Alwala, K.V.; Joulin, A.; Misra, I. Imagebind: One embedding space to bind them all. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 15180–15190.
8. Zitkovich, B.; Yu, T.; Xu, S.; Xu, P.; Xiao, T.; Xia, F.; Wu, J.; Wohlhart, P.; Welker, S.; Wahid, A.; et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Proceedings of the Conference on Robot Learning. PMLR, 2023, pp. 2165–2183.
9. Kim, M.J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246* **2024**.
10. Driess, D.; Xia, F.; Sajjadi, M.S.; Lynch, C.; Chowdhery, A.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; Huang, W.; et al. Palm-e: An embodied multimodal language model **2023**.
11. Choy, C.; Gwak, J.; Savarese, S. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 3075–3084.
12. Tang, H.; Liu, Z.; Zhao, S.; Lin, Y.; Lin, J.; Wang, H.; Han, S. Searching efficient 3d architectures with sparse point-voxel convolution. In Proceedings of the European conference on computer vision. Springer, 2020, pp. 685–702.
13. Radford, A.; Kim, J.W.; Xu, T.; Brockman, G.; McLeavey, C.; Sutskever, I. Robust speech recognition via large-scale weak supervision. In Proceedings of the International conference on machine learning. PMLR, 2023, pp. 28492–28518.

 doi:10.20944/preprints202603.1293.v1

14. Zhang, Z.; Geiger, J.; Pohjalainen, J.; Mousa, A.E.D.; Jin, W.; Schuller, B. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)* **2018**, *9*, 1–28.
15. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* **2020**.
16. Deiana, A.M.; Tran, N.; Agar, J.; Blott, M.; Di Guglielmo, G.; Duarte, J.; Harris, P.; Hauck, S.; Liu, M.; Neubauer, M.S.; et al. Applications and techniques for fast machine learning in science. *Frontiers in big Data* **2022**, *5*, 787421.
17. Reed, S.; Zolna, K.; Parisotto, E.; Colmenarejo, S.G.; Novikov, A.; Barth-Maron, G.; Gimenez, M.; Sulsky, Y.; Kay, J.; Springenberg, J.T.; et al. A generalist agent. *arXiv preprint arXiv:2205.06175* **2022**.
18. Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A.M.; Hauth, A.; Millican, K.; et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* **2023**.
19. Sze, V.; Chen, Y.H.; Yang, T.J.; Emer, J.S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE* **2017**, *105*, 2295–2329.
20. Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M.W.; Keutzer, K. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*; Chapman and Hall/CRC, 2022; pp. 291–326.
21. Kulkarni, U.; Hosamani, A.S.; Masur, A.S.; Hegde, S.; Vernekar, G.R.; Chandana, K.S. A survey on quantization methods for optimization of deep neural networks. In Proceedings of the 2022 international conference on automation, computing and renewable systems (ICACRS). IEEE, 2022, pp. 827–834.
22. Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W. Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics* **2022**, *7*, eabm6074.
23. Kronauer, T.; Pohlmann, J.; Matthé, M.; Smejkal, T.; Fettweis, G. Latency analysis of ros2 multi-node systems. In Proceedings of the 2021 IEEE international conference on multisensor fusion and integration for intelligent systems (MFI). IEEE, 2021, pp. 1–7.
24. Hooker, S. The hardware lottery. *Communications of the ACM* **2021**, *64*, 58–65.
25. Wang, Q.; Oswald, D. Confidential Computing on Heterogeneous CPU-GPU Systems: Survey and Future Directions. *arXiv preprint arXiv:2408.11601* **2024**.
26. Park, J.; Kwon, O.; Lee, Y.; Kim, S.; Byeon, G.; Yoon, J.; Nair, P.J.; Hong, S. A case for speculative address translation with rapid validation for gpus. In Proceedings of the 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024, pp. 278–292.
27. Cao, K.; Liu, Y.; Meng, G.; Sun, Q. An overview on edge computing research. *IEEE access* **2020**, *8*, 85714–85728.
28. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics* **2016**, *32*, 1309–1332.
29. Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* **2016**.
30. Zhang, X.; He, R.; Tong, K.; Man, S.; Tong, J.; Li, H.; Zhuang, H. Complex Motion Planning for Quadruped Robots Using Large Language Models. In Proceedings of the 2024 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2024, pp. 1–5.
31. Capodieci, N.; Cavicchioli, R.; Bertogna, M.; Paramakuru, A. Deadline-based scheduling for GPU with preemption support. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), 2018, pp. 119–130.
32. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE internet of things journal* **2016**, *3*, 637–646.
33. Niu, C.; Ding, Y.; Lu, J.; Huang, Z.; Zeng, H.; Dai, Y.; Tu, X.; Lv, C.; Wu, F.; Chen, G. Collaborative Learning of On-Device Small Model and Cloud-Based Large Model: Advances and Future Directions. *arXiv preprint arXiv:2504.15300* **2025**.
34. Team, G.R.; Abeyruwan, S.; Ainslie, J.; Alayrac, J.B.; Arenas, M.G.; Armstrong, T.; Balakrishna, A.; Baruch, R.; Bauza, M.; Blokzijl, M.; et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020* **2025**.
35. Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. Do as i can, not as i say: Grounding language in robotic affordances. 2022.
36. Zeng, W.; Lu, S.; Yin, K.; Niu, X.; Dai, M.; Wang, J.; Pang, J. Behavior foundation model for humanoid robots. *arXiv preprint arXiv:2509.13780* **2025**.

 doi:10.20944/preprints202603.1293.v1

37. Yu, J.; Fu, L.; Huang, H.; El-Refai, K.; Ambrus, R.A.; Cheng, R.; Irshad, M.Z.; Goldberg, K. Real2render2real: Scaling robot data without dynamics simulation or robot hardware. *arXiv preprint arXiv:2505.09601* **2025**.
38. Firoozi, R.; Tucker, J.; Tian, S.; Majumdar, A.; Sun, J.; Liu, W.; Zhu, Y.; Song, S.; Kapoor, A.; Hausman, K.; et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research* **2025**, *44*, 701–739.
39. Li, X.; He, X.; Zhang, L.; Wu, M.; Li, X.; Liu, Y. A comprehensive survey on world models for embodied ai. *arXiv preprint arXiv:2510.16732* **2025**.
40. Xu, P.; Zhu, X.; Clifton, D.A. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2023**, *45*, 12113–12132.
41. Williams, S.; Waterman, A.; Patterson, D. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM* **2009**, *52*, 65–76.
42. Prashanthi S, K.; Sahoo, K.K.; Ranjan Saikia, A.; Gupta, P.; Vinay Joshi, A.; Pansari, P.; Simmhan, Y. Pagoda: An Energy and Time Roofline Study for DNN Workloads on Edge Accelerators. *arXiv e-prints* **2025**, pp. arXiv–2509.
43. Goyal, A.; Hadfield, H.; Yang, X.; Blukis, V.; Ramos, F. Vla-0: Building state-of-the-art vlas with zero modification. *arXiv preprint arXiv:2510.13054* **2025**.
44. Sapkota, R.; Cao, Y.; Roumeliotis, K.I.; Karkee, M. Vision-Language-Action (VLA) Models: Concepts, Progress, Applications and Challenges. *arXiv preprint arXiv:2505.04769* **2025**.
45. Chen, J.; Wang, J.; Chen, L.; Cai, C.; Lu, J. NanoVLA: Routing Decoupled Vision-Language Understanding for Nano-sized Generalist Robotic Policies. *arXiv preprint arXiv:2510.25122* **2025**.
46. Chi, C.; Xu, Z.; Feng, S.; Cousineau, E.; Du, Y.; Burchfiel, B.; Tedrake, R.; Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research* **2025**, *44*, 1684–1704.
47. Team, O.M.; Ghosh, D.; Walke, H.; Pertsch, K.; Black, K.; Mees, O.; Dasari, S.; Hejna, J.; Kreiman, T.; Xu, C.; et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213* **2024**.
48. Wang, Z.; Li, Z.; Mandlekar, A.; Xu, Z.; Fan, J.; Narang, Y.; Fan, L.; Zhu, Y.; Balaji, Y.; Zhou, M.; et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257* **2024**.
49. Wu, Y.; Wang, H.; Chen, Z.; Pang, J.; Xu, D. On-device diffusion transformer policy for efficient robot manipulation. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, pp. 14073–14083.
50. Zhang, S.; Fang, Q.; Yang, Z.; Feng, Y. Llava-mini: Efficient image and video large multimodal models with one vision token. *arXiv preprint arXiv:2501.03895* **2025**.
51. Yao, Y.; Yu, T.; Zhang, A.; Wang, C.; Cui, J.; Zhu, H.; Cai, T.; Chen, C.; Li, H.; Zhao, W.; et al. Efficient GPT-4V level multimodal large language model for deployment on edge devices. *Nature Communications* **2025**, *16*, 5509.
52. Saha, S.; Xu, L. Vision transformers on the edge: A comprehensive survey of model compression and acceleration strategies. *Neurocomputing* **2025**, p. 130417.
53. Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A.C.; Lo, W.Y.; et al. Segment anything. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision, 2023, pp. 4015–4026.
54. Zhang, C.; Han, D.; Qiao, Y.; Kim, J.U.; Bae, S.H.; Lee, S.; Hong, C.S. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289* **2023**.
55. Zhou, C.; Li, X.; Loy, C.C.; Dai, B. EdgeSAM: Prompt-in-the-loop distillation for SAM. *International Journal of Computer Vision* **2025**, *133*, 8452–8468.
56. Lang, A.H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; Beijbom, O. Pointpillars: Fast encoders for object detection from point clouds. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 12697–12705.
57. Chandorkar, A.; Tercan, H.; Meisen, T. Rethinking Backbone Design for Lightweight 3D Object Detection in LiDAR. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, pp. 1698–1706.
58. Liu, Z.; Tang, H.; Amini, A.; Yang, X.; Mao, H.; Rus, D.; Han, S. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv preprint arXiv:2205.13542* **2022**.
59. Speckhard, D.T.; Misiunas, K.; Perel, S.; Zhu, T.; Carlile, S.; Slaney, M. Neural architecture search for energy efficient always-on audio models. *arXiv preprint arXiv:2202.05397* **2022**.
60. Gerganov, G. whisper. cpp: Port of OpenAI's Whisper model in C/C++, 2023.

 doi:10.20944/preprints202603.1293.v1

61. Nassereldine, A.; Liu, D.; Xu, C.; Qin, R.; Shi, Y.; Xiong, J. PI-Whisper: Designing an Adaptive and Incremental Automatic Speech Recognition System for Edge Devices, 2024.
62. Xu, M.; Jin, A.; Wang, S.; Su, M.; Ng, T.; Mason, H.; Han, S.; Lei, Z.; Deng, Y.; Huang, Z.; et al. Conformer-based speech recognition on extreme edge-computing devices. In Proceedings of the Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track), 2024, pp. 131–139.
63. Feng, D.; Haase-Schütz, C.; Rosenbaum, L.; Hertlein, H.; Glaeser, C.; Timm, F.; Wiesbeck, W.; Dietmayer, K. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems* **2020**, *22*, 1341–1360.
64. Huang, R.; Yu, M.; Tsoi, M.; Ouyang, X. MMEdge: Accelerating On-device Multimodal Inference via Pipelined Sensing and Encoding, 2025.
65. Bechtel, M.; Yun, H. Analysis and mitigation of shared resource contention on heterogeneous multicore: An industrial case study. *IEEE Transactions on Computers* **2024**, *73*, 1753–1766.
66. Li, R.; Guan, N.; Jiang, X.; Guo, Z.; Dong, Z.; Lv, M. Worst-case time disparity analysis of message synchronization in ROS. In Proceedings of the 2022 IEEE Real-Time Systems Symposium (RTSS). IEEE, 2022, pp. 40–52.
67. Jellum, E.R.; Bryne, T.H.; Johansen, T.A.; Orlandić, M. The syncline model-analyzing the impact of time synchronization in sensor fusion. In Proceedings of the 2022 IEEE Conference on Control Technology and Applications (CCTA). IEEE, 2022, pp. 1446–1453.
68. Lv, X.; Wang, B.; Dou, Z.; Ye, D.; Wang, S. LCCNet: LiDAR and camera self-calibration using cost volume network. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2894–2901.
69. Cheng, L.; Guo, L.; Zhang, T.; Bang, T.; Harris, A.; Hajij, M.; Sartipi, M.; Cao, S. CalibRefine: Deep Learning-Based Online Automatic Targetless LiDAR–Camera Calibration with Iterative and Attention-Driven Post-Refinement, 2025.
70. Teper, H.; Bell, O.; Günzel, M.; Gill, C.; Chen, J.J. Bridging the Gap between ROS˜ 2 and Classical Real-Time Scheduling for Periodic Tasks. *arXiv preprint arXiv:2408.03696* **2024**.
71. Gallego, G.; Delbrück, T.; Orchard, G.; Bartolozzi, C.; Taba, B.; Censi, A.; Leutenegger, S.; Davison, A.J.; Conradt, J.; Daniilidis, K.; et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence* **2020**, *44*, 154–180.
72. Gehrig, D.; Gehrig, M.; Hidalgo-Carrió, J.; Scaramuzza, D. Video to events: Recycling video datasets for event cameras. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3586–3595.
73. Tang, Y.; Liao, H.; Nie, T.; He, J.; Qu, A.; Chen, K.; Ma, W.; Li, Z.; Sun, L.; Xu, C. E3AD: An emotion-aware vision-language-action model for human-centric end-to-end autonomous driving. *arXiv preprint arXiv:2512.04733* **2025**.
74. Ruan, S.; Wang, R.; Shen, X.; Liu, H.; Xiao, B.; Shi, J.; Zhang, K.; Huang, Z.; Liu, Y.; Chen, E.; et al. A survey of multi-sensor fusion perception for embodied AI: Background, methods, challenges and prospects. *arXiv preprint arXiv:2506.19769* **2025**.
75. He, Z.; Fan, X.; Peng, Y.; Shen, Z.; Jiao, J.; Liu, M. Empointmovseg: Sparse tensor-based moving-object segmentation in 3-d lidar point clouds for autonomous driving-embedded system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2022**, *42*, 41–53.
76. Zeng, S.; Liu, J.; Dai, G.; Yang, X.; Fu, T.; Wang, H.; Ma, W.; Sun, H.; Li, S.; Huang, Z.; et al. Flightllm: Efficient large language model inference with a complete mapping flow on fpgas. In Proceedings of the Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2024, pp. 223–234.
77. Ivanov, A.; Dryden, N.; Ben-Nun, T.; Li, S.; Hoefler, T. Data movement is all you need: A case study on optimizing transformers. *Proceedings of Machine Learning and Systems* **2021**, *3*, 711–732.
78. Karumbunathan, L.S. Nvidia jetson agx orin series. *A Giant Leap Forward for Robotics and Edge AI Applications. Technical Brief* **2022**.
79. Esmaeilzadeh, H.; Blem, E.; Sankaralingam, R.S.A.K.; Burger, D. RETROSPECTIVE: Dark Silicon and the End of Multicore Scaling.
80. Akkad, G.; Mansour, A.; Inaty, E. Embedded deep learning accelerators: A survey on recent advances. *IEEE Transactions on Artificial Intelligence* **2023**, *5*, 1954–1972.
81. Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150* **2019**.

 doi:10.20944/preprints202603.1293.v1

82. Tan, X.; Wu, G.; Li, Z.; Liu, K.; Zhang, C. Autonomous emergency collision avoidance and collaborative stability control technologies for intelligent vehicles: a survey. *IEEE Transactions on Intelligent Vehicles* **2024**.
83. Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In Proceedings of the 2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC). IEEE, 2014, pp. 10–14.
84. Swaminathan, T.P.; Silver, C.; Akilan, T.; Kumar, J. Benchmarking deep learning models on nvidia jetson nano for real-time systems: An empirical investigation. *Procedia Computer Science* **2025**, *260*, 906–913.
85. van den Hoven, G.M. Introducing a performance observation framework to ros2. PhD thesis, Master's thesis, Mathematics and Computer Science, 2024.
86. Wen, J.; Zhu, Y.; Li, J.; Zhu, M.; Tang, Z.; Wu, K.; Xu, Z.; Liu, N.; Cheng, R.; Shen, C.; et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters* **2025**.
87. Luo, S.; Tan, Y.; Huang, L.; Li, J.; Zhao, H. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378* **2023**.
88. Gill, S.S.; Golec, M.; Hu, J.; Xu, M.; Du, J.; Wu, H.; Walia, G.K.; Murugesan, S.S.; Ali, B.; Kumar, M.; et al. Edge AI: A taxonomy, systematic review and future directions. *Cluster Computing* **2025**, *28*, 18.
89. Kumar, A.; Fu, Z.; Pathak, D.; Malik, J. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034* **2021**.
90. Ha, D.; Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122* **2018**, *2*.
91. Ren, P.; Zhang, K.; Zheng, H.; Li, Z.; Wen, Y.; Zhu, F.; Ma, S.; Liang, X. Surfer: A world model-based framework for vision-language robot manipulation. *IEEE Transactions on Neural Networks and Learning Systems* **2025**.
92. Gao, Y.; Zhang, Y.; Cai, Z.; Huang, D. Test-Time Adaptive Object Detection with Foundation Model. *arXiv preprint arXiv:2510.25175* **2025**.
93. Zhou, S.; Katragadda, S.; Huang, G. Learning IMU Bias with Diffusion Model. *arXiv preprint arXiv:2505.11763* **2025**.
94. Vargas, J.; Alsweiss, S.; Toker, O.; Razdan, R.; Santos, J. An overview of autonomous vehicles sensors and their vulnerability to weather conditions. *Sensors* **2021**, *21*, 5397.
95. Sigron, P.; Aschwanden, I.; Bambach, M. Compensation of geometric, backlash, and thermal drift errors using a universal industrial robot model. *IEEE Transactions on Automation Science and Engineering* **2023**, *21*, 6615–6627.
96. Sakayori, G.; Ishigami, G. Modeling of slip rate-dependent traversability for path planning of wheeled mobile robot in sandy terrain. *Frontiers in Robotics and AI* **2024**, *11*, 1320261.
97. Tölle, M.; Gruner, T.; Palenicek, D.; Günster, J.; Liu, P.; Watson, J.; Tateo, D.; Peters, J. Towards Safe Robot Foundation Models. *arXiv preprint arXiv:2503.07404* **2025**.
98. Yang, S.; Nachum, O.; Du, Y.; Wei, J.; Abbeel, P.; Schuurmans, D. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129* **2023**.
99. Sharifi, S.; Stocco, A.; Briand, L.C. System safety monitoring of learned components using temporal metric forecasting. *ACM Transactions on Software Engineering and Methodology* **2025**, *34*, 1–43.
100. Mirzaeian, A.; Kosecka, J.; Homayoun, H.; Mohsenin, T.; Sasan, A. Diverse knowledge distillation (dkd): A solution for improving the robustness of ensemble models against adversarial attacks. In Proceedings of the 2021 22nd International Symposium on Quality Electronic Design (ISQED). IEEE, 2021, pp. 319–324.
101. Shen, Y.; Vander Mynsbrugge, J.; Roshandel, N.; Bouchez, R.; FirouziPouyaei, H.; Scholz, C.; Cao, H.L.; Vanderborght, B.; Joosen, W.; Paolillo, A. SentryRT-1: A Case Study in Evaluating Real-Time Linux for Safety-Critical Robotic Perception. In Proceedings of the Proceedings of the 19th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2025), 2025, pp. 35–41.
102. Sobhani, H.; Choi, H.; Kim, H. Timing analysis and priority-driven enhancements of ROS 2 multi-threaded executors. In Proceedings of the 2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE, 2023, pp. 106–118.
103. Gomes, C.P.; Selman, B.; Crato, N.; Kautz, H. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of automated reasoning* **2000**, *24*, 67–100.
104. Park, J.; Kim, P.; Ko, D. Real-time open-vocabulary perception for mobile robots on edge devices: a systematic analysis of the accuracy-latency trade-off. *Frontiers in Robotics and AI* **2025**, *12*, 1693988.
105. Navardi, M.; Aalishah, R.; Fu, Y.; Lin, Y.; Li, H.; Chen, Y.; Mohsenin, T. GenAI at the edge: Comprehensive survey on empowering edge devices. In Proceedings of the Proceedings of the AAAI Symposium Series, 2025, Vol. 5, pp. 180–187.

 doi:10.20944/preprints202603.1293.v1

106. Maruyama, Y.; Kato, S.; Azumi, T. Exploring the performance of ROS2. In Proceedings of the Proceedings of the 13th international conference on embedded software, 2016, pp. 1–10.
107. Sünderhauf, N.; Brock, O.; Scheirer, W.; Hadsell, R.; Fox, D.; Leitner, J.; Upcroft, B.; Abbeel, P.; Burgard, W.; Milford, M.; et al. The limits and potentials of deep learning for robotics. *The International journal of robotics research* **2018**, *37*, 405–420.
108. Shi, Y.; Yang, K.; Jiang, T.; Zhang, J.; Letaief, K.B. Communication-efficient edge AI: Algorithms and systems. *IEEE communications surveys & tutorials* **2020**, *22*, 2167–2191.
109. Ishikawa-Aso, T.; Kato, S. ROS 2 Agnocast: Supporting unsized message types for true zero-copy publish/subscribe IPC, 2025.
110. Skevik, K.A.; Plagemann, T.; Goebel, V.; Halvorsen, P. Evaluation of a zero-copy protocol implementation. In Proceedings of the Proceedings 27th EUROMICRO Conference. 2001: A Net Odyssey. IEEE, 2001, pp. 324–330.
111. Abaza, H.; Roy, D.; Trach, B.; Chang, W.; Saidi, S.; Motakis, A.; Ren, W.; Liu, Y. Managing End-to-End Timing Jitters in ROS2 Computation Chains. In Proceedings of the International Conference on Real-Time Networks and Systems (RTNS), 2024.
112. Salimpour, S.; Peña-Queralta, J.; Paez-Granados, D.; Heikkonen, J.; Westerlund, T. Sim-to-real transfer for mobile robots with reinforcement learning: from nvidia isaac sim to gazebo and real ros 2 robots. *arXiv preprint arXiv:2501.02902* **2025**.
113. Jiang, Y.; Cheng, S.; Ding, Y.; Gao, F.; Qi, B. AsyncVLA: Asynchronous Flow Matching for Vision Language Action Models, 2025, [arXiv:cs.RO/2511.14148].
114. Yang, F.; Yu, B.; Zhou, Y.; Luo, X.; Tu, Z.; Liu, C. Edge-based multimodal sensor data fusion with vision language models (vlms) for real-time autonomous vehicle accident avoidance, 2025.
115. Zhang, B.; Liu, R.W.; Liu, J.; Chui, K.T.; Gupta, B.B. Multi-Sensor Data Fusion Meets Edge Computing for Intelligent Surface Vehicles. *IEEE Internet of Things Magazine* **2025**.
116. Lou, Y.; Song, Q.; Xu, Q.; Tan, R.; Wang, J. Uncertainty-encoded multi-modal fusion for robust object detection in autonomous driving, 2023.
117. Cho, M.; Cao, Y.; Sun, J.; Zhang, Q.; Pavone, M.; Park, J.J.; Yang, H.; Mao, Z.M. Cocoon: Robust multi-modal perception with uncertainty-aware sensor fusion, 2024.
118. Park, K.; Kim, Y.; Kim, D.; Choi, J.W. Resilient sensor fusion under adverse sensor failures via multi-modal expert fusion, 2025.
119. Brödermann, T.; Sakaridis, C.; Fu, Y.; Van Gool, L. Cafuser: Condition-aware multimodal fusion for robust semantic perception of driving scenes, 2025.
120. Song, Z.; Peng, L.; Hu, J.; Yao, D.; Zhang, Y. Timealign: A multi-modal object detection method for time misalignment fusing in autonomous driving, 2024.
121. Wang, S.; Caesar, H.; Nan, L.; Kooij, J.F. Unibev: Multi-modal 3d object detection with uniform bev encoders for robustness against missing sensor modalities, 2024.
122. Liu, G.H.; Siravuru, A.; Prabhakar, S.; Veloso, M.; Kantor, G. Learning end-to-end multimodal sensor policies for autonomous navigation. In Proceedings of the Conference on robot learning. PMLR, 2017, pp. 249–261.
123. Yang, K.; Lin, W.Y.; Barman, M.; Condessa, F.; Kolter, Z. Defending multimodal fusion models against single-source adversaries, 2021.
124. Wali, K. Hardware-software co-design for power-efficient edge-AI systems. *J Artif Intell Mach Learn Data Sci* **2024**, *2*, 2754–60.
125. Hamdi, M.A.; Daghero, F.; Sarda, G.M.; Van Delm, J.; Symons, A.; Benini, L.; Verhelst, M.; Pagliari, D.J.; Burrello, A. MATCH: model-aware TVM-based compilation for heterogeneous edge devices, 2025.
126. Gross, W.J.; Meyer, B.H.; Ardakani, A. Hardware-aware design for edge intelligence. *IEEE Open Journal of Circuits and Systems* **2020**, *2*, 113–127.
127. Lv, K.; Guo, H.; Guo, Q.; Qiu, X. DuoDecoding: Hardware-aware Heterogeneous Speculative Decoding with Dynamic Multi-Sequence Drafting. *arXiv preprint arXiv:2503.00784* **2025**.
128. Liu, H.I.C.; Brehler, M.; Ravishankar, M.; Vasilache, N.; Vanik, B.; Laurenzo, S. TinyIREE: An ML execution environment for embedded systems from compilation to deployment, 2022.
129. Rognlien, M.; Que, Z.; Coutinho, J.G.; Luk, W. Hardware-aware optimizations for deep learning inference on edge devices. In Proceedings of the International Symposium on Applied Reconfigurable Computing. Springer, 2022, pp. 118–133.
130. Wei, J.; Li, Q.; Cao, S.; Ma, L.; Hao, Z.; Zhang, Y.; Hu, X.; Cao, T. T-MAN: Enabling End-to-End Low-Bit LLM Inference on NPUs via Unified Table Lookup, 2025.

 doi:10.20944/preprints202603.1293.v1

131. Park, T.; Lee, G.; Kim, M.S. MobileRAG: A Fast, Memory-Efficient, and Energy-Efficient Method for On-Device RAG. *arXiv preprint arXiv:2507.01079* **2025**.
132. Dettmers, T.; Lewis, M.; Belkada, Y.; Zettlemoyer, L. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale, 2022.
133. Frantar, E.; Ashkboos, S.; Hoefler, T.; Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2022.
134. Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.M.; Wang, W.C.; Xiao, G.; Dang, X.; Gan, C.; Han, S. Awq: Activation-aware weight quantization for on-device llm compression and acceleration, 2024.
135. Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.
136. Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691* **2023**.
137. Gupta, A.; Guo, H.; Yuan, Y.; Zhou, Y.; Mendis, C. FLuRKA: Fast and accurate unified low-rank & kernel attention. *arXiv preprint arXiv:2306.15799* **2023**.
138. Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C.H.; Gonzalez, J.; Zhang, H.; Stoica, I. Efficient memory management for large language model serving with pagedattention, 2023.
139. Smith, J.S.; Lin, C.H.; Tuli, S.; Jeelani, H.; Gao, S.; Shen, Y.; Jin, H.; Hsu, Y.C. FlexiGPT: Pruning and Extending Large Language Models with Low-Rank Weight Sharing. *arXiv preprint arXiv:2501.14713* **2025**.
140. Leal, I.; Choromanski, K.; Jain, D.; Dubey, A.; Varley, J.; Ryoo, M.; Lu, Y.; Liu, F.; Sindhwani, V.; Vuong, Q.; et al. Sara-rt: Scaling up robotics transformers with self-adaptive robust attention, 2024.
141. Valsan, P.K.; Yun, H.; Farshchi, F. Addressing isolation challenges of non-blocking caches for multicore real-time systems. *Real-Time Systems* **2017**, *53*, 673–708.
142. Yun, H.; Yao, G.; Pellizzoni, R.; Caccamo, M.; Sha, L. Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms. In Proceedings of the 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE, 2013, pp. 55–64.
143. Yun, H.; Ali, W.; Gondi, S.; Biswas, S. BWLOCK: A dynamic memory access control framework for soft real-time applications on multicore platforms. *IEEE Transactions on Computers* **2016**, *66*, 1247–1252.
144. Aghilinasab, H.; Ali, W.; Yun, H.; Pellizzoni, R. Dynamic memory bandwidth allocation for real-time GPU-based SoC platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2020**, *39*, 3348–3360.
145. Fang, J.; Lin, S.; Yan, Y. RT-DBR: Dynamic Bandwidth Reservation for Real-Time Application on CPU-GPU Heterogeneous SoC. In Proceedings of the Proceedings of the 2024 8th International Conference on Computer Science and Artificial Intelligence, 2024, pp. 594–600.
146. Puente, J.A.; Mezzetti, E.; Troncoso, I.A.; Ferrer, J.A.; Almeida, F.J.C. ROSGuard: A Bandwidth Regulation Mechanism for ROS2-based Applications. *arXiv preprint arXiv:2506.04640* **2025**.
147. Sullivan, C.; Manley, A.; Alian, M.; Yun, H. Per-Bank Bandwidth Regulation of Shared Last-Level Cache for Real-Time Systems. In Proceedings of the 2024 IEEE Real-Time Systems Symposium (RTSS). IEEE, 2024, pp. 336–348.
148. Zuepke, A.; Bastoni, A.; Chen, W.; Caccamo, M.; Mancuso, R. Mempol: polling-based microsecond-scale per-core memory bandwidth regulation. *Real-Time Systems* **2024**, *60*, 369–412.
149. Chen, T.A.; Yang, D.N.; Chen, M.S. Overcoming Forgetting Catastrophe in Quantization-Aware Training. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2023, pp. 17358–17367.
150. Cai, Y.; Yao, Z.; Dong, Z.; Gholami, A.; Mahoney, M.W.; Keutzer, K. Zeroq: A novel zero shot quantization framework. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 13169–13178.
151. Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; Shao, L. Hrank: Filter pruning using high-rank feature map. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 1529–1538.
152. Li, W.; He, Y.; Zhang, X.; Tang, Y. Filter pruning via feature map clustering. *Intelligent data analysis* **2023**, *27*, 911–933.
153. Liu, Z.; Mu, H.; Zhang, X.; Guo, Z.; Yang, X.; Cheng, K.T.; Sun, J. Metapruning: Meta learning for automatic neural network channel pruning. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 3296–3305.

 doi:10.20944/preprints202603.1293.v1

154. Lin, S.; Ji, R.; Yan, C.; Zhang, B.; Cao, L.; Ye, Q.; Huang, F.; Doermann, D. Towards optimal structured cnn pruning via generative adversarial learning. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 2790–2799.
155. Ashkboos, S.; Croci, M.L.; Nascimento, M.G.d.; Hoefler, T.; Hensman, J. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024* **2024**.
156. Chen, M.; Shao, W.; Xu, P.; Wang, J.; Gao, P.; Zhang, K.; Luo, P. Efficientqat: Efficient quantization-aware training for large language models. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 10081–10100.
157. Ashkboos, S.; Mohtashami, A.; Croci, M.L.; Li, B.; Cameron, P.; Jaggi, M.; Alistarh, D.; Hoefler, T.; Hensman, J. Quarot: Outlier-free 4-bit inference in rotated llms, 2024.
158. Liu, Z.; Zhao, C.; Fedorov, I.; Soran, B.; Choudhary, D.; Krishnamoorthi, R.; Chandra, V.; Tian, Y.; Blankevoort, T. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406* **2024**.
159. Liu, Z.; Yuan, J.; Jin, H.; Zhong, S.; Xu, Z.; Braverman, V.; Chen, B.; Hu, X. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750* **2024**.
160. Zhang, T.; Yi, J.; Xu, Z.; Shrivastava, A. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization, 2024.
161. Hooper, C.; Kim, S.; Mohammadzadeh, H.; Mahoney, M.W.; Shao, Y.S.; Keutzer, K.; Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization, 2024.
162. Shutova, A.; Malinovskii, V.; Egiazarian, V.; Kuznedelev, D.; Mazur, D.; Surkov, N.; Ermakov, I.; Alistarh, D. Cache me if you must: Adaptive key-value quantization for large language models, 2025.
163. Tomar, A.; Hooper, C.; Lee, M.; Xi, H.; Tiwari, R.; Kang, W.; Manolache, L.; Mahoney, M.W.; Keutzer, K.; Gholami, A. XQuant: Breaking the Memory Wall for LLM Inference with KV Cache Rematerialization, 2025.
164. Boroujerdian, B.; Genc, H.; Krishnan, S.; Cui, W.; Faust, A.; Reddi, V. Mavbench: Micro aerial vehicle benchmarking. In Proceedings of the 2018 51st annual IEEE/ACM international symposium on microarchitecture (MICRO). IEEE, 2018, pp. 894–907.
165. Chen, Y.H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits* **2016**, *52*, 127–138.
166. Koppula, S.; Orosa, L.; Yağlıkçı, A.G.; Azizi, R.; Shahroodi, T.; Kanellopoulos, K.; Mutlu, O. EDEN: Enabling energy-efficient, high-performance deep neural network inference using approximate DRAM. In Proceedings of the Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 2019, pp. 166–181.
167. Lee, Y. Thermal-aware design and management of embedded real-time systems. In Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021, pp. 1252–1255.
168. Lin, C.; Wang, K.; Li, Z.; Pu, Y. A workload-aware DVFS robust to concurrent tasks for mobile devices. In Proceedings of the Proceedings of the 29th Annual International Conference on Mobile Computing and Networking, 2023, pp. 1–16.
169. Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; Zhou, D. Mobilebert: a compact task-agnostic bert for resource-limited devices, 2020.
170. Yao, Z.; Yazdani Aminabadi, R.; Zhang, M.; Wu, X.; Li, C.; He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers, 2022.
171. Xiao, G.; Lin, J.; Seznec, M.; Wu, H.; Demouth, J.; Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models, 2023.
172. Dowling, A.; Jiang, L.; Cheng, M.C.; Liu, Y. Regulating CPU temperature with thermal-aware scheduling using a reduced order learning thermal model, 2025.
173. Jacob, M.N. ThermML@ Edge: Thermally-Robust Inference on Commodity SBCs via Precision, Threading, and Policy Design. *Authorea Preprints* **2025**.
174. Buchanan, R.; Agrawal, V.; Camurri, M.; Dellaert, F.; Fallon, M. Deep imu bias inference for robust visual-inertial odometry with factor graphs. *IEEE Robotics and Automation Letters* **2022**, *8*, 41–48.
175. Altawaitan, A.; Stanley, J.; Ghosal, S.; Duong, T.; Atanasov, N. Learned IMU Bias Prediction for Invariant Visual Inertial Odometry. *arXiv preprint arXiv:2505.06748* **2025**.
176. Kushawaha, N.; Pathan, R.; Pagliarani, N.; Cianchetti, M.; Falotico, E. Adaptive Drift Compensation for Soft Sensorized Finger Using Continual Learning. In Proceedings of the 2025 IEEE 8th International Conference on Soft Robotics (RoboSoft). IEEE, 2025, pp. 1–6.
177. George Thuruthel, T.; Gardner, P.; Iida, F. Closing the control loop with time-variant embedded soft sensors and recurrent neural networks. *Soft Robotics* **2022**, *9*, 1167–1176.

 doi:10.20944/preprints202603.1293.v1

178. Niu, S.; Wu, J.; Zhang, Y.; Chen, Y.; Zheng, S.; Zhao, P.; Tan, M. Efficient test-time model adaptation without forgetting. In Proceedings of the International conference on machine learning. PMLR, 2022, pp. 16888–16905.
179. Niu, S.; Wu, J.; Zhang, Y.; Wen, Z.; Chen, Y.; Zhao, P.; Tan, M. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400* **2023**.
180. Gong, T.; Jeong, J.; Kim, T.; Kim, Y.; Shin, J.; Lee, S.J. Note: Robust continual test-time adaptation against temporal correlation. 2022, Vol. 35, pp. 27253–27266.
181. Zhu, L.; Hu, L.; Lin, J.; Chen, W.M.; Wang, W.C.; Gan, C.; Han, S. PockEngine: Sparse and Efficient Fine-tuning in a Pocket. *2023 56th IEEE/ACM International Symposium on Microarchitecture (MICRO)* **2023**, pp. 1381–1394.
182. Cai, H.; Gan, C.; Zhu, L.; Han, S. Tinytl: Reduce memory, not parameters for efficient on-device learning. 2020, Vol. 33, pp. 11285–11297.
183. Zaken, E.B.; Goldberg, Y.; Ravfogel, S. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models **2022**. pp. 1–9.
184. Niu, S.; Miao, C.; Chen, G.; Wu, P.; Zhao, P. Test-time model adaptation with only forward passes. *arXiv preprint arXiv:2404.01650* **2024**.
185. O'Connell, M.; Shi, G.; Shi, X.; Azizzadenesheli, K.; Anandkumar, A.; Yue, Y.; Chung, S.J. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics* **2022**, *7*, eabm6597.
186. Ames, A.D.; Coogan, S.; Egerstedt, M.; Notomista, G.; Sreenath, K.; Tabuada, P. Control barrier functions: Theory and applications. In Proceedings of the 2019 18th European control conference (ECC). Ieee, 2019, pp. 3420–3431.
187. Wabersich, K.P.; Zeilinger, M.N. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica* **2021**, *129*, 109597.
188. Tearle, B.; Wabersich, K.P.; Carron, A.; Zeilinger, M.N. A predictive safety filter for learning-based racing control. *IEEE Robotics and Automation Letters* **2021**, *6*, 7635–7642.
189. Fisac, J.F.; Akametalu, A.K.; Zeilinger, M.N.; Kaynama, S.; Gillula, J.; Tomlin, C.J. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control* **2018**, *64*, 2737–2752.
190. Cheng, R.; Orosz, G.; Murray, R.M.; Burdick, J.W. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2019, Vol. 33, pp. 3387–3395.
191. Xiao, W.; Wang, T.H.; Gan, C.; Hasani, R.; Lechner, M.; Rus, D. Safediffuser: Safe planning with diffusion probabilistic models. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2023.
192. Desai, A.; Ghosh, S.; Seshia, S.A.; Shankar, N.; Tiwari, A. SOTER: a runtime assurance framework for programming safe robotics systems. In Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2019, pp. 138–150.
193. Chen, S.; Sun, Y.; Li, D.; Wang, Q.; Hao, Q.; Sifakis, J. Runtime safety assurance for learning-enabled control of autonomous driving vehicles. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022, pp. 8978–8984.
194. Katz, G.; Barrett, C.; Dill, D.L.; Julian, K.; Kochenderfer, M.J. Reluplex: An efficient SMT solver for verifying deep neural networks. In Proceedings of the International conference on computer aided verification. Springer, 2017, pp. 97–117.
195. Katz, G.; Huang, D.A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. The marabou framework for verification and analysis of deep neural networks. In Proceedings of the International conference on computer aided verification. Springer, 2019, pp. 443–452.
196. Ruan, W.; Huang, X.; Kwiatkowska, M. Reachability analysis of deep neural networks with provable guarantees. *arXiv preprint arXiv:1805.02242* **2018**.
197. Tayal, M.; Singh, A.; Jagtap, P.; Kolathaya, S. Cp-ncbf: A conformal prediction-based approach to synthesize verified neural control barrier functions. *arXiv preprint arXiv:2503.17395* **2025**.
198. Zhang, J.; Hoxha, B.; Fainekos, G.; Panagou, D. Conformal Prediction in the Loop: Risk-Aware Control Barrier Functions for Stochastic Systems with Data-Driven State Estimators. *IEEE Control Systems Letters* **2025**.

 doi:10.20944/preprints202603.1293.v1

199. Jansen, N.; Könighofer, B.; Junges, S.; Serban, A.; Bloem, R. Safe reinforcement learning using probabilistic shields. In Proceedings of the 31st International Conference on Concurrency Theory (CONCUR 2020). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 3–1.
200. Corsi, D.; Amir, G.; Rodríguez, A.; Sánchez, C.; Katz, G.; Fox, R. Verification-guided shielding for deep reinforcement learning. *arXiv preprint arXiv:2406.06507* **2024**.
201. Zolfagharian, A.; Abdellatif, M.; Briand, L.C.; et al. Smarla: A safety monitoring approach for deep reinforcement learning agents. *IEEE Transactions on Software Engineering* **2024**.
202. Casini, D.; Blaß, T.; Lütkebohle, I.; Brandenburg, B. Response-time analysis of ROS 2 processing chains under reservation-based scheduling. In Proceedings of the 31st Euromicro Conference on Real-Time Systems. Schloss Dagstuhl, 2019, pp. 1–23.
203. Choi, H.; Xiang, Y.; Kim, H. PiCAS: New design of priority-driven chain-aware scheduling for ROS2. In Proceedings of the 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE, 2021, pp. 251–263.
204. Gutiérrez, C.S.V.; Juan, L.U.S.; Ugarte, I.Z.; Vilches, V.M. Real-time Linux communications: an evaluation of the Linux communication stack for real-time robotic applications. *arXiv preprint arXiv:1808.10821* **2018**.
205. Baig, M.A.; Han, L.; Yang, D.; Danaish. A cloud robotics architecture for greenhouse strawberry harvesting with single-pass data collection, 2026.
206. Pradhan, A.; Ottaviano, D.; Jiang, Y.; Huang, H.; Zhang, J.; Zuepke, A.; Bastoni, A.; Caccamo, M. Predictable Memory Bandwidth Regulation for DynamIQ Arm Systems. In Proceedings of the 2025 IEEE 31st International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). IEEE, 2025, pp. 126–137.
207. Pöhnl, M.; Eltzschig, C.; Blass, T. Shared-memory-based lock-free queues: The key to fast and robust communication on safety-critical edge devices, 2023.
208. Bell, O.; Gill, C.; Zhang, X. Hardware acceleration with zero-copy memory management for heterogeneous computing. In Proceedings of the 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). IEEE, 2023, pp. 28–37.
209. Mayoral-Vilches, V.; Reina-Muñoz, J.M.; Crespo-Álvarez, M.; Mayoral-Vilches, D. ROS 2 on a Chip, Achieving Brain-Like Speeds and Efficiency in Robotic Networking. *arXiv preprint arXiv:2404.18208* **2024**.
210. Kang, Y.; Hauswald, J.; Gao, C.; Rovinski, A.; Mudge, T.; Mars, J.; Tang, L. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News* **2017**, *45*, 615–629.
211. Teerapittayanon, S.; McDanel, B.; Kung, H.T. Distributed deep neural networks over the cloud, the edge and end devices. In Proceedings of the 2017 IEEE 37th international conference on distributed computing systems (ICDCS). IEEE, 2017, pp. 328–339.
212. Eshratifar, A.E.; Esmaili, A.; Pedram, M. Bottlenet: A deep learning architecture for intelligent mobile cloud computing services. In Proceedings of the 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, 2019, pp. 1–6.
213. Zhang, W.; He, Z.; Liu, L.; Jia, Z.; Liu, Y.; Gruteser, M.; Raychaudhuri, D.; Zhang, Y. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In Proceedings of the Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, 2021, pp. 201–214.
214. Li, P.; Aijaz, A. Task-oriented connectivity for networked robotics with generative ai and semantic communications, 2025.
215. Ali, A.H.; Navardi, M.; Mohsenin, T. Energy-Aware FPGA Implementation of Spiking Neural Network with LIF Neurons. *arXiv preprint arXiv:2411.01628* **2024**.
216. Zheng, X.; Liu, Y.; Lu, Y.; Hua, T.; Pan, T.; Zhang, W.; Tao, D.; Wang, L. Deep learning for event-based vision: A comprehensive survey and benchmarks. *arXiv preprint arXiv:2302.08890* **2023**.
217. Woodward, K.; Kanjo, E.; Papandroulidakis, G.; Agwa, S.; Prodromakis, T. A Hybrid Edge Classifier: Combining TinyML-Optimised CNN With RRAM-CMOS ACAM for Energy-Efficient Inference. *IEEE Transactions on Knowledge and Data Engineering* **2026**, *38*, 2122–2135. https://doi.org/10.1109/TKDE.2026.3655717.
218. Aalishah, R.; Navardi, M.; Mohsenin, T. EdgeNavMamba: Mamba Optimized Object Detection for Energy Efficient Edge Devices. *arXiv preprint arXiv:2510.14946* **2025**.
219. Wu, P.; Escontrela, A.; Hafner, D.; Abbeel, P.; Goldberg, K. Daydreamer: World models for physical robot learning. In Proceedings of the Conference on robot learning. PMLR, 2023, pp. 2226–2240.

 doi:10.20944/preprints202603.1293.v1

220. Feng, T.; Wang, W.; Yang, Y. A survey of world models for autonomous driving. *arXiv preprint arXiv:2501.11260* **2025**.
221. Tölle, M.; Grüner, T.; Palenicek, D.; et al. Runtime Monitoring for Safe Deep Learning Systems: A Survey. *ACM Computing Surveys* **2023**, *56*, 1–38.
222. Wu, T.; Song, C.; Zeng, P. Efficient federated learning on resource-constrained edge devices based on model pruning. *Complex & Intelligent Systems* **2023**, *9*, 6999–7013.